Ali Taylan Akyürek

64229

PA1-2 Report:

Overview:

This project aims to: a server connects to the core api and transmit data from api to authorized clients. Execution starts, server register the usernames. Then client specifies the phase, he should enter 0 for authentication phase, and 1 for querying phase. If the user choses the authentication phase, server asks username and password, if they are correct according to userInfo.txt, user will be accepted by server and he will receive a token (which is hash function of username + “29”)to use in querying phase. If the user choses querying phase, if he was authenticated before, he can search articles by appending 0, and search journals by appending 1 to head of input string (at runtime, with scanner) and server gives title, year and topics information to client. User should be authenticated first in order to access the service.

Authentication:

The protocol is as follows:

\*Server did save the users first, without their passwords. With this method:

**public** **void** getUsers() **throws** IOException {

String user;

File users = **new** File(

"/Users/alitaylanakyurek/eclipse-workspace/PA1-2 yeni bastan/src/coreNetServer/userinfo.txt");

BufferedReader userInfo = **new** BufferedReader(**new** FileReader(users));

user = userInfo.readLine();

**int** i=0;

**while** (user != **null**) {

i++;

**if**(i%2 == 1) {

**if**(!userInfoList.contains(user)) {

userInfoList.add(user);

}

}

user = userInfo.readLine();

}

}

\*Then first, server ask from user his username, if that username is not registered before, server sends following TCP payload:

payload = **new** Payload((**byte**)0,(**byte**)2,"User does not exist");

oos.writeObject(payload);

\*As specified in pdf, its type field is (byte)2 which is a auth\_Fail message, if server send this client cant move to next step because of these lines on client side:

**while** (payload.type !=(**byte**)3) {

System.***out***.println("Enter your id");

\*

\*

\*

payload = (Payload)connectionToServer.ois.readObject();

}

\*User should keep trying until he enters a valid username (moreover, in this phase user cannot type quit and break the loop, he only can enter “no request status received” for terminating connection). If user enters a valid username, server will send following TCP payload to server:

payload = **new** Payload((**byte**)0,(**byte**)3,"Correct ID, please enter your password");

oos.writeObject(payload);

\*As specified in pdf, its type field is (byte)3 which is a auth\_ Success message. If client takes this, he can move to password verification phase. In this phase, server record correct user’s password with:

String password = userList.get(userList.indexOf(line) + 1);

(line was the user entry that makes client to receive auth\_ Success message before, ie. correct username)

\*After that, server asks for client input. If client’s input is different than the correct password, server sends following payload:

payload = **new** Payload((**byte**)0,(**byte**)1,"Wrong password " + (3 - i) + " tries left");

oos.writeObject(payload);

\*Which is a auth\_Challange message. And if client enters his password wrong 3 times, system will send following payload:

payload = **new** Payload((**byte**)0,(**byte**)2,"No more tries left");

oos.writeObject(payload);

\*Which will cause client to move from start again.

\*In order to client to move next phase, he should type password correctly and receive following auth\_Success message which is:

payload = **new** Payload((**byte**)0,(**byte**)3,"Correct password");

oos.writeObject(payload);

\*Client can’t move forward without that message, because:

**if** (payload.type == (**byte**)3) {

String tempToken;

tempToken = ((Payload) connectionToServer.ois.readObject()).toString();

connectionToServer.users.put(temp,tempToken);

//following println is to show result of registered users and unique tokens

System.***out***.println(connectionToServer.users);

**break**;

}

**if**(payload.type == (**byte**)2) {

**break**;

}

\*Code above is normally in a while loop and for user move, he should break the while loop which happens with auth\_Success message (In fact it’s happening also in auth\_Fail message but in result, client and server do not register the users, it just breaks the line and user starts again ).

\*auth\_Challange works samely, nothing happens when server sends auth\_Challange message but there is a counter:

i=1;(i starts from 1 because first time client send password is in outside of the loop)

**while** (!message.equals("quit")||i!=3) {

\*

\*

\*

i++;

}

\*Because of that counter, while he he enters the password, if client does not take auth\_Success message for 3 times, he should start over.

Querying:

\*If user choses the querying phase, first system will ask his username, if there is a user that has this name in server’s hashmap(builted based on userInfo.txt) that user will pass the first phase of protocol and system will take consider its requests. That is possible because of these lines:

payload = (Payload) ois.readObject();

line = payload.toString();

noResponse(line);

(in line above, userInfoList is the users that system recorded at the beginning, server just holds the usernames in this phase, clients are not should be necessarily authorized to pass this phase)

**if**(userInfoList.contains(line)) {

\*

\*

\*

These lines are actual progress

\*

\*

}

**else** {

\*

\*

\*

payload = **new** Payload((**byte**)0,(**byte**)3,"No Response - no valid username - type quit for trying again ");

oos.writeObject(payload);

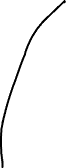
\*

\*

\*It should be kept in mind that, until now, client does not need to be authorized. Only usernames are being kept by server**, he only should enter a username that exists in the** **userinfo.txt.**



\*If the client enters a valid username, process starts. Clients will append their token to head of their strings. System does that automatically if they are authorized, via a hashMap that keeps info of authorized users and their tokens. However, the clients that has not authorized haven’t got tokens and system will append “000000” to head of their strings:



String tokenTemp;

**if**(connectionToServer.users.containsKey(message)) {



tokenTemp=connectionToServer.users.get(message);

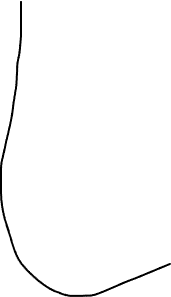
}

**else** {

tokenTemp = "000000";

}

String headerPlusMessage;



\*

\*

\*

headerPlusMessage = tokenTemp + message;

\*Then, server will perform hash function to the usernames that asked first (not in start of authentication phase, starting of querying phase, the username that has been indicated 2 headliner above, **with bold**). And compare them with the token appended head of the string that client enters. With these lines:

**if**(line.substring(0,6).compareTo(Integer.*toString*((username+"29").hashCode()).substring(0, 6)) == 0) {

\*If client has a valid token, line above will be true. Otherwise that lines will be executed:

**else** {

payload = **new** Payload((**byte**)1,(**byte**)3,"No Response - no valid token - type quit for trying again");

oos.writeObject(payload);

oos.flush();

oos.reset();

}

\*Then, user should write 1 or 0 at starting of their inputs (inputs via taken by scanner). Result of that, payload structure’s type byte will be set:

**byte** tempByte;

**if**(message.substring(0, 1).compareTo("1") == 0) {

tempByte=(**byte**)1;

}

**else** **if** (message.substring(0, 1).compareTo("0") == 0){

tempByte=(**byte**)0;

}

**else** {

tempByte= (**byte**)3;

}

payload = **new** Payload((**byte**)1,tempByte,headerPlusMessage);

connectionToServer.sendPayload (payload);

\*According to that type info, server will get article or journal from api and transmit the data’s title, year and topics information to the client. With these lines:

**if**(payload.type == (**byte**)0) {

String correctData = changeJsonData(getDataFromApi(0,line.substring(7, line.length())));

ayload = **new** Payload((**byte**)1,(**byte**)0,correctData);

oos.writeObject(payload);

}

**else** **if**(payload.type == (**byte**)1) {

String correctData = changeJsonData(getDataFromApi(1,line.substring(7, line.length())));

payload = **new** Payload((**byte**)1,(**byte**)1,correctData);

oos.writeObject(payload);

}

**else**{

payload = **new** Payload((**byte**)1,(**byte**)3,"Append 0 or 1 to head of your string");

oos.writeObject(payload);

\*In first phase, user cannot exit from the phase without authentication or receiving “wrong password 3 times - no tries left” message from server. But during phase 2, client can break the loop anytime with entering input “quit”, and then he will go to the “which phase” phase.

\*There is a 20 seconds timeout on every input scanner get from user, if user does not respond in 20 seconds, connection will be closed.

\*Server will break the connection if the client enters the input “no request status received”.

\*Usernames and passwords are in userinfo.txt, first username then in next line there is password associated. Password authentications are based on that structure, if places of lines will be changed, correct output cannot be received. If any user should be add, it should be added as follows: to first line the username, below line the password associated.

\*I used some lines from PS-2 codes but they do not affect the main solution or protocols. They are just methods like disconnecting the connection etc.

Some execution results:

Graphical user interface, text, application, email

Description automatically generated.Graphical user interface, text, application

Description automatically generated with medium confidenceGraphical user interface, text, application

Description automatically generatedGraphical user interface, text

Description automatically generated with medium confidence