CS 436 Project: Network programming (Simple RPG game) 100 points

This assignment is to be done in two-person groups. Do not show your code to other groups and do not look at other students' code. Prevent other students from accessing your code (do not put it in a public directory). However, you are free to use any tools you like (library, search engines), to discuss problems with others, and to seek the help of the instructor.

1. Python3

You will implement this project in Python3. How to learn Python3? There are many great tutorials on the Internet. The following is a link to a very easy and helpful tutorial for beginners with hands-on examples in the popular PyCharm IDE. Study the first 2 hours of the video.
https://www.youtube.com/watch?v=_uQrJ0TkZlc&t=16696s

The playlist tutorial by Corey Schafer is also great to learn Python with examples.
https://www.youtube.com/watch?v=YYXdXT2l-Gg&list=PL-osiE80TeTt2d9bfVyTiXJA-UTHn6WwU

You can use any IDE to develop your Python program. Some popular IDEs are Atom, VSC, and replit online IDE.

2. Socket programming in Python

Refer to the socket programming lecture and video. Also, a simple socket programming example is given in the project section of the course. Download the attached folder. I have also provided an instruction to run the files in the project section. We implement this project with **UDP**.

Note: At the client program, change the server to "localhost". Then you can run the server and multiple clients at different terminals all at the same machine.

3. Project Objective:

In this project, you will develop a client-server application that implements a RPG game using socket programming in Python language.

4. Implementation

In this project, we define 2 programs: (1) client.py, (2) server.py

The server program stores two lists: (1) a list of users, (2) a list of confirmed fight requests.

We assume we have 4 users with usernames: (1) A, (2) B, (3) C, (4) D. For simplicity, we assume each user's password is the same as its username. The server program stores a list of these 4 users. For each user, the server stores username, password, and other info.

In our RPG game, there is a server and multiple clients. The clients are the gamers. The gamers want to defeat other gamers.

The structure of a gamer includes some fields as following. A gamer's state is the collection of these values for that gamer.

1. username
2. password
3. lives: the number of gamer's lives. It is initialized to 2.
4. avatar: an image file that the gamer uploads to the server.
5. sword: a weapon. Could be at strength [0-3]. Used to defeat other gamers.
6. shield: a protector against the other gamers' swords. Could be at strength [0-3].
7. slaying-potion: Could be at strength [0-3]. Used to defeat other gamers.
8. healing-potion: a protector against other gamers' slaying potions. Could be at strength [0-3].

The client program stores all info about the individual gamer who runs that client. The server stores all info about all gamers. The server identifies the gamers by their usernames. We assume the server has already defined the gamers and assigned each a username and a password.

The values of sword, shield, and potions are initialized to -1, meaning "unassigned". The first time the user logs in to the game, they select values for these items according to the game rules as defined below. A gamer could be either active or inactive. Initially, all gamers are inactive. The server refers to the gamer's sword strength and lives to figure out whether the gamer is active or not. If the gamer's sword strength is -1, the gamer is still inactive. Once the user logged in to the game and assigned the values, the sword strength value will change to a number in range [0-3]. Then, the gamer is considered active. After a few rounds of the game, the gamer loses all its lives. If a gamer's lives is 0, the gamer is considered inactive.

The first time that a user logs into the game, it is assigned a total of 10 strengths for its 4 properties: sword, shield and potions. The gamer can select how to distribute the 10 strengths among its sword, shield, and potions. For example, a gamer may get a sword at strength 3, a shield at strength 3, a slaying-potion at strength 2, and a healing-potion at strength 2. Strength 3 is the strongest and strength 0 is the weakest. The client program displays the 4 properties in order, and the gamer selects a number in [0-3] to assign to each one. If the total of the assigned strengths is not 10, the client program will reject that assignment and asks the gamer to try again. For example, a gamer may get a sword at strength 3, a shield at strength 3, a slaying-potion at strength 2, and a healing potion at strength 1. This assignment will be rejected as the total strengths 3+3+2+1=9, which is less than 10.

Assume gamer A wants to fight with gamer B. The rules of this game are as follows.

1. If gamer A requests to fight with gamer B using sword with strength 2, the server compares that with B's shield strength.
    a. If B's shield strength is 2 (the same as A's sword strength request), both A and B lose 1 life. A loses 2 sword strengths. B loses 2 shield strengths.
    b. If B's shield strength is 1 (any number less than 2, the A's sword strength request), A gains 1 life and B loses 1 life. A loses 2-1 = 1 sword strengths because B's shield strength was 1. B loses 1 shield strengths because B's shield strength was 1.
    c. If B's shield strength is 3 (any number greater than 2, the A's requested sword strength), A loses 1 life and B gains 1 life. A loses 2 sword strengths because A's requested sword strength was 2. B loses 3-2=1 shield strengths because B's shield strength was 3 and A's requested sword strength was 2.
    d.
2. The same game rules as used by B's shield against A's sword are used by B's healing-potion against A's slaying-potion.

The server stores all the gamers' info in a table and keeps track of their info.

The structure of a fight request includes some fields as following:

1. requester: the username of the fight requester
2. boss: the username of the adversary who the fight requester wants to fight with
3. fighting-item: either (1) sword or (2) slaying-potion
4. fighting-item-strength: a number in range [0-3]
5. winner: the final winner of the fight, who will be either (1) requester, (2) boss, (3) none

The client program

When a gamer runs the game program, they follow the steps given below.

1. The client program asks the gamer to enter its username and password to log in.
2. The client program will send the username and password to the server. If it matches a legitimate user on the server, the server will accept it. Otherwise, it will reject it.
3. If accepted, the gamer will be directed to the next steps. If rejected, the client program asks the gamer to try again and enter a new username and password.
4. We assume the login has been successful. Then, the server will check the gamer's lives. If it is 0, it prompts a message that the game is over for this gamer.
5. If the gamer's lives is above 0, the server will send the gamer's current state to the client and the gamer's info will show up on the screen. The initial values of the gamer's sword, shield, and potions are -1, meaning "unassigned". If the server sends the value -1 for these items, the client should display "unassigned" for their values.
6. Then, the gamer can upload a jpg image file as its avatar. The file will be stored at a folder on the server with name username.jpg. If the gamer has already uploaded an avatar image in the past, a new upload will replace the old one.
7. If this is the first time the gamer plays the game, its sword value is still -1, and the gamer is still inactive. If that is the case, the client program prompts a message: "You

are given 10 strengths". The gamer should select how to assign the 10 strengths to its sword, shield, and potions according to the game rules are described above.

8. After a successful assignment of strengths, the gamer can start playing. The client will send the gamer's current state to server, and the sever will store them in its table.
9. The gamer's lives is initialized to 2 by default. Each time the gamer defeats another gamer, it will gain a life according to the game rules defined above. Each time a gamer is defeated by another gamer, it will lose one life according to the game rules defined above. When the gamer reaches 0 life, the game is over for this gamer, and the gamer cannot play the game anymore. The gamer turns to inactive mode.
10. At the next step, the gamer can request the server to send the list of other active gamers. If the user does not want to get a list of other active gamers, the client program will quit. An active gamer is a gamer with at least one life who has assigned strengths to its sword, shield and potions. If there is no other active gamer, the client program will quit. Otherwise, the server will send the list of the other active gamers' "usernames". The server will not send any other info about active gamers. The server will not send any info about inactive gamers.
11. Then, the gamer can request to download another active gamer's avatar file. Then the server will send the file, and the file will be stored on the client's machine.
12. Then, the gamer can ask the server to send the list of confirmed fight requests. When the server responds, a complete list of confirmed requests will show up on the screen. It must look like a structured table or list.
    Note: If the printed list of requests on the screen looks like a sequential statement rather than a structured table or list, **you will miss 10 points out of 100**.
13. The gamer can send a fight request to the server. In a fight request, the gamer selects another gamer and enters the strengths it wants to use against that gamer. For example, gamer "A" wants to fight with gamer "B" with a sword with strength 3. If gamer "A" does not have enough sword strength, the client program will reject this fight request. Otherwise, the request will be sent to the server.
14. When the server receives a fight request, it first checks whether the gamer has enough strength according to the gamer's info on the server. If not, it will reject the fight request. Otherwise, it confirms the request and stores that in the server's list of confirmed fight requests. Then, the server compares the requested strength of gamer A with the according strength of gamer B. The server uses the game rules defined above. The server will update the lives and strengths of A and B. Then, it responds A with A's current state, including its lives and properties' strengths.
15. Then, the gamer can send a new fight request.
16. The user can request to get a list of active gamers and their current state from the server. Then the server will send all info on all active gamers, and the complete info will show up on the screen for the user. After that, the client program will quit the game with no further question.
    Note: If the printed list of gamers and their info on the screen looks like a sequential statement rather than a structured table or list, **you will miss 10 points out of 100**.
17. If the client program quits, the gamer's state will remain in the server's table and can be accessed the next time the gamer runs the client program and logs in.

Note: Whenever the server receives or sends a message, it should prompt a message on the screen and report what it is doing. For example, “Received a request from user A”, “User A is authenticated”, “Sent the list of requests to user A”, “Confirmed a request for user A”. **You will miss 10 points out of 100** if your server does not prompt the messages.

5. Notes to test the program

Run the following test steps and take a **snapshot** of your terminals at each step. Make a Report.docx file and enter the snapshot of each step to the report file. Zoom in and take a snapshot that the text in it is large enough and visible to read.

Note that you can run all the programs on the same computer. Use ‘localhost’ for the server’s name in the client program.

6. Test your program as following:

First run server.py in one terminal and then run client.py program in another terminal.
Note that in all steps the server should prompt messages on what it has done in that step.

Step 1. Enter username A and password G. The server will reject that. The client displays a menu of two items: (1) try again, (2) quit. The user selects try again.

Step 2. Enter username A and password A. The server will accept that and checks gamer A’s lives. It is not zero. Thus, the user still can play. The server will send the gamer’s state, including sword, shield, and potions strengths and lives, to the client, and they will show up on the screen for the user. Any -1 value will show up as “unassigned”.

Step 3. The client asks whether the user wants to upload an avatar file. Then, the user enters y as yes. Then the user will enter the name of the file that already exists in the current folder. Then, the file will be uploaded to the server and stored on the server’s folder with name A.jpg because the current gamer’s username is A. Take a snapshot of the server’s folder with file.

Step 4. As the gamer’s sword value is still -1, this is the first time the gamer plays the game. The client program prompts a message: “You are given 10 strengths”. The gamer can select how to assign them to its sword, shield, and potions. The program displays the 4 properties in order, and the gamer selects a number in [0-3] to assign to that.

a. Assign 4,3,2,1 to the 4 items. It will be rejected as one of the values is above 3.
b. Assign 3,3,2,1 to the 4 items. It will be rejected as the total is less than 10.
c. Assign 3,3,3,2 to the 4 items. It will be rejected as the total is greater than 10.
d. Assign 3,3,3,1 to the 4 items. It will be accepted. The client program will send the assigned values to the server to be stored in the server’ table.

Step 5. The client asks whether the user wants to see a list of other active gamers. The client answers y as yes. The server responds that no other active user exists yet. The client program quits.

Step 6. Run another client program for user B. User B assigns 2,2,3,3 to its sword, shield, slaying-potion and healing potion. User B uploads its avatar file. Take a snapshot of the server's folder with the A and B's files.

The client asks whether the user wants to get the list of active users and their current states. The user enters y as yes. The request will be sent to the server. The server returns a list of one item, which is A's username. The client program will display that.

Then, the client program asks whether the user wants to download an active gamer's avatar file. The user enters y as yes. The client program asks the user to enter the username (or number) of the other active gamer. User B enters A's username (or number). The client sends a request to the server. The server responds with the file. The A's avatar will be downloaded to user B's folder. Take a snapshot of user B's folder, including user A's avatar file.

Step 7. Run another client program for user C. User D remains inactive for the rest of our test. B and C will log in.

User C assigns 3,1,3,3 to its sword, shield, slaying-potion and healing potion. Now the 3 active users have the following strengths and lives.

Active user	sword	shield	slaying-potion	healing-potion	lives
A	3	3	3	1	2
B	2	2	3	3	2
C	3	1	3	3	2

User C also uploads its avatar file. Take a snapshot of the server's folder with C's avatar file.

User C requests to download B's avatar. Take a snapshot of the C's folder including B's file.

User C requests to get a list of active users, and a list of A and B will show up on the screen.

Step 8. Gamer B sends a fight request with gamer A with 2 sword strength. B loses 1 life. A gains 1 life. B loses 2 sword strength. A loses 2 shield strength. The client asks whether the user wants to send a new fight request, and the user answers n as no. Then B requests to get a list of active users. Now the 3 active users have the following strengths and lives.

Active user	sword	shield	slaying-potion	healing-potion	lives
A	3	1	3	1	3
B	0	2	3	3	1
C	3	1	3	3	2

Step 9. Gamer C sends a fight request with gamer B with 3 slaying-potion strength. Both C and B lose 1 life. C loses 3 slaying-potion strength. B loses 3 healing-potion strength. The client asks whether the user wants to send a new fight request, and the user answers n as no. Then C requests to get a list of active users. Now A is inactive because it has 0 lives. Only B and C are active. The 2 active users have the following strengths and lives. The gray row will not show up.

Active user	Sword	shield	slaying-potion	healing-potion	lives
A	3	1	3	1	3
B	0	2	3	0	0
C	3	1	0	3	1

Step 10. Gamer A logs in and gets its current state, which will be 3,1,3,1,3 for sword, shield, slaying-potion, healing-potion, and lives respectively.

Then, A sends a fight request with gamer C with 3 sword strength. A gains 1 life. C loses 1 life. A loses 1 sword strength. C loses 1 shield strength. The client asks whether the user wants to send a new fight request, and the user answers n as no. Then, A requests to get a list of active users. Only A is active. The 1 active user have the following strengths and lives. The gray rows will not show up.

Active user	sword	shield	slaying-potion	healing-potion	lives
A	2	1	3	1	4
B	0	2	3	0	0
C	3	0	0	3	0

Step 11. Gamer B attempts to log in but gets a game over message.

7. Submission:

Note: You will miss 50 points if you do not follow the test instructions given above.

Submit a zip file containing the following files:

1. client.py
2. server.py
3. Report.docx

Note: It is OK to add more files. For instance, you can make a class or a json structure to work with the messages and store it in a separate file and then include it in the other files.

In the Report file enter the names of team members and the screenshots of each step of test.

Put all the files in a folder named YourTeamMembers and compress it into a .zip file. Try to make a nice user interface. You may get some extra credit if you design a creative user interface for your application. Tkinter is a useful package in Python for user interface.