CSE3011 NETWORK PROGRAMMING

LAB EXPERIMENT 9

NAME – B PRATYUSH

REGISTRATION NUMBER – 19BCN7114

LAB SLOT – L1+L2

FACULTY – PROF. MUNEESWARI

Experiment Description: Implement Secured Socket Communication between Client and Server

Server Side Code

SecureServer.java

```
import java.util.*;
import java.io.*;
import java.net.*;

class TimeOutTask extends TimerTask {
	boolean isTimedOut = false;

	public void run() {
		isTimedOut = true;
	}
}

public class SecureServer {
	public static void main(String args[]) throws IOException {
```

```
ServerSocket serverSocket = new ServerSocket(7777);
System.out.println("Server running and waiting for client...");
Socket clientSocket = serverSocket.accept();
PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
DataInputStream din=new
DataInputStream(clientSocket.getInputStream());
DataOutputStream dout=new
DataOutputStream(clientSocket.getOutputStream());
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));

// Server waits for a client to send its user ID
String id = in.readLine();

// Server generates an OTP and waits for client to send this
Random r = new Random();
String otp = new String();
for(int i=0 ; i < 8 ; i++) {
        otp += r.nextInt(10);
}
System.out.println(otp);

// Server starts a timer of 10 seconds during which the OTP is
valid.
TimeOutTask task = new TimeOutTask();
```

```
Timer t = new Timer();
t.schedule(task, 100000L);

// Server listens for client to send its ID and OTP to check if it is
// valid
String newId = in.readLine();
String newOtp = in.readLine();
String str="",str2="";
if(newId.equals(id)) {
	// User ID is verified
	if(task.isTimedOut) {
		// User took more than 100 seconds and hence the
OTP is invalid
		out.println("Time out!");
	} else if(!newOtp.equals(otp)) {
		out.println("Incorrect OTP!");
	} else {
		out.println("Logged In!");
		while(!str.equals("stop")){
			str=din.readUTF();
			System.out.println("client says: "+str);
			str2=br.readLine();
			dout.writeUTF(str2);
			dout.flush();
			}
			din.close();
	}
```

```
                }
                clientSocket.close();
                serverSocket.close();
                System.exit(0);
        }
}
```

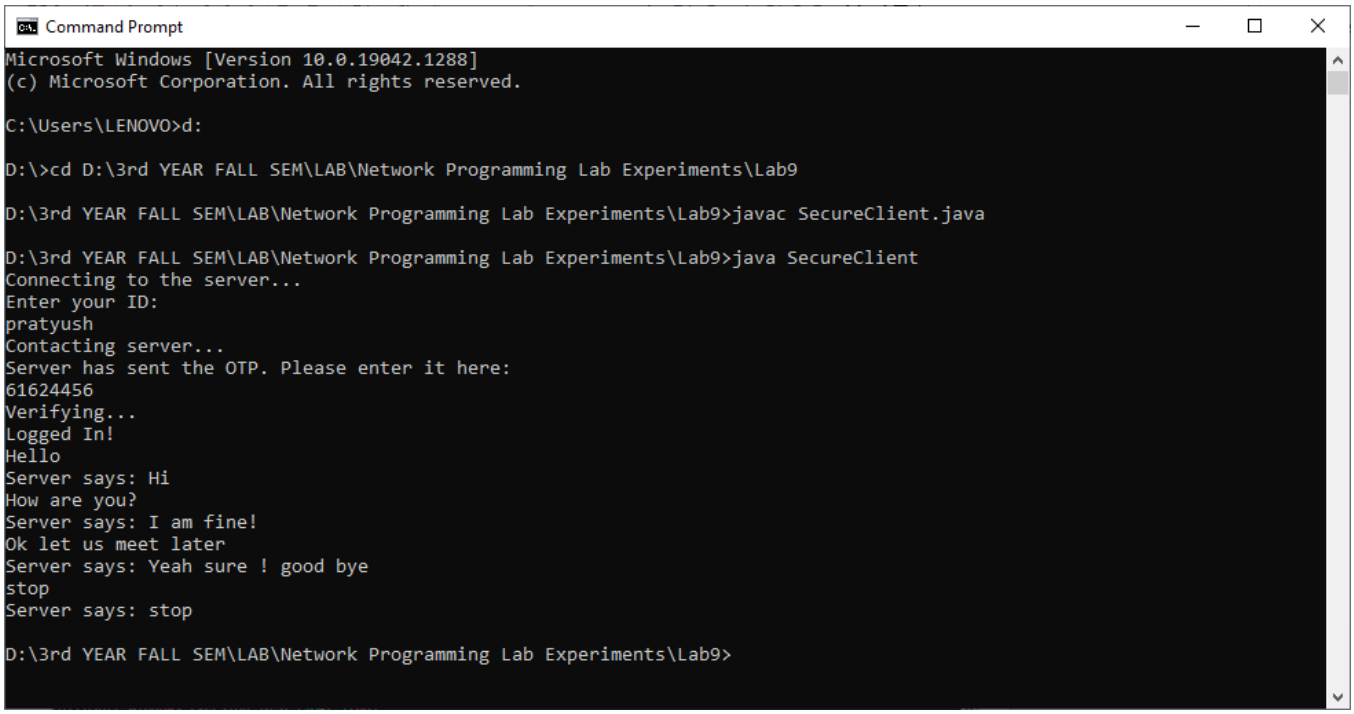
Client Side Code

SecureClient.java

```
import java.util.*;

import java.io.*;

import java.net.*;


public class SecureClient {

	public static void main(String args[]) throws IOException {

		Scanner scan = new Scanner(System.in);

		System.out.println("Connecting to the server...");

		Socket clientSocket = new Socket("localhost", 7777);

		PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true);

		BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

		DataInputStream din=new
DataInputStream(clientSocket.getInputStream());

		DataOutputStream dout=new
DataOutputStream(clientSocket.getOutputStream());

		BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));


		// Client enters ID. This will be used by the program for verifying
who

		// is communicating as well as check the OTP against the ID, on
the

		// server side

		System.out.println("Enter your ID:");

		String id = scan.nextLine();
```

```
                System.out.println("Contacting server...");
                out.println(id);
                System.out.println("Server has sent the OTP. Please enter it
here:");
                String otp = scan.nextLine();
                System.out.println("Verifying...");
                out.println(id);
                out.println(otp);
                System.out.println(in.readLine());
                String str="",str2="";
                while(!str.equals("stop")){
                        str=br.readLine();
                        dout.writeUTF(str);
                        dout.flush();
                        str2=din.readUTF();
                        System.out.println("Server says: "+str2);
                        }

                        dout.close();

                in.close();
                out.close();
                clientSocket.close();
        }
}
```

Client Side Output



Server Side Output