Concordia University

Engineering and Computer Science

COMP 6231
Distributed Systems Design

Project
Software Failure Tolerant Distributed Player Status System (DPSS)

Submitted To:
Prof. Mohamed Taleb

Submitted By:
Mo Rokibul Islam (40060110)

Date of Submission:
August 12, 2020

1. Description

Distributed Player Status System (DPSS) is aims to connect a group of servers located in three different geolocations. This distributed system will be used to manage player status, create account, transfer account between servers, suspend account across multiple game servers. The users of the system are players and administrator, and both are identified by a unique *Username*.

The administrator could perform:

1. Update Player Status
2. Suspend Existing Player Account

Player only limited to following kinds of work:

1. Create New Account
2. Transfer Account from one server to another.

There are three servers are used and they are in three different geolocations.

1. North America (NA): IP-addresses starting with 132 such as 132.xxx.xxx.xxx
2. Europe (EU): IP-addresses starting with 93 such as 93.xxx.xxx.xxx
3. Asia (AS): IP-addresses starting with 182 such as 182.xxx.xxx.xxx

There will be three separate groups of the same servers and each group replica contains a Replica managers (RM) and corresponding Replica servers. Apart from that there will be a Front End (FE) server and the implementation of CORBA Front End has been considered as a Leader replica. The entire server system would be Replicas, FE, and RM. In this project, **Active Replication** will be used to handle a single software (non-malicious Byzantine) failure.

2. Protocols Used

1. **IDL Compiler:** To register remote objects using.
2. **ORB:** Using CORBA naming service register the objects remotely. Then Access the objects by clients. Clients and servers are running on different threads
3. **Java Programming Language**
4. **UDP/IP Socket:** Inter Server Communication is done using UDP Protocol. As we try to access any distributed information. The servers need to communicate with each other.

3. Active Replication

There are five phases to perform a client request (copied from class lecture slide)

1. **Request**: Front End attaches a unique id and uses totally ordered reliable multicast to send request to RMs.
2. **Coordination**: The multicast delivers requests to all the RMs in the same (total) order.
3. **Execution**: Every RM executes the request. They are state machines and receive requests in the same order, so the effects are identical. The id is put in the response.
4. **Agreement**: No agreement is required because all RMs execute the same operations in the same order, due to the properties of the totally ordered multicast.
5. **Response**: FEs collect responses from RMs. FE may just use one or more responses.

3. Architecture

3.1 Model Video

Clients only can communication channel with Front-End (FE) server. We have implemented FE that accepts the Clients request through CORBA. Once FE receive a request, it sends the request to Sequencer then Sequencer send the messages with a sequence id to all Replica Managers (RM). This is multicasting of each request from Sequencer to all RMs and which conform that all RMs perform the same operations in the same order. Each RMs process each request according to sequence id and send it to the corresponding servers (replica server). Specific servers execute and send output to the FE. FE receives three outputs from all replicas. And finally, send a single corrected result to the client. The total process is completed by without client's concern.

The communication Sequencer to RMs are done by multicast UDP/IP connection. Communication between individual RM and corresponding servers have been established by the UDP connection. Each Replica server and FE communicate with each other by reliable UDP/IP connection. Within group, if the operation requires data from other geolocation servers, then the server makes a UDP request to the target server and gets the required data using a socket communication.

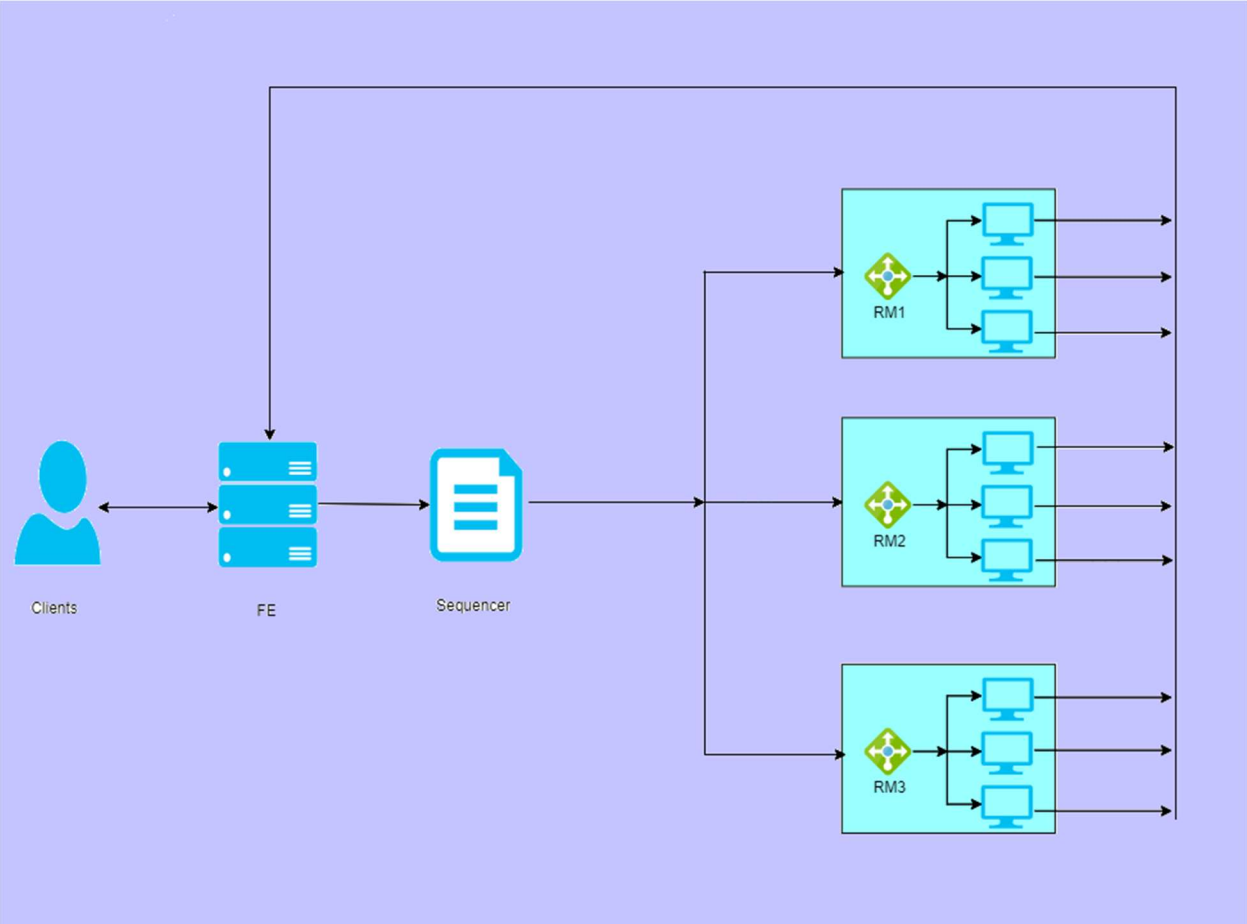


Fig 1: Model of Active Replication of a DPSS.

3.2 Decision Process

Once FE receives replies of the client's requests then FE starts to process. As we have 3 replicas, so FE will receive 3 replies for each client's request. The question is how we make sure validity of the output that FE receive from 3 replica servers. Before going to discus, first let have a look at the pattern of input and output message.

Input pattern of the sending request: ***server name + operation name+ input parameter***
Example:

```
String input_message = serverName+":"+"logIn"+":"+userName+":"+password+":"+IP+":";
```

Here, operation name is "logIn" and its input parameters are username, password, and IP address. After that Sequencer add a sequence id at the end of the input message. Actual print view is: *AS:logIn:soma2000:soma2000:182.123.123.123:100001*

Output pattern of the send result to FE: ***input message + output message + Replica server no***
Example:

```
String sendingResult= input_message+";"+Boolean.toString(output_message)+";"+RMno;
```

Actual print view is: *AS:logIn:soma2000:soma2000:182.123.123.123:100001;true;100*

RMno helps to track fault message of RM and its corresponding replica server. FE received many output messages from by different user. However, as the username is unique, so we can get specific output according to username. To analyze the received output, we simply remove last part of the message as they are different from each other. Now we are going to discuss, three possible outcomes of the output:

1. **No Fault**: If FE receives all identical results from replica servers, then FE pass the result to Client.
2. **Single Fault**: If FE found any of the replica send incorrect result i.e. two results are identical, the FE will try to get correct result by resending same message to the corresponding replica. However, if a replica fails to provide correct result for 3 consecutive times then FE inform to corresponding RM to take necessary action to resolve the issue such as by fixing bug or restart the server.
3. **All Faults**: If FE found all the results are different, then FE will resend same request to all RMs to get correct result.

Other issues may arise to receive less than 3 replies due to higher response time from any replica servers and system may crush. Although in this project, we are not going to handle anything related to system crush, but to avoid this, if FE receive less than 3 replies it will be notify that something is going to wrong, please re submit your job again.

4. Functional Requirements

Functional requirements capture the intended behaviour of the system. This section contains the *Actor Goal List* and the *Use Case view*.

4.1 Actor Goal List

Actor	Goal
Player Client	Can use the system to create account and transfer their account from one geolocation server to another.
Administrator Client	Can use the system to enter, get update player status or suspend player account.
Front End	Respond to the plyer and administrator clients depending on their request, passing request to both RM and Sequencer, convey correct reply to clients.
Sequencer	Add a sequence id each client request and pass it to all the RMs,
Replica Manager	Respond to FE and Sequencer depending on their request and send feedback to FE.

4.2 Use Case view

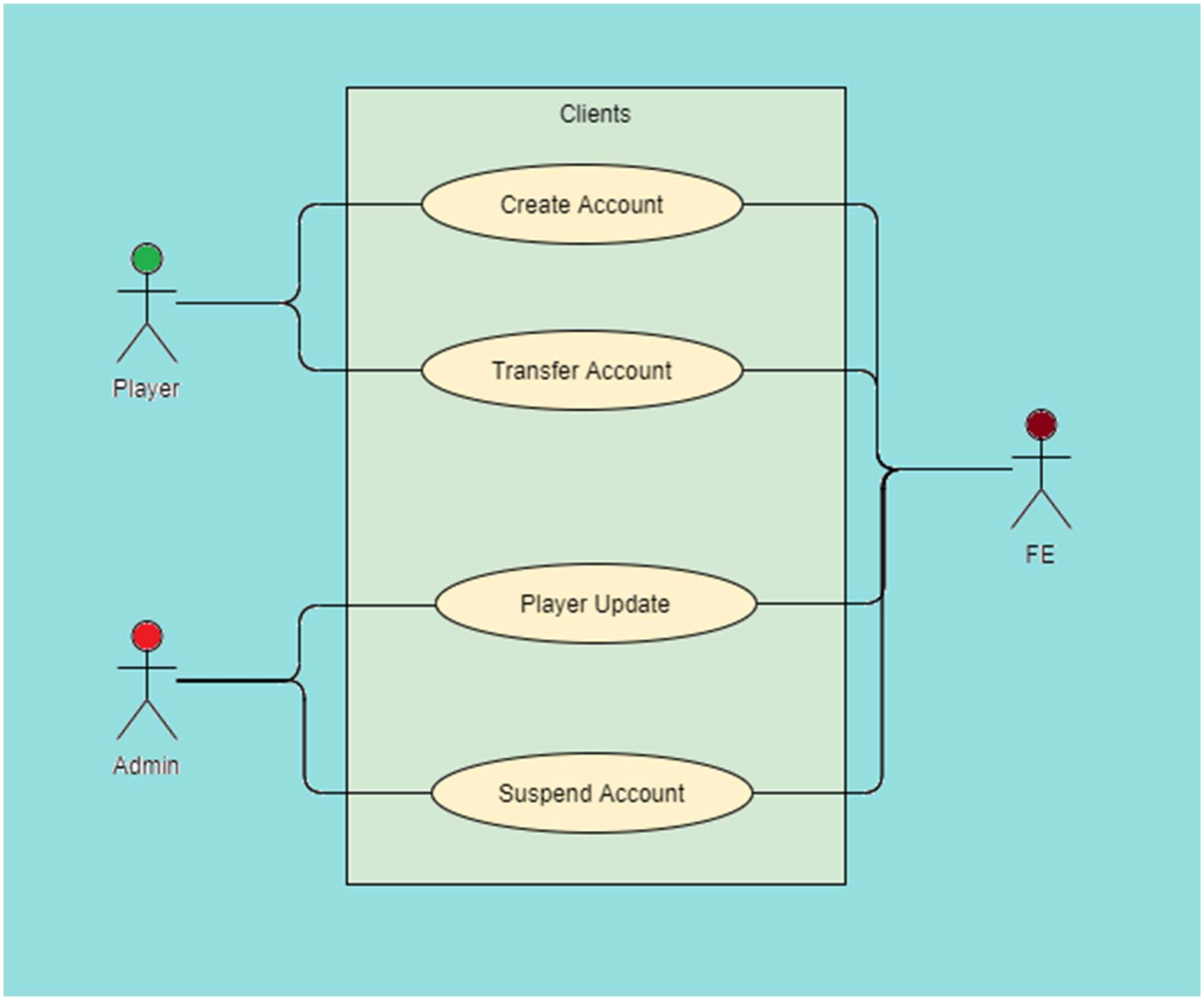


Figure 2. Use case model.

The player can create a new account if he is new user. If he is an existing user, then he can transfer his account from one geolocation server to another once he has successfully logged in to the system. On the other side, the administrator can get updated login information of the players and suspend any player account.

4. Techniques

4.1 Reliable Multicast

Multicast communication requires coordination and agreement. The aim is for members of a group to receive copies of messages sent to the group. Many different delivery guarantees are possible. A process can multicast using a single operation instead of a send to each member.

On initialization
Received := {};

For process p to R-multicast message m to group g
B-multicast(g, m); // $p \in g$ is included as a destination

On B-deliver(m) *at process q with g = group*(m)
if ($m \notin$ *Received*)
then
Received := *Received* $\cup$ {m};
if ($q \neq p$) *then B-multicast*(g, m); *end if*
R-deliver m;
end if

4.2 FIFO Broadcasting:

FIFO broadcast is a strengthening of Reliable broadcast. It guarantees FIFO ordering, which stands for "First In, First Out". To do this, we make use of the sequence number of the message, i.e. totally ordered reliable multicast to send request to RMs and each RM deliver send it to all its corresponding replica server according to lower sequence number i.e. FIFO ordering.

4.3 Data Structure

Each server maintains its in-memory database. This database is implemented as

```
private HashMap<String, ArrayList<ArrayList<String>>> glDatabase.
```

As per the requirements we stored all the data in HashMap with String "key" which is the first letter of Username and values are List of List of String which are information of user. Keys are considered upper case.

We have used PriorityQueue data structure for FIFO in RMs:

```
public static PriorityQueue<Message> fifo = new PriorityQueue<Message>(new
                                MessageComparator());
```

which keep track each message received from FE and send it to corresponding Replica server with sequence order.

The replies of the client's requests are store in a list

```
public ArrayList<String> messagelist = new ArrayList<String>();
```

4.4 Concurrency

Since each user runs on a thread. So, the system needs to be able to handle concurrent requests and provide thread safety to its data. This is achieved by using ***"synchronized"*** blocks whenever there is a need to update the in-memory database.

4.5 Logging

Custom logger is used to log all the messages. Each user operation is logged into the client specific log file. The same is true for each RM and corresponding replica servers too.

5. How to run the Project

1. Start ORB.
2. Start FE server
3. Start Sequencer
4. Start three RMs
5. Launch Client.java to start.

All the commands that I used are:

→ Start ORB

```
start orbd -ORBInitialPort 900
```

→ Start FE

```
start java frontEnd.FrontEnd -ORBInitialPort 900 -ORBInitialHost localhost
```

→ Start Sequencer

```
java sequencer.Sequencer -ORBInitialPort 900 -ORBInitialHost localhost
```

→ Start RMs in individual command line

```
start java rmOne.RM1 -ORBInitialPort 900 -ORBInitialHost localhost
start java rmOne.RM2 -ORBInitialPort 900 -ORBInitialHost localhost
start java rmOne.RM3 -ORBInitialPort 900 -ORBInitialHost localhost
```

→ Run java can be another command line or Eclipse

```
java clients.Cliemt -ORBInitialPort 900 -ORBInitialHost localhost
```

6. Test Scenario and Test Case

We have initialized few static data in our database as per requirement.

First Name	Last Name	Age	Username	Password	IP Address
Sami	Islam	7	sami2014	sami2014	132.123.123.123
Jami	Islam	1	jami2020	jami2020	93.123.123.123
Rokib	Islam	35	rokib2000	rokib2000	93.123.123.123
Soma	Islam	33	soma2000	soma2000	182.123.123.123
Ariful	Islam	25	arif2000	arif2000	93.123.123.123
Robiul	Islam	60	robi1000	robi1000	132.123.123.123
Roma	Islam	20	roma1000	roma100	132.123.123.123
Rimjhim	Islam	2	rimjhim2020	rimjhim2020	182.123.123.123

Both username and password for admin are fixed which is “admin”.

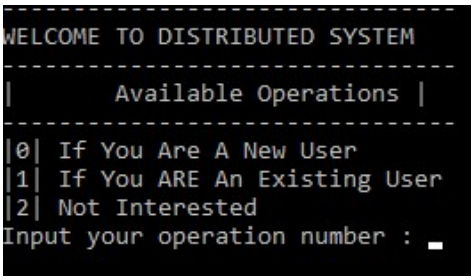
6.1 Test Scenario

No	Requirement Name	Test Scenario	Precondition	Test Case
1	Player Facility	LogIn	None	1. Validate with valid username 2. Validate with invalid username 3. Validate with invalid password 4. Validate with invalid password 5. Validate with invalid IP Address 6. Validate with invalid IP Address
2	Player Facility	Create Account	Correct IP Address	1. Validate with username length 2. Validate with password length 3. Validate with correct age format 4. Validate uniqueness of Username
3	Player Facility	Transfer Account	LogIn	1. Validate with invalid IP Address 2. Validate with invalid IP Address
4	Admin Facility	Player Update	Admin LogIn	1. It will show updated player LogIn and LogOut Status
5	Admin Facility	Suspend Player Account	Admin LogIn and Player Offline	1. Validate with valid username 2. Validate with invalid username
6	Player Facility	LogOut	LogIn	1. Player will LogOut and main window will display.

6.2 Test Case

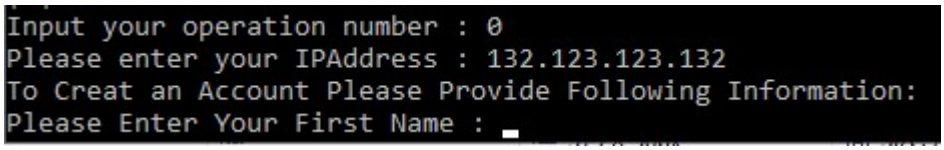
6.2.1. Start Game

This first window of assignment

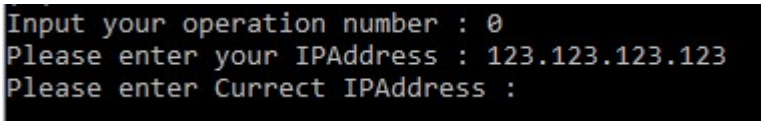


If you press **0**, which means you do not have any account, initially you will be asked to pass IP Address only, after verifying ip address, you will be directed to create an account portal, otherwise provide correct IP address will be suggested.

Test Scenario 01: Validate with valid IP Address



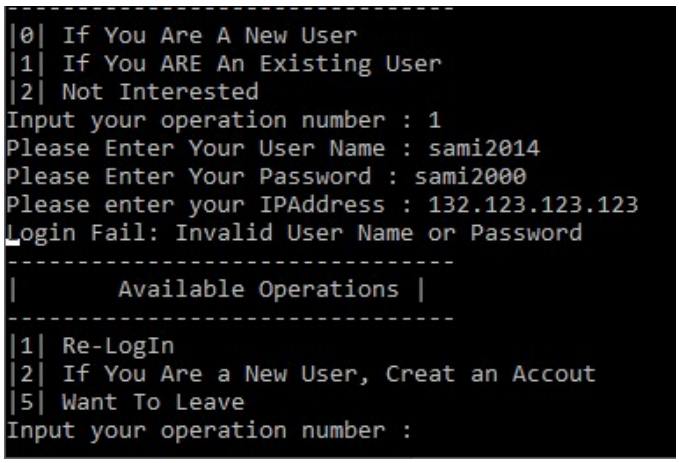
Test Scenario 02: Validate with valid IP Address



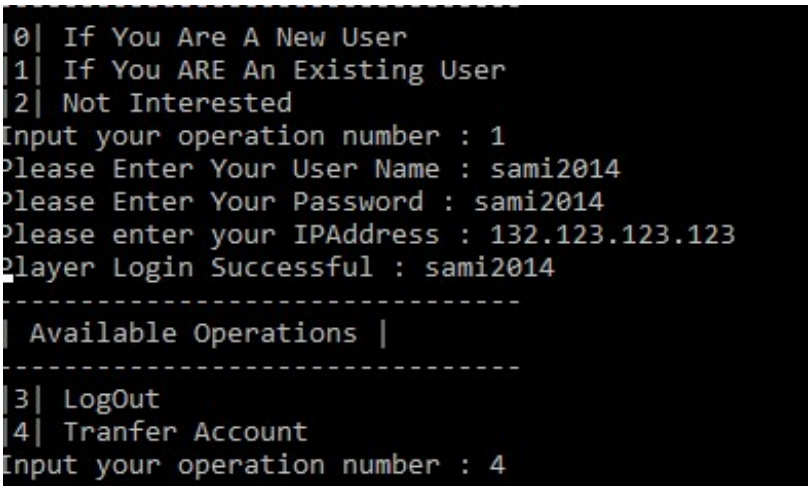
6.2.2. Log-In

If you press **1**, which means you are existing user, so you will be asked to pass your username, password, and IP Address, you will get message according to valid information. For admin, as we are not saving admin information in database, so, both **username** and **password** of the admin are "**admin**" and with any valid format of IP address will let you pass log-in.

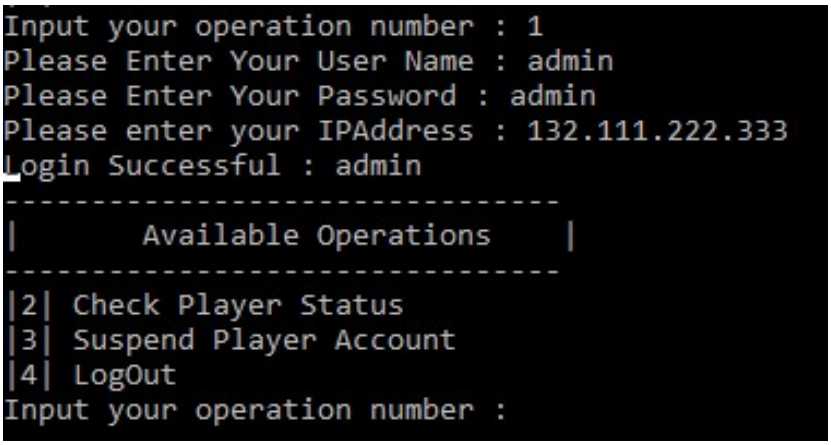
Test Scenario 03: Validate with invalid username and password



Test Scenario 04: Validate with valid username, password, and IP address.



Test Scenario 05: Check Log-In with valid Information of admin



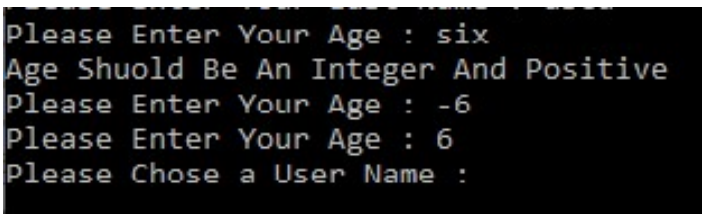
For the case of invalid information of admin, you will be asked to Log-in again.

6.2.3. Create New Account

As you already provided IP Address, so you will not be asked again to provide it.

Test Scenario 06: Validate Correct Age format

As **Age** must be positive integer, correct format will be asked if you provide wrong format. The input “six” and “-6” gives fail result.



Test Scenario 07: Validate with Username length
Username should be between 6 and 16 character. Here, when we proved username 5 and 17 characters, but we see both cases were failed.

```
Please Chose a User Name : rim12
Username is Too Short or Too Long
Please Select Usrename Between 6 and 16 Character: rimjhimislam30301
Username is Too Short or Too Long
Please Select Usrename Between 6 and 16 Character: rim3030
Please Chose Password:
```

Test Scenario 07: Validate with uniqueness of Username
Username is unique identity, so it will check if the same username is exits among all the server. So, it will check all the database. If same username is exits you will be asked to open your account again.

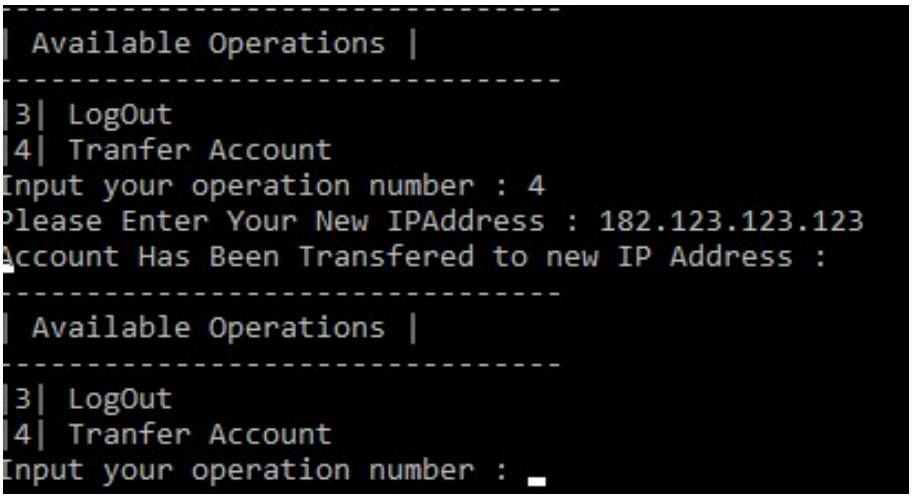
```
Please Chose a User Name : sami2014
Please Chose Password: sami2014
This User Name is Already Used, Please Try Another One:
Please Chose a User Name : samm2014
Login Player Account Successfully Created  as User Name: samm2014
```

Test Scenario 08: Check Password length
Same as check for Password, password should be at least 6 characters. When length of password is 4, which gives result fail.

```
Please Chose Password: abcd
Pasword is Too Short
Please Select Password Larger Than 6 Character: abcd1234
Login Player Account Successfully Created  as User Name: abcd1234
```

6.2.4. Transfer Account
It is player client operation. Precondition admin must be logged in.

Test Scenario 09:
It is a client operation and you need to provide a new valid IP address where you want to transfer your account. You will be asked to provide a valid IP address if you pass opposite.



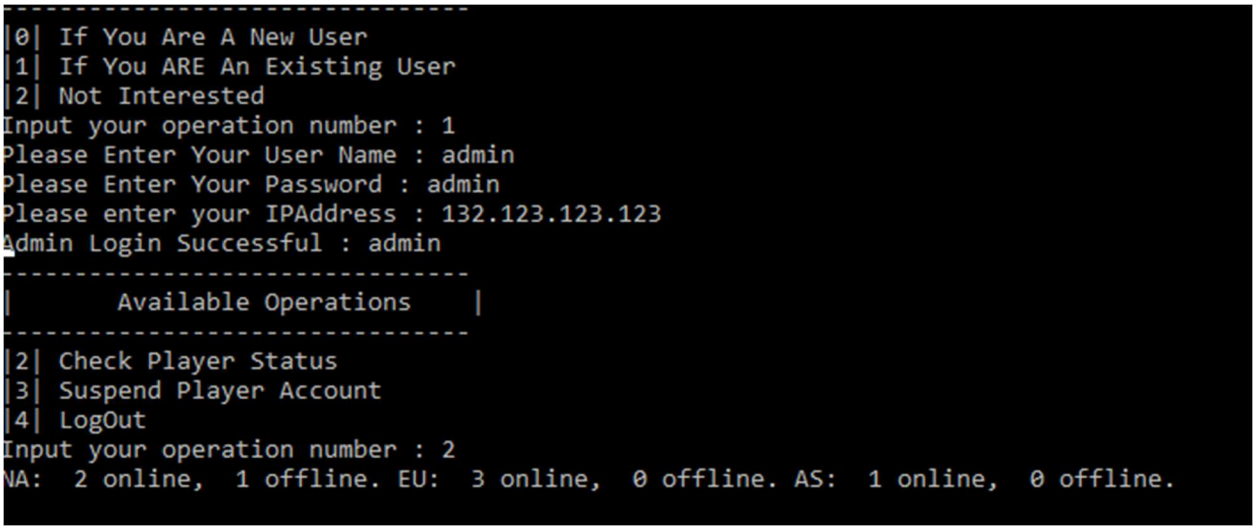
In this scenario, a player wants to transfer his account from NA to AS server. But client provided incorrect IP, so asked for correct format of IP, once provided the operation were done successfully. Even after successful moved account into new geolocation, if clients change his mind and would like to come back to his old location, simply the same procedure is needed to follow.

6.2.5. Check Player Status

It is an admin operation. Admin can get all the players status either a player is login or logout. Admin will get only total number of players' status from each server individually. Precondition admin must be logged in.

Test Scenario 10:

Once admin log-in successfully, he will get options to choose to get player updated status.



We see in NA server location 2 players are logged in and 1 player is logged out.

6.2.6. Remove Account

This operation is limited to amin only. There are preconditions, first admin needs to logged in and player should be off line.

Test Scenario 11: Suspend player account **Pass**

It is an admin operation and you need to provide a valid username that you want to remove from database.

```
--------------------------------
|       Available Operations    |
--------------------------------
|2| Check Player Status
|3| Suspend Player Account
|4| LogOut
Input your operation number : 3
Please Enter Account User Name You Want to Suspend : sami2014
Account Suspendded Succesfully
--------------------------------
|       Available Operations    |
--------------------------------
|2| Check Player Status
|3| Suspend Player Account
|4| LogOut
Input your operation number : 2
NA:  2 online,  0 offline. EU:  3 online,  0 offline. AS:  1 online,  0 offline.
```

We see that the logged off player from previous scenario 10 has been removed, so we 0 player in offline in NA server.

Test Scenario 12: Suspend player account Failed

You will be asked valid username address if username is not found. You will have an option retry to remove player account if you want.

```
--------------------------------
|       Available Operations    |
--------------------------------
|2| Check Player Status
|3| Suspend Player Account
|4| LogOut
Input your operation number : 3
Please Enter Account User Name You Want to Suspend : sami2014
This Account Has Already Suspendded OR Incorrect Account:
```

Now, admin tried to remove the account that he already removed; he is informed by the message.

6.2.7 Fault Tolerance

Test Scenario 13: Single Fault with one-time fault

Log-In for username “soma2000”, the RM1 will provide incorrect result, so FE resend the same request to RM1 and finally get all identical result.

Test Scenario 14: Single Fault with consecutive three-time fault

Log-In for username “robi000”, the RM1 will provide incorrect result for 3 successive time, now FE inform to RM1 and asked to fix the bug and receive accurate result.

Test Scenario 15: All Fault with one-time fault

All out outputs are Boolean except updated player status which is limited by admin. When “admin” provokes updated player status, the all the replica servers provide incorrect result. So, FE resend the same message with fault tag to all RMs and finally receive accurate result.

7. Challenges

The main challenge was decision making process discussed in details in the section 3.2. Once FE receives replies of the client’s requests from replica servers. Another challenge was delay response from replica (rare happened), so FE does not get all 3 response but to make decision 3 responses are required. Another challenge was inter-communication among the individual replica servers. If an operation requires data from other geolocation servers, then the server makes a UDP request to the target server and gets the required data using a socket communication. For example, when a player wants to create a new account, we check that the username is already exist or not. So, we need a conformation from all the severs. When a player wants to move his account to another geolocation then he need to communicate other server that he wanted to move. Same as for when admin want player status and suspend a player account he needs to communicate among the servers.