Final Design Document

1)
Feature List

- user accounts on the game server
- user login/logout
- create a game room
- list online users in the system
- invite users to a game room
- join or leave a game room
- chat while playing the game (public and private messaging)
- interactive game playing (realtime update of game status to all players)
- handle player disconnection and reconnection (pause and resume game)
- allow observers in a game room

Additional features:

- Player can communicate via multiple channels
- Player stats and information are associated with their account
- Players can send messages to friends outside of the game

2)

System Overview: The the key motivation for our project is to reap the benefit of digitizing the Mafia party game. As mentioned in the initial proposal, the in-person or “party” version of the game (as it will be known from now on) suffers from some shortcomings by playing in person. As mafia is a game which combines both open discourse and deceit, combined with multiple “teams”, there needs to be a person filling a moderator role instead of enjoying the game themselves. This role can be immediately filled by the computer system, meaning all individuals can enjoy the game as players. Also, by removing the number of human actors we can generate a more efficient and error free system than one comprised of a human moderator, especially beneficial to a new player group. As players are communicating from remote locations via a computer network, digitizing the game also removes the possibility of dishonest play stemming from players peeking to determine another player's role and ensures that no player is revealed by the sound of their actions. There is also added functionality which can be easily added to the digital version relative to the party version. For instance, after several rounds of playing the party version with the same group of people, players may begin to notice “tells” in each other's behaviour, giving them a non-logic based reason for suspecting someone’s intentions. The digital version can offset this behaviour by allowing players to select a pseudonym per round to disguise their identity anew. Likewise, in the party version when a player is killed, they have no more interest in the game, as they cannot communicate with anyone else until the game is over. In the digital version, a very simple “graveyard” chat can be implemented, which allows the dead players to discuss the ongoing game freely without their input being seen by the still living

players. This allows time for the dead to analyze and comment on the game as it continues, prolonging their interest. This idea can also be extended to the mafiosos to allow them to communicate during the nighttime, something impossible to achieve in the party version. Finally, digitizing the game allows for it to be enjoyed by a group of people who may not be able to meet in person and can have more appeal for younger generations.

The key objectives of the project are:

- Remove the role of a human moderator
- Create a more efficient experience than one which is run by a human moderator
- Ensure the anonymity of the players
- Allow for multiple conversation channels

The scope for this project is:

- Create a moderation system which records and displays all game state information to the players in an easy to understand and correct fashion. This system will initiate the game session, display all user conversation and handle all voting. It will also monitor the game state and end the round when either the innocent or Mafia win conditions are met.
- Generate an initial system state which allows each player to set a new pseudonym
- Allow Mafia to communicate via a Mafia only channel during the nighttime and allow the dead players to communicate during with each other once they are killed.

System Analysis: The system, being a text based game, can tolerate short delays since each player is only sending short messages. The delay can't be too long because the players will become bored waiting or other players won't see their opinion on who to "murder" or "lynch" before the vote. Likewise, because discussion and debate are they key activities in the game, people will need the input from the other players quickly to analyze their arguments. The system cannot tolerate loss in order to be fair to all players, as everyone's input both in conversation and in voting must be taken into consideration. If a player votes for a certain outcome or brings forward certain evidence and the system loses this information, it will not be a fair game. The system requires little network throughput because the hosts will only be sending small sized messages in the form of text commands and text discussions. Considering this analysis it makes sense to implement the system's transport-layer protocol using TCP. TCP provides the necessary reliability and our game can handle the small delay that TCP will have over UDP due to the low system throughput.

Use Cases:

Name	Create a player account

Actor	Player	
Goal	Create a persistent account to connect to games of Mafia	
Precondition	Player has launched Mafia for the first time	
Steps	**Actor**	**Server**
	2. Player types /new and enters their username and password	1. Prompt player to enter a username and password or create a new account 3a. Server responds that the username is taken, and to enter a new username 3b. Server responds that the account has been created
Postcondition	Login info is recorded	

Name	Login to account	
Actor	Player	
Goal	Login to the player's persistent account to begin playing Mafia	
Precondition	Player has launched Mafia for not the first time	
Steps	**Actor**	**Server**
	2. Player enters their username and password	1. Prompt player to enter a username and password or create a new account 3a. Server responds that the connection was successful and sends the user to the game lobby 3b. Server responds that the password/username is incorrect and prompts to try again

Postcondition	

Name	Logout of account	
Actor	Player	
Goal	Logout of account to exit Mafia	
Precondition	Player is logged into Mafia	
Steps	**Actor**	**Server**
	1. Player types /logout into the text channel 3. Player confirms logout with y selection	2. Server prompts with y/n options to confirm logout 4. Player is logged out, Mafia closes
Postcondition	Player is no longer logged in	

Name	Prompt user to create a pseudonym	
Actor	Players	
Goal	Create a *unique* pseudonym	
Precondition	Player has launched Mafia and connected to a game	
Steps	**Actor**	**Server**
	2. Player enters their pseudonym	1. Prompt player to enter a *pseudonym* 3. Server acknowledgement, “pseudonym accepted”
Postcondition	Player is in the world of **Mafia**	

Name	Message sent / message received
Actor	Players

Goal	To enable the **innocent** and the **mafioso** to interact with their respective groups	
Precondition	Player has successfully connected to the Mafia game and *created* a pseudonym	
Steps	**Actor**	**Server**
	1. Enters message/command they wish to send in the query text area	2a. Displays message on the main thread 2b. Performs action defined by the player's entered message/command and displays the outcome of the player entered command
Postcondition	Player can see what the outcome of their actions is	

Name	Voting to lynch player during daytime	
Actor	Player	
Goal	Begin a vote to decide whom to *lynch*	
Precondition	Game is in play and it must be the daytime cycle	
Steps	**Actor**	**Server**
	1. A player begins the *lynch* 3. Other players can join in and vote on *lynching*	2. Player's pseudonym and whom the player has voted to lynch is displayed to the public 4a. When enough votes have been cast on a player, the player is killed, and their name is displayed to the public 4b. The vote times out if not enough players vote
Postcondition	Player can see the current status of his vote	

Name	Voting to murder player during nighttime	
Actor	Player	
Goal	Begin a vote to decide whom to *murder*	
Precondition	Game is in play and it must be the night time cycle and the player is a mafioso.	
Steps	**Actor**	**Server**
	1. A player begins the *murder* vote 3. Other players can join in and vote on the *murder*	2. Player's pseudonym and whom the player has voted to murder is displayed to the other mafiosos. 4a. When enough votes have been cast on a player, the player is murdered 4b. The vote times out if not enough players vote
Postcondition	If a player has been murdered, their name is displayed at the beginning of the next day.	

Name	Invite users to a game room	
Actor	PlayerA	
Goal	PlayerB recieves an invititation to join a game room.	
Precondition	PlayerA and PlayerB are logged into the server.	
Steps	**Actor**	**Server**
	1. PlayerA types in "/invite name". (name is PlayerB's account name) 3. PlayerB types "/accept PlayerA" to accept PlayerA's invitation and join the room.	2. Sends PlayerB the message "PlayerA has invited you to game room X" 4. Server moves PlayerB to the

		room of PlayerA's invite.
Postcondition	PlayerA and PlayerB are in the same game room.	

Name	Allow observers in game	
Actor	Player (observers)	
Goal	Allow user to be able to view the ongoing Mafia game	
Precondition	A Mafia game must have already begun	
Steps	**Actor**	**Server**
	1. Potential player asks to be able to join the ongoing game as an observer	2. Accept the player's request to join the game as an observer and allows the user to be able to view the main window of the game, and communicate with the dead players.
Postcondition	Player can now view the ongoing Mafia game	

Name	List online users in the system	
Actor	Player	
Goal	Allow the potential players to view the Mafia players	
Precondition	The player must be logged into the server and not in a game	
Steps	**Actor**	**Server**
	1. Inputs command that allows user to view the users in the system	2. Server accepts the command and begins displaying the users in the system
Postcondition	User is now able to view the users in the system	

Name	Handle player disconnection and reconnection	
Actor	Player	
Goal	Allow a player to reconnect to a game after disconnecting.	
Precondition	Player is in an active game.	
Steps	**Actor**	**Server**
	1. Player disconnects from server via /quit. 3. Player logs into the server, and re-joins the game room.	2. Server pauses the game and notifies all players in the room: “Player has disconnected: game paused. 4. Server unpauses the game and notifies all the players in the room: “Player has reconnected: game resumed.”
Postcondition	Server resumes gameplay.	

Name	Join game room	
Actor	Player	
Goal	Give player ability to join the game room	
Precondition	A game must have been established	
Steps	**Actor**	**Server**
	1. Player decides to join an arbitrary game room	2. The server acknowledges the request and joins the player to the game specified game
Postcondition	Player is now part of a Mafia game	

Name	Leave game room
Actor	Player

Goal	Give player ability to leave the game room	
Precondition	Player must be part of an ongoing game	
Steps	**Actor**	**Server**
	1. Player decides to leave the current game	2. The server acknowledges the request and boots the player to the game lobby
Postcondition	Player is now in the *game lobby*	

Message Format:

Every outgoing client message will specify what type of command it is (a byte id), the size of the attached data array and the data array itself.

Client Message format: [byte type, int dataSize, byte[] data]

Type	data format
0: CreateAccount	= [string username, string password]
1: Login	= [string username, string password]
2: Logout	= [ ]
3: SetAlias	= [string pseudo]
4: Join	= [byte roomId]
5: Invite	= [string username]
6: ListUsers	= [ ]
7: ListRooms	= [ ]
8: Chat	= [string message]
9: Vote	= [string username]
10: GetGameStatus	= [ ]

Each outgoing server message will have an associated type (byte id), a display string length along with the string that will be shown on the client side, the size of data attached, and the data array.

Server message format: [byte type, int msgLength, int dataSize, string[] msg, byte[] data]

Type	data format
11: ServerMessage	= [ ]
12: InviteNotify	= [string username, byte roomId]

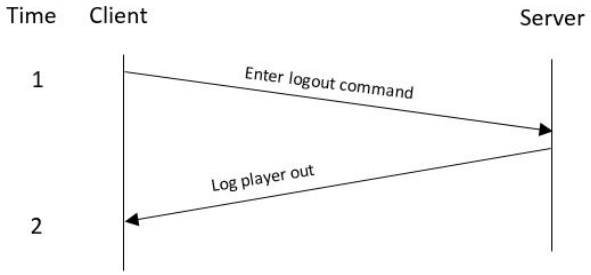
(notify some player that username invited them)

13: ListUsers = [int numUsers, string[] usernames]
14: ListRooms = [(byte roomId, numPlayers, maxPlayers, active), …]
15: Acknowledge = [bool success, string message]
16: StatusChange = [string message]
17. BanUser = [string message]

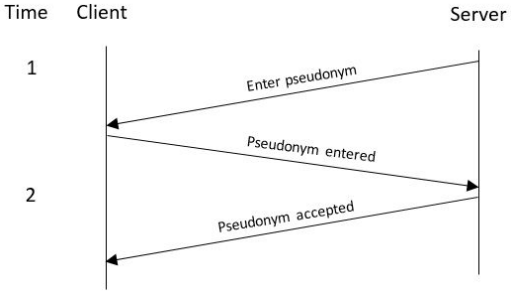
Order of Message Exchange:

Create a player account: ***Login to account:***

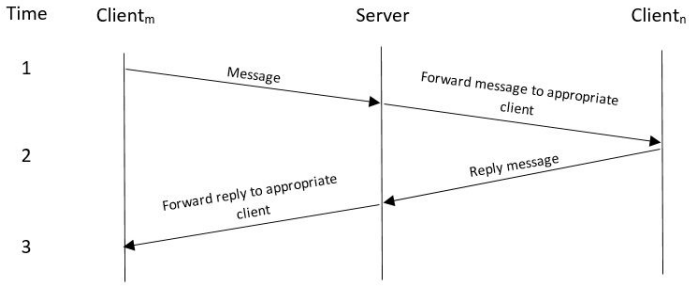
Logout of account:



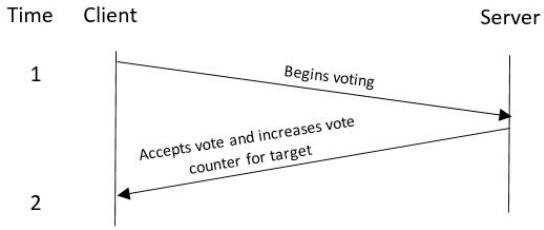
Create pseudonym:



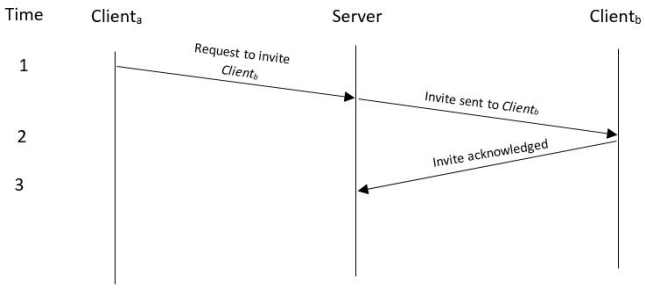
Message sent / received:



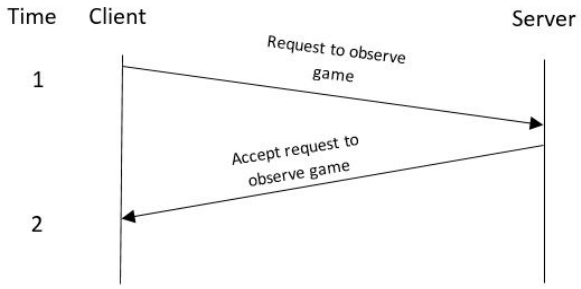
Vote to lynch player (daytime/night time):



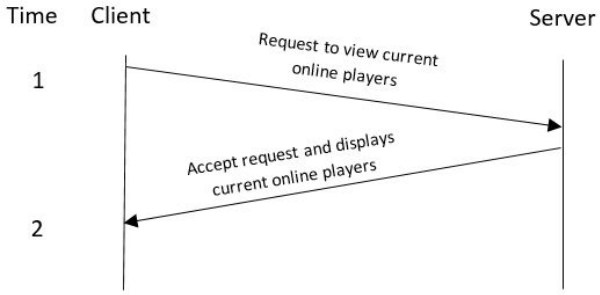
Game invitation:



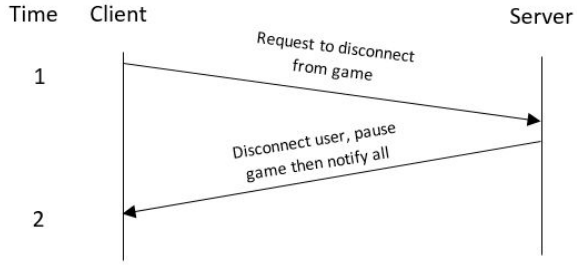
Request to observe:



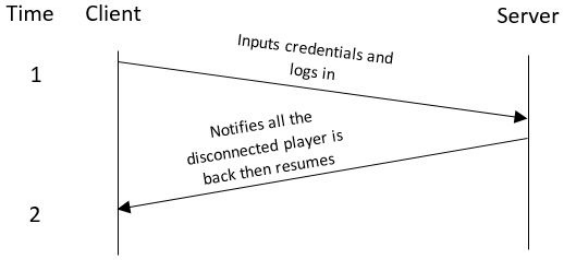
List online users:



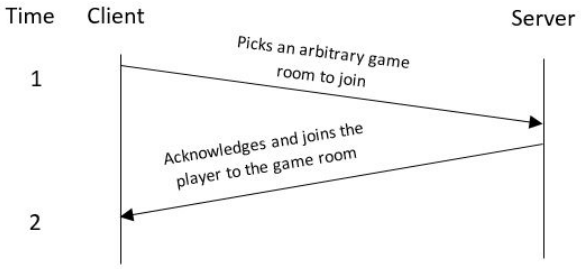
Disconnect:



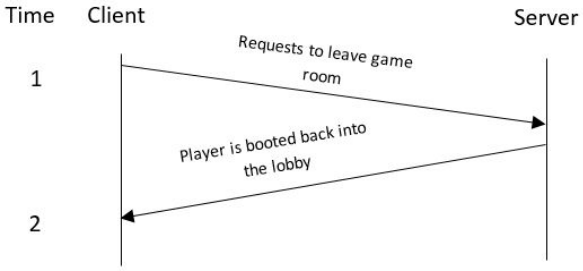
Reconnect:



Join game room:



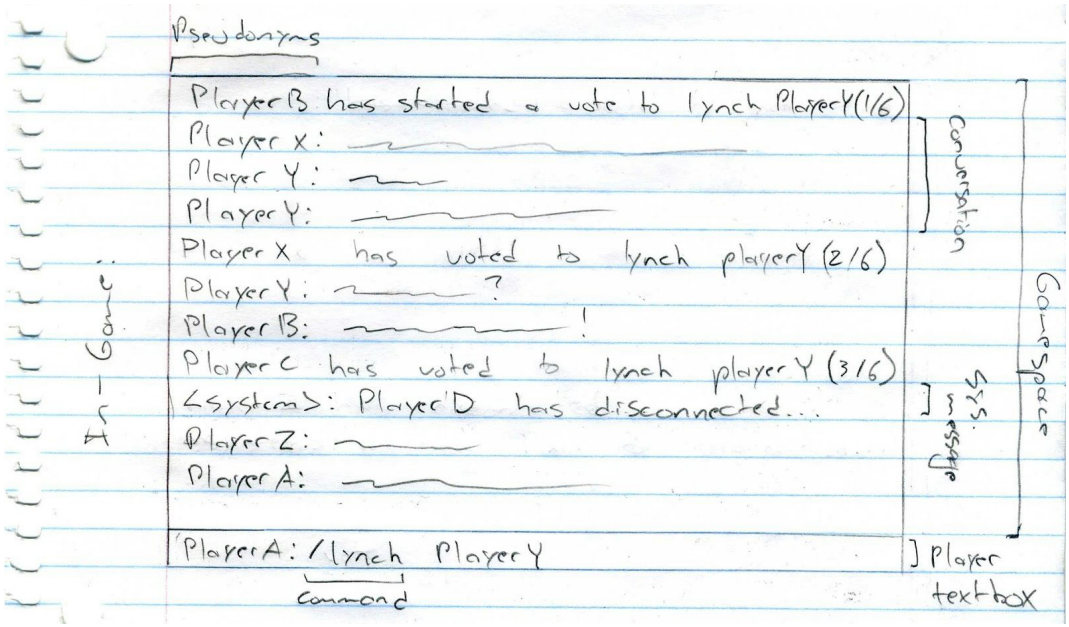
Leave game room:



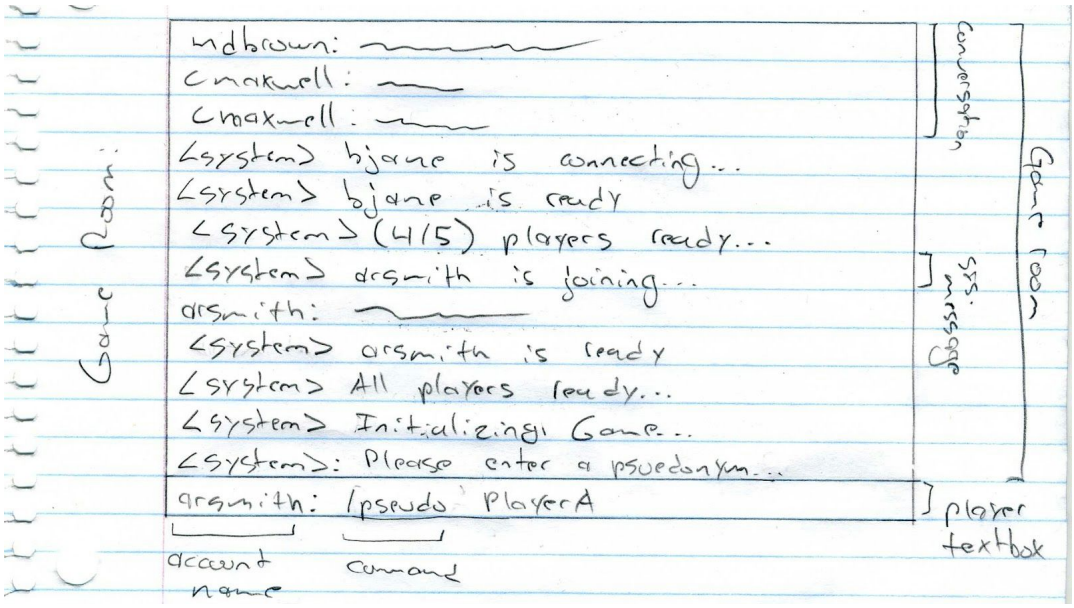
Message Handling: Our message formats have been generalized to include an integer code identifier for both the client and server messages. This identifier will allow us to quickly route the functionality for both the client and server. In the general case, when a message is received by either the client or the server, it will be sorted by its identifier into a handler function for the message type functionality. Inside of this function is where specific message checking will occur.

For instance, if the player accidentally sends a blank username or password when logging in, the function would first check to ensure that both elements are non-empty before attempting to validate their credentials. The server code itself will also include functionality to parse the client side input and generate a correct response, such as updating the game space, confirming login information or processing a vote in progress.

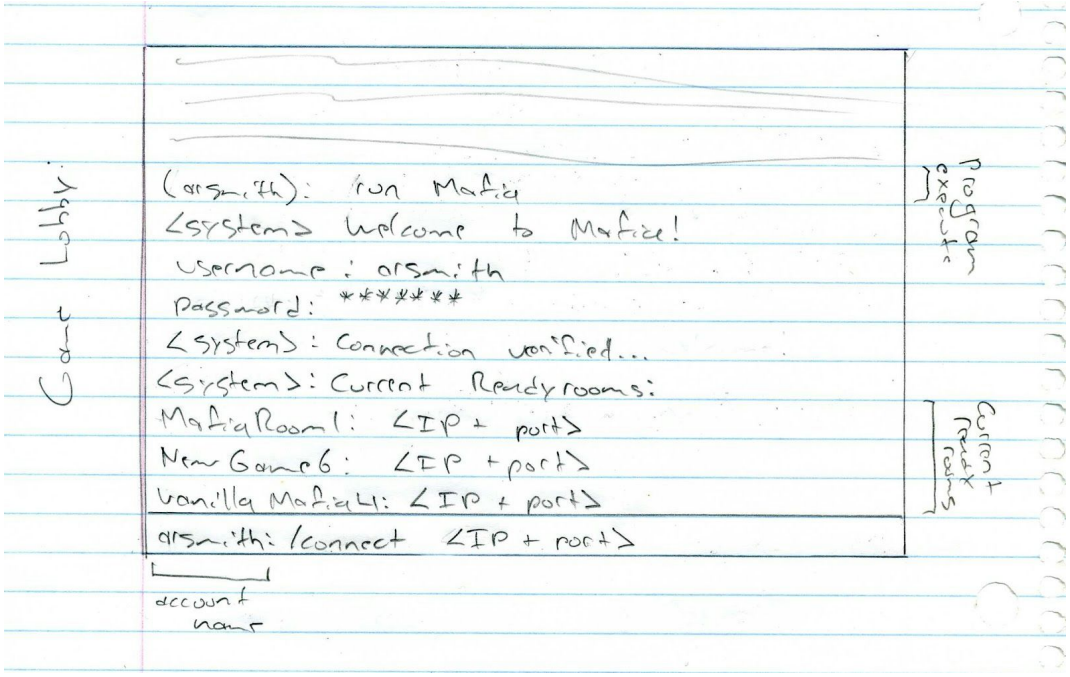
User Interface: Mafia is primarily a game about detecting people's motives based on their words and actions. Because the action set of the game is limited to lynching and murdering players only, it is the text of the game's conversation which is of critical importance. Due to the main focus of the game being text base, not much is gained from providing a complex UI, and in fact a simple UI which emphasizes the text is more beneficial. To therefore facilitate simplicity of design, the UI for the game and server interactions will be text based. By using such a UI, the player can keep a close eye on the conversation without being distracted by other UI elements. Likewise, as there is such a small set of possible actions for the players to take, there is really no need for specific UI elements to facilitate the lynching and murdering process. These actions along with any meta instructions related to server commands will be handled via /<command> in the text bar of the game, with the text bar's default functionality set to send messages only as can be seen here:



The game ready room will be highly similar to the in-game interface, as it will be converted directly into the game space once the requisite number of players has been met and all connected players are ready. It will continue to use the same /<command> structure as in the game to handle ready room functions such as indicating the player is ready and setting a pseudonym:

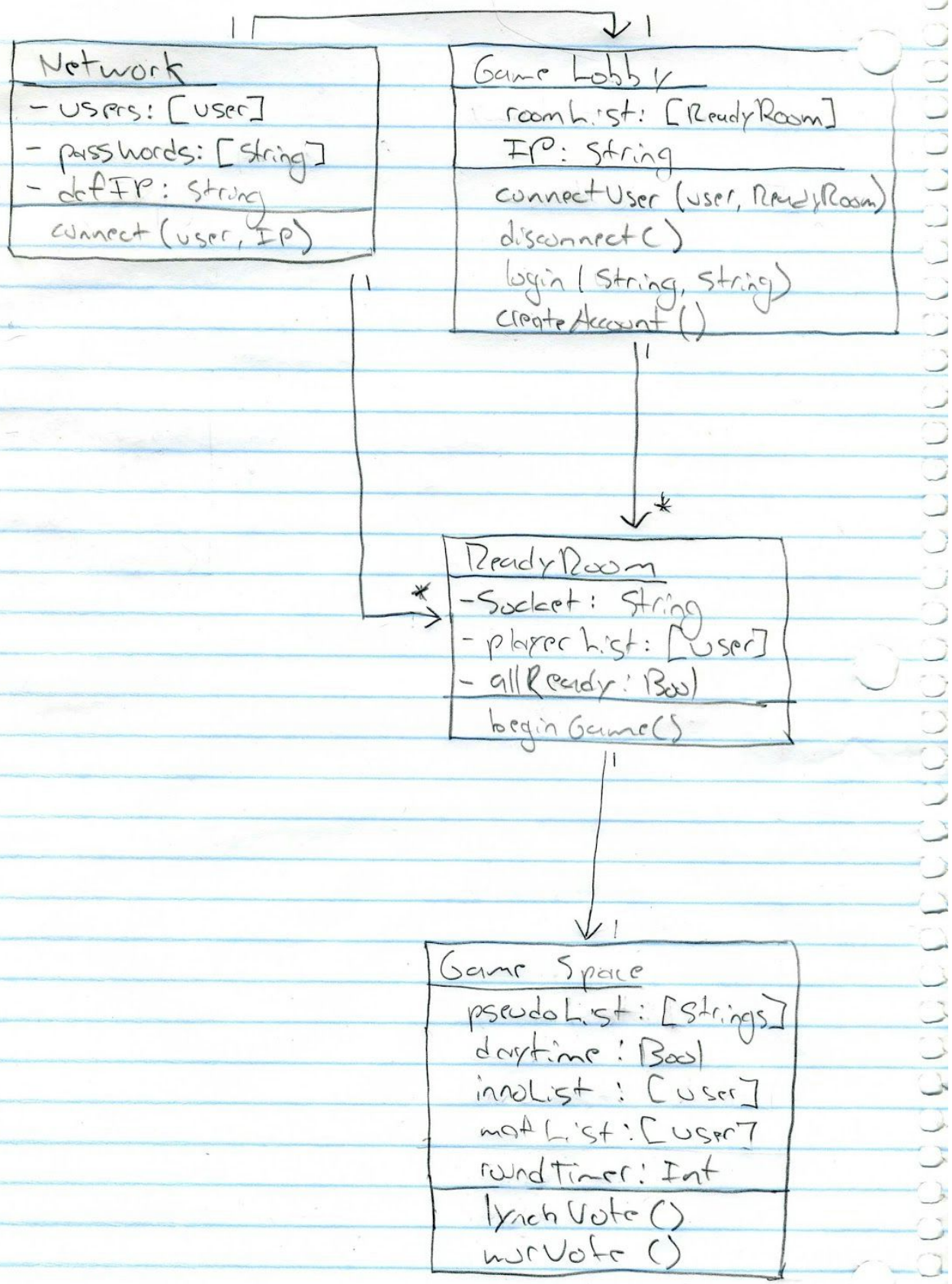


Finally, the game lobby will be the screen the user first connects to. It is the most simple as it only deals with player authentication and current game room creation and display:



Though the usage of the /<command> and text display is somewhat clunky, it should come easily to anyone who has used a text based chat program.

Object Oriented Design: As described in the interface section above, the game has three main states which interact in a layered manner. All three of these layers are dependant on the base network protocols which each will need to access in some way. The network protocols provide the basic game/server access and allow the player to connect to the general game lobby and connect to a game room. The network protocols therefore provide a base and necessary functionality for the other layers. Moving on to the layers themselves, the lowest is the game lobby. When a player connects to Mafia, this will be the level on which they enter. The lobby will have some number of game room objects associated with it, both those which are in play and those which are waiting to begin. This allows the player to connect to a game room of their choosing, and so move up to the middle layer. The middle layer of the program, the ready room, has associated with it a game instance object. This game instance is the desired game state of Mafia, but the maintenance of the game state falls to the ready room. It monitors the number of players and their ready state and only transfers to the game state when all connected players are ready to begin the game. When this condition is reached, the ready room passes the players into the game space, the highest layer. The game space holds all of the Mafia game related functionality, and it is inside this layer that the game is actually played. Players can escape back down the layers by disconnecting from the game, which will place them back at the game lobby level, or by logging out at the lobby level, which will close the Mafia client and dump the player back in their local system.


Network
- users: [user]
- passwords: [string]
- defIP: string
connect(user, IP)
Game Lobby
roomList: [ReadyRoom]
IP: string
connectUser(user, ReadyRoom)
disconnect()
login(string, string)
createAccount()
ReadyRoom
-Socket: String
- playerList: [user]
- allReady: Bool
beginGame()
Game Space
pseudoList: [strings]
daytime: Bool
innoList: [user]
mafList: [user]
roundTimer: Int
lynchVote()
murVote()
1
*

Expectations: Our expectations for this assignment are quite high. Because the game Mafia in a digital format can be seen as an extension of a text based chat client, we can assure that the game runs smoothly, provided we ensure that our message transfer protocols run smoothly. To achieve this, we will have to put special emphasis on ensuring our formats and exchange work correctly and efficiently. Provided this is achieved, the added game specific functionality should not slow down the game exchange. Obviously, we will need to generate the game functionality and server interactions in a manner which are easy and simple to follow. integration of help functions for new users will also add to the game experience by getting new users up to speed quickly.

Plan: The execution of the project will depend on the integrity of the message formats and transfer protocol, which is where we will begin. Initial testing and verification of the message functionality in a simple messenger style will be the first goal. Testing and updating our message formats will allow us to have a solid base on which to build the rest of the game, and the messenger infrastructure we build can be integrated into the end game space. Finally we can use what has been created thus far to complete that actual gamespace. At this point we will begin the implementation of game specific functions, such as team assignment, the transfer from night to day, voting to lynch, multiple conversation channels and player observation. We will look to have these first tasks finished by Friday, March 18. Once we are sure that the message protocols can capture all needed message transfer, and can interface with some basic game functions, we will begin on the server communication and connection. This will allow us to develop the basic game lobby, account functions, invitations and server connect and disconnect and any data storage related code. From here we can move onto the creation of the ready room construct. Because the ready room acts as a staging area for a game of Mafia, much of the functionality will transfer over both from the completed game lobby and into the gamespace. Completing the ready room will allow us to integrate in the message functionality, along with some of the basic game commands, such as setting the player status to ready and creating pseudonyms. We will aim to have this tasks completed by Friday March 25. Once these final tasks are completed, the system should be in a state to begin final testing and then beginning adding advanced functionality if time permits. These final checks and further implementation will ocur up to the due date of April 1, and also account for any previous due date runover. For this deliverable we will begin by fleshing out some of our class and function definitions defined in our UML diagram in preparation for adding actual functionality.