

GROUP CHAT APPLICATION

Computer Science Workshop 2 (CSE 3141)

End-Term Project

PROBLEM STATEMENT

GUI based Chatting Application

It is an implementation of multithreaded server. It accepts connections from an arbitrary number of clients; any message sent from one client is broadcast to all clients. In addition to ServerSockets, it demonstrates the use of threads. Because there are interactions among clients, this server needs to keep track of all the clients it has at any one time.

Instruction for implementation:

• Design GUI based inter face for client server communication.

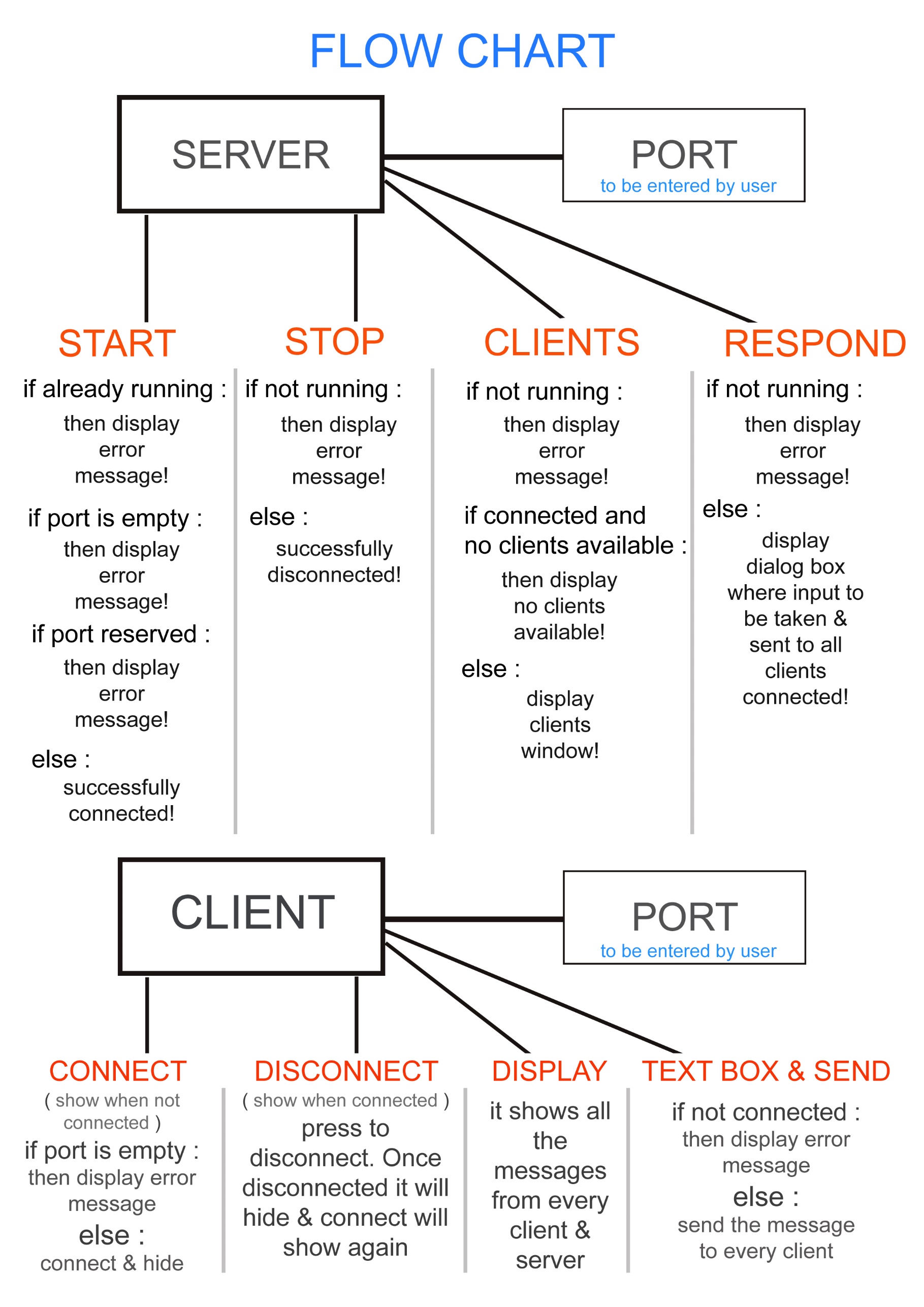
• Server window should contain at least four buttons (START, STOP, RESPOND and ONLINE USERS). The start and stop button to start and stop the server, respond button to respond the client and online user will display the number of clients connected to the server.

• Client window should contain at least three buttons (CONNECT, DISCONNECT, SEND). Connect and disconnect to establish and close connection with server. By clicking the send button client will send the message. Client windows must display the client names like client1, client2 etc. The messages should be displayed in the text area.

• You can add more features and customize your chat application to make it more user friendly.

PROBEM ANALYSIS

* We want to create an application by which we can have group conversations between us.
* Each member will join a fixed port and they can all communicate with each other.
* We will be creating a server though which all interactions are going to take place. The server is going to manage this using multithreading concept.
* The server can also send responses to all its clients connected with it.
* A client will send a message and that will be visible to every other member connected to the server including the client itself.
* Now the Server and client windows are going to be GUI Based Window and will be using the Swing Framework of JAVA.



CODING PART (JAVA)

// The Server program starts at **Server.java**

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** java.awt.Font;

**import** javax.swing.SwingConstants;

**import** java.awt.Color;

**import** javax.swing.JSeparator;

**import** javax.swing.UIManager;

**import** javax.swing.JTextField;

**import** javax.swing.JButton;

**import** java.awt.event.ActionListener;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**import** java.util.ArrayList;

**import** java.awt.event.ActionEvent;

**public** **class** Server {

**private** JFrame frame;

**private** JTextField textField;

**private** **boolean** serverStatus;

**private** ServerSocket listener;

**private** **static** ArrayList<ClientHandler> *clients*;

**private** ServerThread serverThread;

/\*\*

\* Launch the application.

\*/

**public** **static** **void** main(String[] args) {

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

Server window = **new** Server();

window.frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

**public** Server() {

**this**.serverStatus=**false**;

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

**private** **void** initialize() {

frame = **new** JFrame();

frame.getContentPane().setBackground(Color.***WHITE***);

frame.setBounds(100, 100, 535, 710);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.getContentPane().setLayout(**null**);

frame.setTitle("SERVER");

frame.setResizable(**false**);

JLabel lblNewLabel = **new** JLabel("SERVER");

lblNewLabel.setForeground(UIManager.*getColor*("Button.disabledForeground"));

lblNewLabel.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel.setFont(**new** Font("Arial", Font.***PLAIN***, 50));

lblNewLabel.setBounds(98, 10, 311, 81);

frame.getContentPane().add(lblNewLabel);

JSeparator separator = **new** JSeparator();

separator.setBounds(20, 82, 480, 5);

frame.getContentPane().add(separator);

textField = **new** JTextField();

textField.setHorizontalAlignment(SwingConstants.***CENTER***);

textField.setFont(**new** Font("Tahoma", Font.***PLAIN***, 25));

textField.setBounds(251, 115, 127, 42);

frame.getContentPane().add(textField);

textField.setColumns(10);

JLabel lblNewLabel\_1 = **new** JLabel("PORT");

lblNewLabel\_1.setFont(**new** Font("Arial", Font.***PLAIN***, 35));

lblNewLabel\_1.setBounds(135, 115, 104, 42);

frame.getContentPane().add(lblNewLabel\_1);

JButton btnNewButton = **new** JButton("START");

btnNewButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**if**(serverStatus==**true**) {

JOptionPane.*showMessageDialog*(**null**,"Server Already Running!");

}**else** {

**if**(textField.getText().trim().equals("")) {

JOptionPane.*showMessageDialog*(**null**,"Please Enter a PORT!");

**return**;

}

**try** {

listener=**new** ServerSocket(Integer.*parseInt*(textField.getText()));

*clients*=**new** ArrayList<>();

serverStatus=**true**;

serverThread=**new** ServerThread(listener, *clients*);

serverThread.start();

JOptionPane.*showMessageDialog*(**null**,"Server Started now at PORT "+textField.getText());

}**catch** (Exception qqq) {

JOptionPane.*showMessageDialog*(**null**,"Something already running in this PORT!");

}

}

}

});

btnNewButton.setForeground(Color.***WHITE***);

btnNewButton.setBackground(Color.***DARK\_GRAY***);

btnNewButton.setBorderPainted(**false**);

btnNewButton.setFont(**new** Font("Arial", Font.***PLAIN***, 40));

btnNewButton.setBounds(131, 186, 258, 81);

frame.getContentPane().add(btnNewButton);

JButton btnStop = **new** JButton("STOP");

btnStop.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**if**(serverStatus==**false**) {

JOptionPane.*showMessageDialog*(**null**,"Server not running!");

}**else** {

serverStatus=**false**;

**try** {

listener.close();

serverThread=**null**;

}**catch** (Exception ej) {

// **TODO**: handle exception

}

}

}

});

btnStop.setForeground(Color.***WHITE***);

btnStop.setFont(**new** Font("Arial", Font.***PLAIN***, 40));

btnStop.setBorderPainted(**false**);

btnStop.setBackground(Color.***DARK\_GRAY***);

btnStop.setBounds(131, 288, 258, 81);

frame.getContentPane().add(btnStop);

JButton btnClients = **new** JButton("CLIENTS");

btnClients.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**if**(serverStatus==**true**) {

**if**(*clients*.size()==0) {

JOptionPane.*showMessageDialog*(**null**,"No clients are available right now!");

}**else** {

AllClients allClients=**new** AllClients(*clients*);

allClients.setVisible(**true**);

}

}**else** {

JOptionPane.*showMessageDialog*(**null**,"Server not started");

}

}

});

btnClients.setForeground(Color.***WHITE***);

btnClients.setFont(**new** Font("Arial", Font.***PLAIN***, 40));

btnClients.setBorderPainted(**false**);

btnClients.setBackground(Color.***DARK\_GRAY***);

btnClients.setBounds(131, 392, 258, 81);

frame.getContentPane().add(btnClients);

JSeparator separator\_1 = **new** JSeparator();

separator\_1.setBounds(20, 606, 480, 5);

frame.getContentPane().add(separator\_1);

JLabel lblNewLabel\_2 = **new** JLabel("made by BISWAMOHAN DWARI");

lblNewLabel\_2.setForeground(Color.***GRAY***);

lblNewLabel\_2.setFont(**new** Font("Arial", Font.***PLAIN***, 20));

lblNewLabel\_2.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel\_2.setBounds(46, 621, 431, 42);

frame.getContentPane().add(lblNewLabel\_2);

JButton btnRespond = **new** JButton("RESPOND");

btnRespond.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**if**(serverStatus==**false**) {

JOptionPane.*showMessageDialog*(**null**,"Server not started yet!");

**return**;

}**else** **if**(*clients*.size()==0) {

JOptionPane.*showMessageDialog*(**null**,"No clients Available now!");

**return**;

}

String msgString=JOptionPane.*showInputDialog*("Enter your response to all users!");

**if**(msgString==**null**) **return**;

**for**(ClientHandler crawl:*clients*) crawl.getOutputChannel().println("[SERVER] : "+msgString);

}

});

btnRespond.setForeground(Color.***WHITE***);

btnRespond.setFont(**new** Font("Arial", Font.***PLAIN***, 40));

btnRespond.setBorderPainted(**false**);

btnRespond.setBackground(Color.***DARK\_GRAY***);

btnRespond.setBounds(131, 493, 258, 81);

frame.getContentPane().add(btnRespond);

}

}

// Now this Server.java needs three java files…

// 1. ClientHandler.java

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.PrintWriter;

**import** java.net.Socket;

**import** java.util.ArrayList;

**import** java.util.Date;

**import** java.util.Random;

**public** **class** ClientHandler **extends** Thread{

**private** Socket client;

**private** String nameString;

**private** BufferedReader in;

**private** PrintWriter out;

**private** ArrayList<ClientHandler> clients;

**public** **boolean** isAvailable() {

**return** !client.isClosed();

}

**public** PrintWriter getOutputChannel() {

**return** **this**.out;

}

**public** ClientHandler(Socket clientSocket,ArrayList<ClientHandler> clients) **throws** Exception{

**this**.client=clientSocket;

**this**.in=**new** BufferedReader(**new** InputStreamReader(client.getInputStream()));

**this**.out=**new** PrintWriter(client.getOutputStream(),**true**);

**this**.clients=clients;

Random random=**new** Random();

**this**.nameString=((**char**)(65+random.nextInt(25)))+""

+((**char**)(97+random.nextInt(25)))+""

+((**char**)(97+random.nextInt(25)))+""

+((**char**)(97+random.nextInt(25)))+""

+((**char**)(97+random.nextInt(25)));

}

**public** String getIPAdressString() {

**return** client.getInetAddress().toString();

}

**public** String getNameString() {

**return** **this**.nameString;

}

**public** String getTimeOfJoining() {

**return** **new** Date().toString();

}

**public** **void** outToAll(String msg) {

**for**(ClientHandler crawl:clients) crawl.out.println(msg);

}

@Override

**public** **void** run() {

**try** {

**while**(**true**) {

String requestString=in.readLine();

**if**(requestString!=**null**) {

// System.out.println(requestString); // To print what the client has sent

outToAll(getNameString()+" : "+requestString);

}

// out.println("Gotcha!"); // use this in debugging

}

} **catch** (Exception e) {

// **TODO**: handle exception

} **finally** {

out.close();

**try** {

in.close();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

}

// 2. ServerThread.java

**import** java.io.IOException;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**import** java.util.ArrayList;

**public** **class** ServerThread **extends** Thread{

**private** ServerSocket listener;

**private** ArrayList<ClientHandler> clients;

**public** ServerThread(ServerSocket listener,ArrayList<ClientHandler> clients) {

**this**.listener=listener;

**this**.clients=clients;

}

**public** **void** run() {

**while**(**true**) {

System.***out***.println("[SERVER] Waiting for client connection...");

Socket client=**null**;

**try** {

client = listener.accept();

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

}

System.***out***.println("[SERVER] Connected to client...");

ClientHandler clientThread=**null**;

**try** {

clientThread = **new** ClientHandler(client,clients);

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

}

clients.add(clientThread);

clientThread.start();

}

}

}

// 3. AllClients.java

**import** java.awt.BorderLayout;

**import** java.awt.EventQueue;

**import** java.util.ArrayList;

**import** javax.swing.DefaultListModel;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** javax.swing.border.EmptyBorder;

**import** javax.swing.JList;

**import** java.awt.Font;

**public** **class** AllClients **extends** JFrame {

**private** JPanel contentPane;

**private** ArrayList<ClientHandler> clients;

/\*\*

\* Launch the application.

\*/

**public** **static** **void** main(String[] args) {

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

AllClients frame = **new** AllClients(**null**);

frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

**public** AllClients(ArrayList<ClientHandler> clients) {

**this**.clients=clients;

setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

setBounds(100, 100, 878, 606);

contentPane = **new** JPanel();

contentPane.setBorder(**new** EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(**null**);

setTitle("Available Clients...");

JList list = **new** JList();

list.setFont(**new** Font("Arial", Font.***PLAIN***, 25));

list.setBounds(10, 10, 844, 549);

contentPane.add(list);

DefaultListModel<String> model=**new** DefaultListModel<>();

list.setModel(model);

**for**(ClientHandler client : clients)

**if**(client.isAvailable()) model.addElement(client.getIPAdressString()+" : "+client.getNameString()+" : "

+client.getTimeOfJoining());

}

}

// This is all the Server program.

// Next page – Client Program

// The Server program starts at **Client.java**

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** java.awt.Font;

**import** java.awt.Color;

**import** javax.swing.JTextField;

**import** javax.swing.SwingConstants;

**import** javax.swing.DefaultListModel;

**import** javax.swing.JButton;

**import** java.awt.event.ActionListener;

**import** java.io.PrintWriter;

**import** java.net.Socket;

**import** java.util.Scanner;

**import** java.awt.event.ActionEvent;

**import** javax.swing.JList;

**import** javax.swing.JOptionPane;

**import** java.awt.SystemColor;

**public** **class** Client {

**private** JFrame frame;

**private** JTextField textField;

**private** JTextField textField\_1;

**private** **boolean** status=**false**;

**private** Socket socket;

**private** PrintWriter out;

/\*\*

\* Launch the application.

\*/

**public** **static** **void** main(String[] args) {

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

Client window = **new** Client();

window.frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

**public** Client() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

**private** **void** initialize() {

frame = **new** JFrame();

frame.setTitle("Client");

frame.getContentPane().setBackground(Color.***WHITE***);

frame.setBounds(100, 100, 541, 768);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.getContentPane().setLayout(**null**);

frame.setResizable(**false**);

textField = **new** JTextField();

textField.setFont(**new** Font("Arial", Font.***PLAIN***, 25));

textField.setHorizontalAlignment(SwingConstants.***CENTER***);

textField.setBounds(389, 10, 128, 39);

frame.getContentPane().add(textField);

textField.setColumns(10);

JLabel lblNewLabel\_1 = **new** JLabel("PORT");

lblNewLabel\_1.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel\_1.setFont(**new** Font("Arial", Font.***PLAIN***, 20));

lblNewLabel\_1.setBounds(313, 10, 66, 39);

frame.getContentPane().add(lblNewLabel\_1);

JList list = **new** JList();

list.setFont(**new** Font("Arial", Font.***PLAIN***, 25));

list.setBackground(SystemColor.***menu***);

list.setBounds(10, 59, 507, 599);

frame.getContentPane().add(list);

DefaultListModel<String> model=**new** DefaultListModel<>();

list.setModel(model);

JButton btnNewButton = **new** JButton("CONNECT");

JButton btnDisconnect = **new** JButton("DISCONNECT");

// for connect button

btnNewButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**if**(textField.getText().trim().equals("")) {

JOptionPane.*showMessageDialog*(**null**,"Enter the PORT!");

**return**;

}

**try** {

socket=**new** Socket("localhost",Integer.*parseInt*(textField.getText()));

ServerConnection serverConnection=**new** ServerConnection(socket,model);

out=**new** PrintWriter(socket.getOutputStream(),**true**);

serverConnection.start();

} **catch** (Exception e2) {

JOptionPane.*showMessageDialog*(**null**,"Error Connecting to the PORT "+textField.getText());

**return**;

}

status=**true**;

JOptionPane.*showMessageDialog*(**null**,"Connected to PORT "+textField.getText());

btnNewButton.setVisible(**false**);

btnDisconnect.setVisible(**true**);

}

});

btnNewButton.setBackground(Color.***DARK\_GRAY***);

btnNewButton.setForeground(Color.***WHITE***);

btnNewButton.setBorderPainted(**false**);

btnNewButton.setFont(**new** Font("Arial", Font.***PLAIN***, 15));

btnNewButton.setBounds(10, 10, 128, 39);

frame.getContentPane().add(btnNewButton);

// for disconnect button

btnDisconnect.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

status=**false**;

**try** {

socket.close();

}**catch**(Exception e1) {

}

JOptionPane.*showMessageDialog*(**null**,"Disconnected Successfully!");

btnDisconnect.setVisible(**false**);

btnNewButton.setVisible(**true**);

}

});

btnDisconnect.setForeground(Color.***WHITE***);

btnDisconnect.setFont(**new** Font("Arial", Font.***PLAIN***, 15));

btnDisconnect.setBorderPainted(**false**);

btnDisconnect.setBackground(Color.***DARK\_GRAY***);

btnDisconnect.setVisible(**false**);

btnDisconnect.setBounds(148, 10, 155, 39);

frame.getContentPane().add(btnDisconnect);

textField\_1 = **new** JTextField();

textField\_1.setFont(**new** Font("Arial", Font.***PLAIN***, 20));

textField\_1.setBounds(10, 668, 431, 53);

frame.getContentPane().add(textField\_1);

textField\_1.setColumns(10);

JButton btnNewButton\_1 = **new** JButton(">");

btnNewButton\_1.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**if**(status==**false**) {

JOptionPane.*showMessageDialog*(**null**,"Hey! Connect First!");

}**else** {

**if**(!textField\_1.getText().trim().equals(""))

out.println(textField\_1.getText());

}

}

});

btnNewButton\_1.setForeground(Color.***WHITE***);

btnNewButton\_1.setFont(**new** Font("Arial", Font.***PLAIN***, 45));

btnNewButton\_1.setBorderPainted(**false**);

btnNewButton\_1.setBackground(Color.***DARK\_GRAY***);

btnNewButton\_1.setBounds(451, 668, 66, 55);

frame.getContentPane().add(btnNewButton\_1);

}

}

// This Client.java needs a java file

// ServerConnection.java

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.net.Socket;

**import** javax.swing.DefaultListModel;

**public** **class** ServerConnection **extends** Thread{

**private** Socket server;

**private** BufferedReader in;

**private** DefaultListModel<String> list;

**public** ServerConnection(Socket s,DefaultListModel<String> model) **throws** Exception{

**this**.server=s;

**this**.list=model;

**this**.in=**new** BufferedReader(**new** InputStreamReader(server.getInputStream()));

}

**public** **void** run() {

**try** {

**while**(**true**) {

String reString=in.readLine();

**if**(reString==**null**) **break**;

list.addElement(reString);

}

} **catch** (Exception e) {

// **TODO**: handle exception

} **finally** {

**try** {

in.close();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

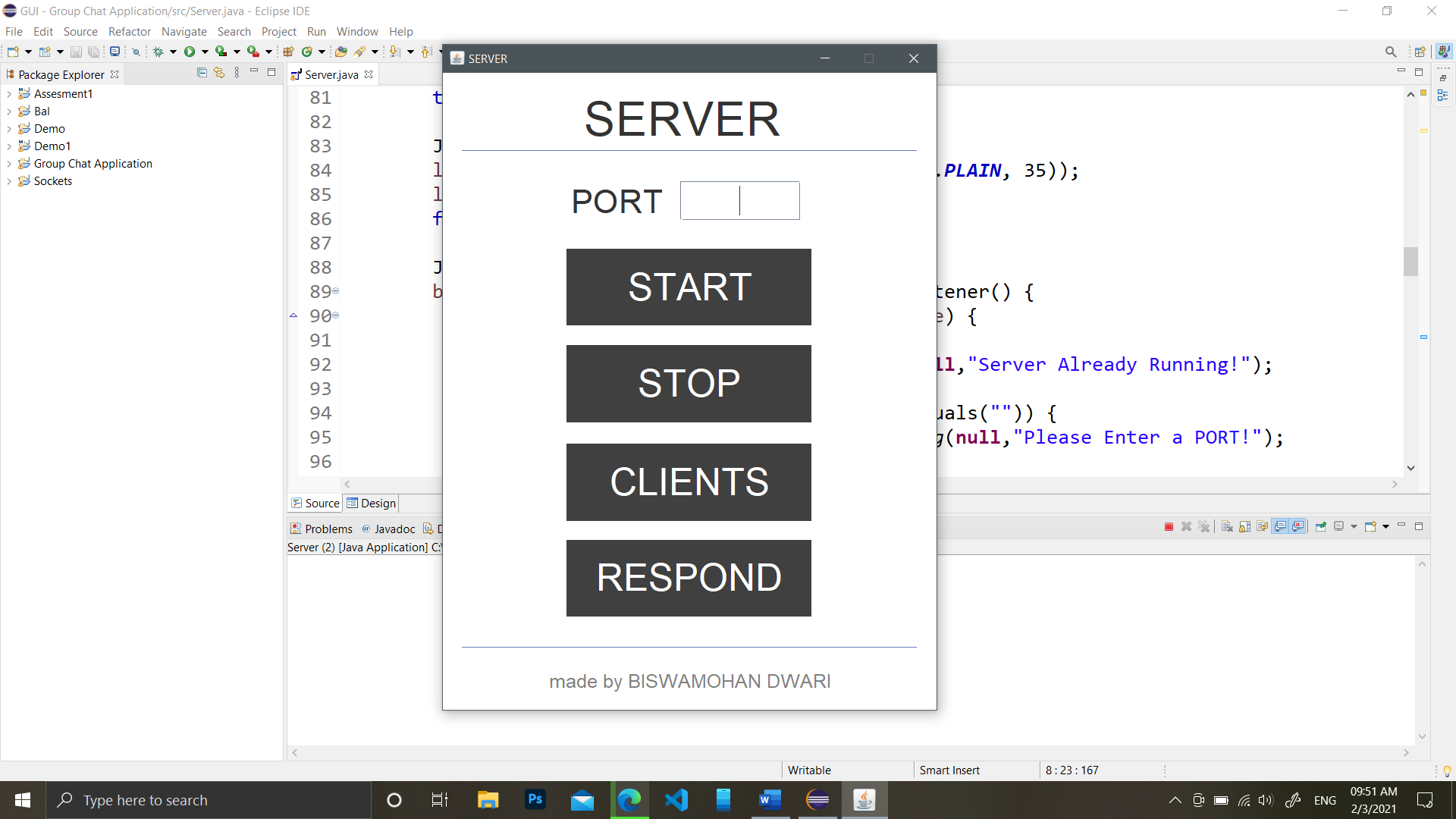
}

}

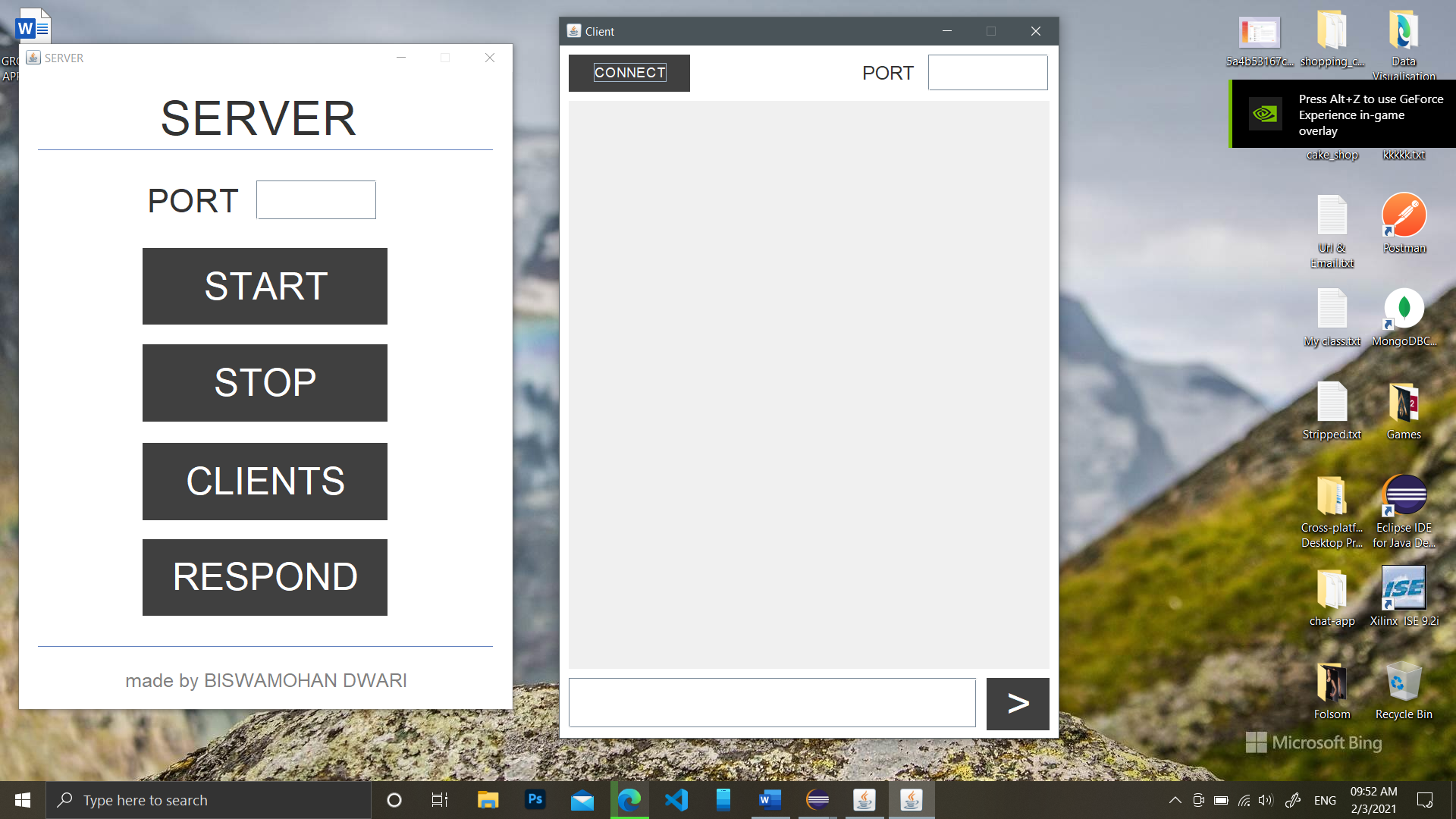
}

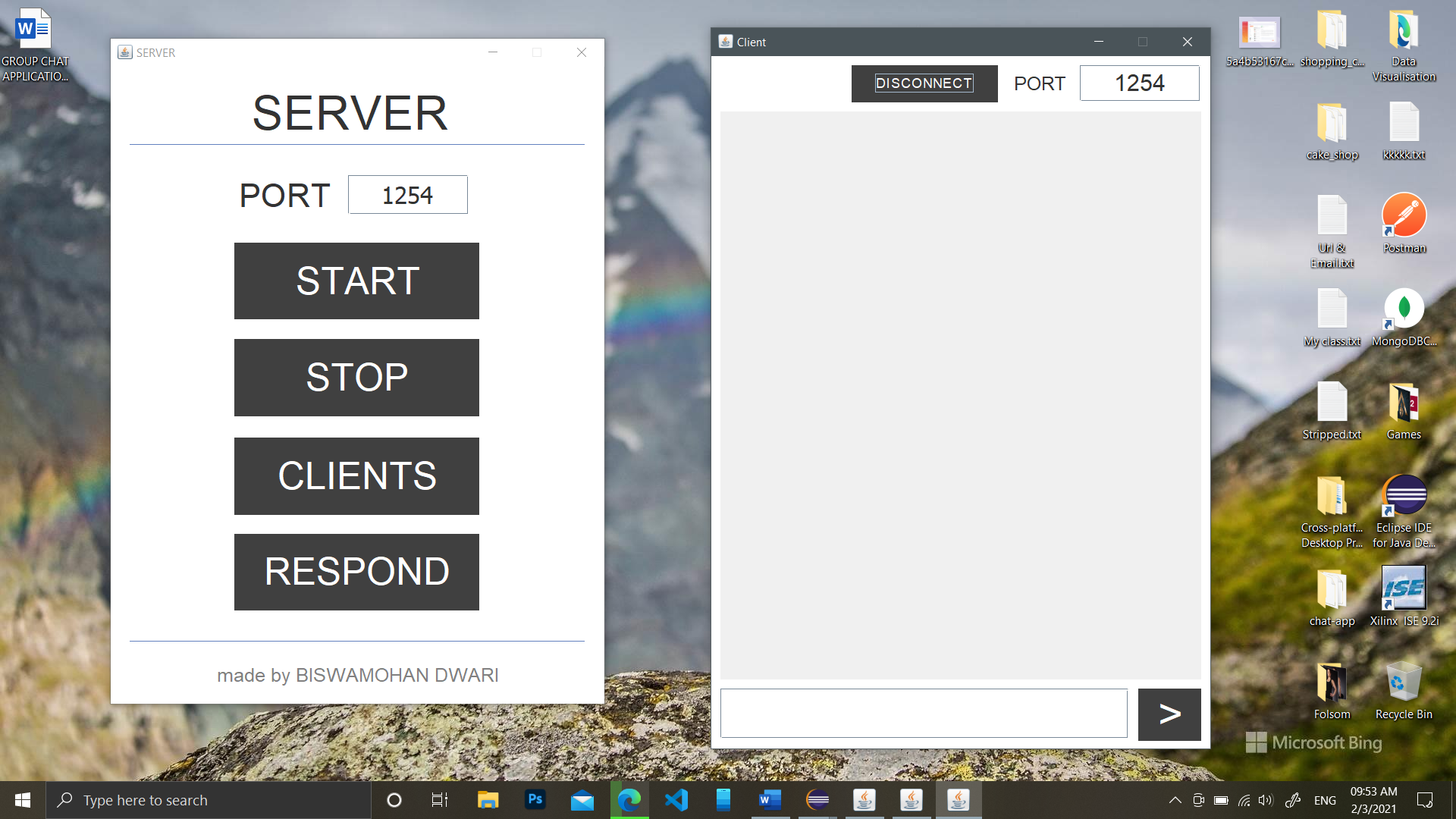
Snapshot of input and output

// Server Window

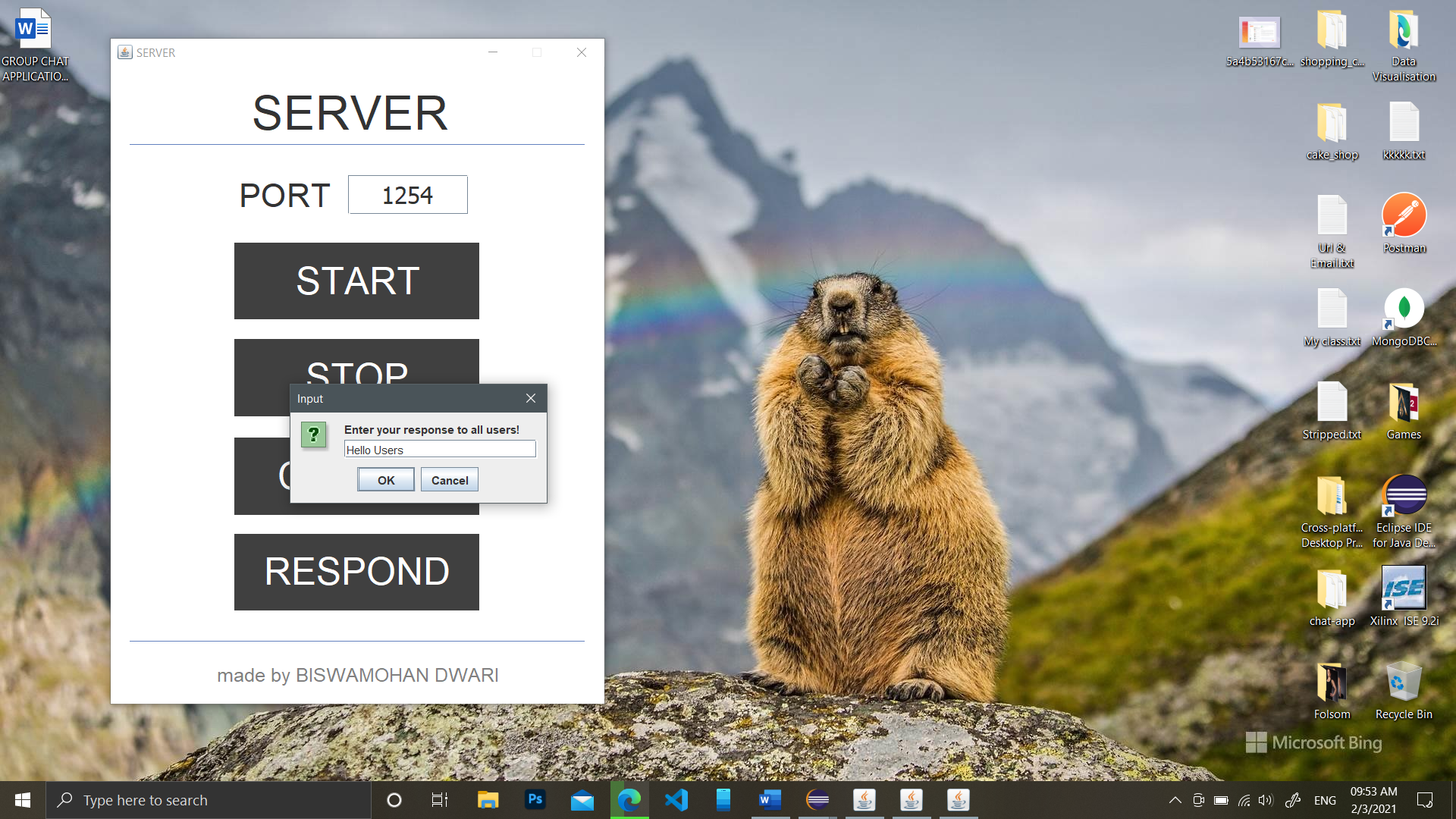


// Client Window

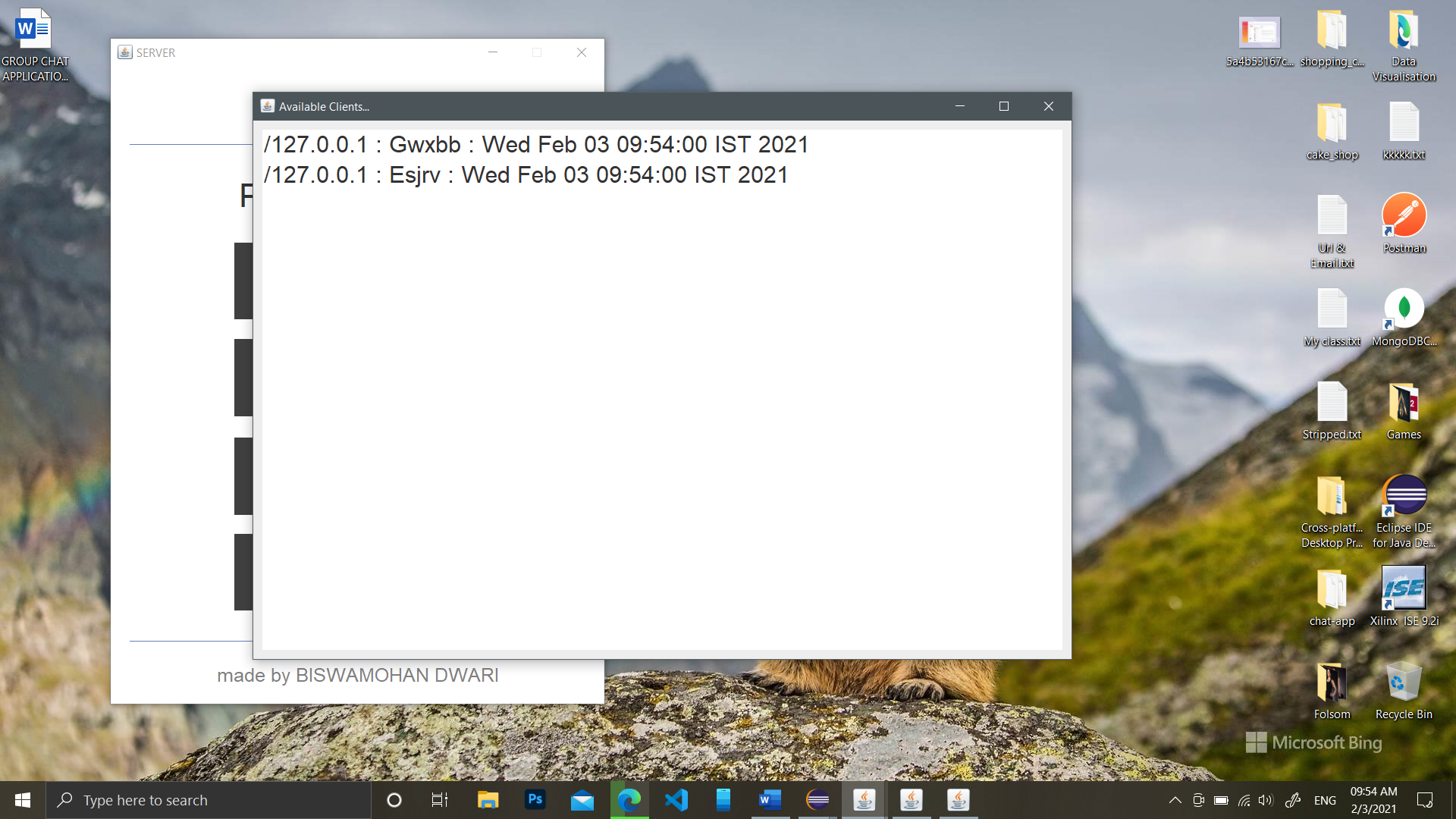


// Server and Client Connected

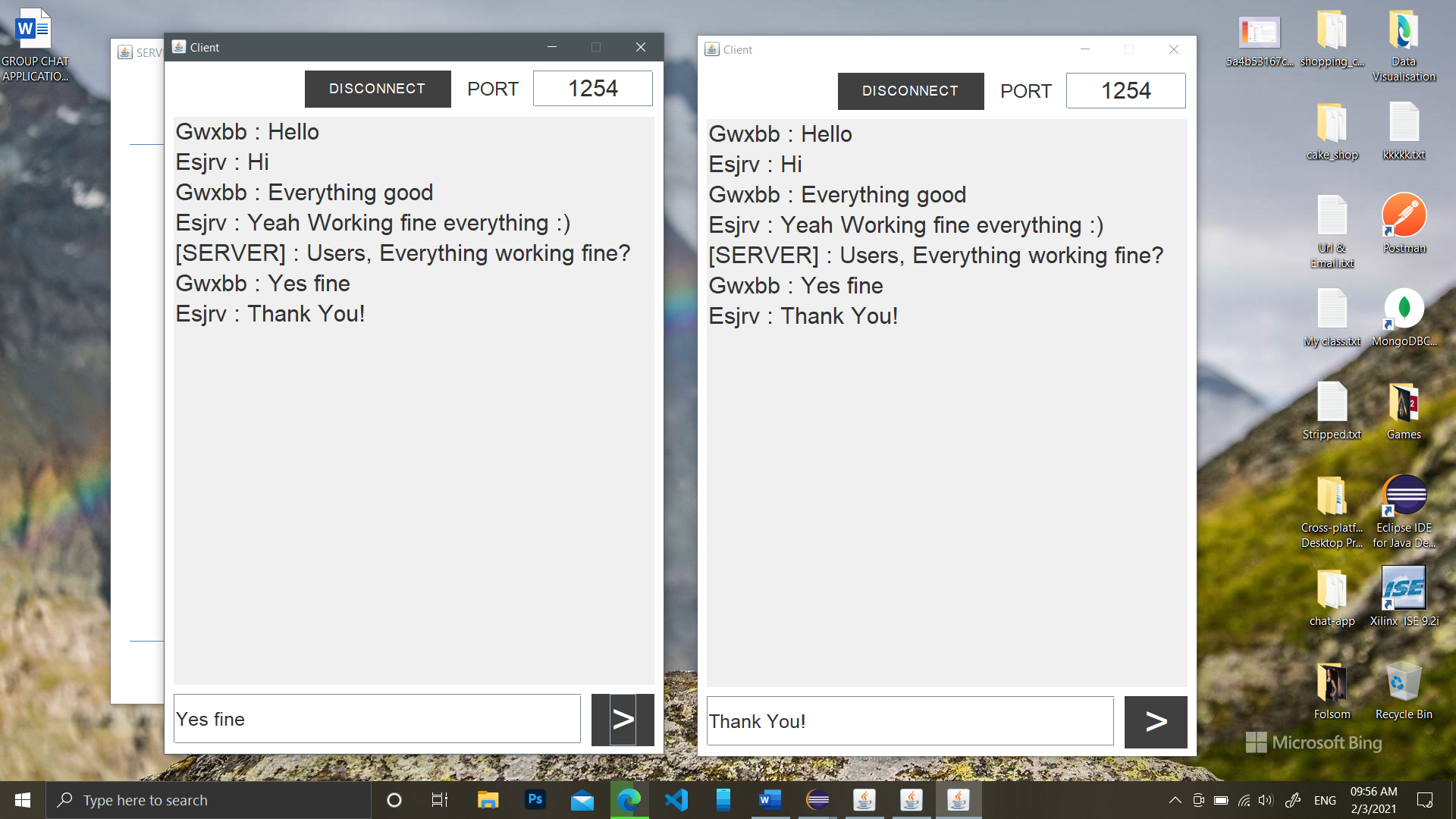
// Server sending responses



// Server Showing All Available Clients…



// Two Clients talking with each other and server sending message…



// notice that the clients are given username randomly by sever…

Author : BISWAMOHAN DWARI

Dt : 02/03/2021

THANK YOU