**IBM CAREER EDUCATION**

**MAIN PROJECT**

**DOMAIN NAME: JAVA**

**CHAT APPLICATION**

**Submitted By,**

Drishti Motwani (18162171013) ,

Jinal Rathore (18162171007)

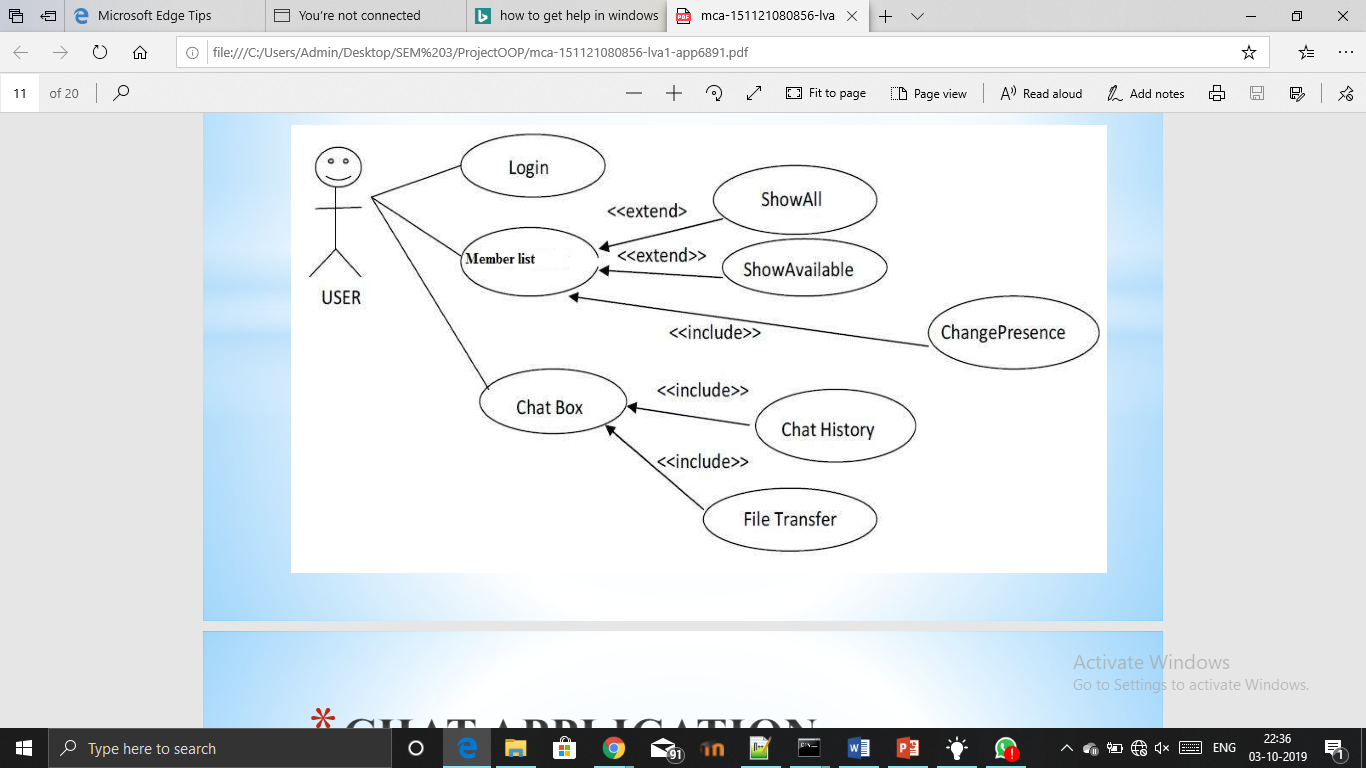
II Year – (CS) ‘A’ Section Ganpat University, Ahmedabad.

**Submitted To,**

A. Saai Sanjeev Achaarya

IBM Software Technical Trainer

**FLOWCHART**



**SOFTWARE SPECIFICATIONS**

* OPERATING SYSTEM : Windows
* ENVIRONMENT : IBM RAD Software

**HARDWARE SPECIFICATIONS**

|  |  |  |
| --- | --- | --- |
|  PROCESSOR | : | PENTIUM IV 2.8MHz |
|  RAM | : | 256 MB SD RAM |
|  MONITOR | : | 15” COLOR |
|  HARD DISK | : | 40 GB |
|  FLOPPY DRIVE | : | 1.44 MB |

**DESCRIPTION**

**INTRODUCTION:**

Chatting refers to any kind of communication that offers a real-time transmission of messages from sender to the receiver. Chatting is a method of using technology to bring people and ideas together despite the geographical barriers. The technology to provide the chatting facility has been available for years, but the acceptance is quite recent. Analysis of chatting provides an overview of the technologies used, available features, functions, system of the architecture of the application, the structure of database of an Instant Messaging application: IChat(IC). The objective of IC application is to facilitate text messaging, group chatting option, data transfer without size restriction which is commonly seen in most of the messaging applications.

Chat system is peer-to-peer where the users exchange text messages and files. The users of the system are the client and the server. The architecture is a distributed programming which consists of two components, the server and the client. The client initiates the communication by requesting for the server location information and display the received chat messages. The server conducts the chat session and manages all the client. The client starts the session by requesting for two parameters, the server name and the port number. The client and the server have two type of communication between them. Firstly, control message where one can join and leave chat session, create a chat room. Secondly, chat message where one can send and receive messages, transfer files from or to their contacts.

**DESCRIPTION:**

* Our project is an example of chat application which is basically based on public chatting.
* It is made of two applications.
* Client application which runs on user’s PC.
* Server application which runs on any PC on the network.
* To start chatting client should get connected to server.

Chat communication is the process of exchanging messages between two systems continuously. Anyone can break the communication. Both systems come with the following same responsibilities.

1. Reading from keyboard. Uses an input stream like BufferedReader connected to System.in.
2. Sending data to the other system what is read from keyboard. Uses an output stream like PrintWriter connected to getOutputStream() method of Socket.

Receiving data from the other system. Uses an input stream like BufferedReader connected to getInputStream() method of Socket

**PROS:**

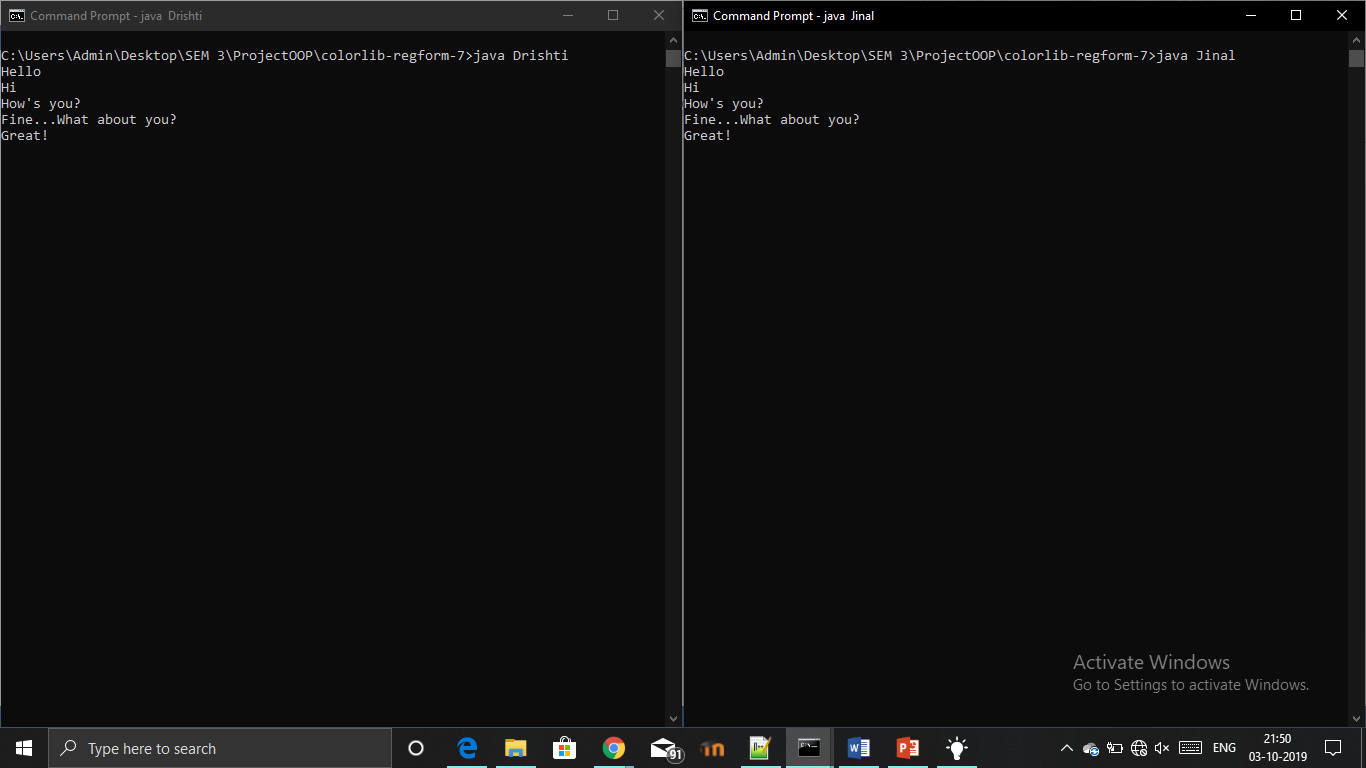
* It seamlessly connects people, cutting across territories.
* Instant messaging provides real-time communication.
* Instant messaging can be a good tool for team building in a workplace.
* Emails easily get cluttered with spam emails, quite unlike when you’re using instant messaging for business.

**CONS:**

* Personal bonding among people seems to be lost.
* Large amount of storage is required to have a history of all chats.
* Users expect quick responses from you, it could easily distract you from your regular work duties. This could, in turn, lead to reduced productivity.
* Instant messaging presents a security threat to your business since they are third party applications.

**CHAT APPLICATION**

**PICTURES**



**CHAT APPLICATION**

**AIM:**

The objective of Chat Application is to facilitate text messaging, group chatting option, data transfer without size restriction which is commonly seen in most of the messaging applications.

**PROGRAM:**

**Server side Java Application:**

import java.io.\*;

import java.net.\*;

public class Jinal

{ public static void main(String[] args)

{ try

{

ServerSocket ss= new ServerSocket (1223);

Socket s = ss.accept ();

DataInputStream in = new DataInputStream( s.getInputStream());

DataOutputStream out = new DataOutputStream( s.getOutputStream());

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

String msgin="";

String msgout="";

while(!msgin.equals("end") )

{

msgin = in.readUTF();

System.out.println(msgin);

msgout = rd.readLine();

out.writeUTF(msgout);

out.flush();

}

s.close();

} catch (Exception e)

{e.printStackTrace();}

}

}

**Client side Java Application**

import java.io.\*;

import java.net.\*;

public class Drishti

{

public static void main( String args[])

{

try

{

Socket s = new Socket("127.0.0.1",1223);

DataInputStream in = new DataInputStream(s.getInputStream());

DataOutputStream out = new DataOutputStream(s.getOutputStream());

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

String msgin="",msgout="";

while(!msgin.equals(" end"))

{

msgout=rd.readLine();

out.writeUTF(msgout);

msgin=in.readUTF();

System.out.println(msgin);

}

s.close();

}

catch (Exception e)

{e.printStackTrace();}

}

}

**EXPLANATION ABOUT PROJECT:**

**Users**

The users will be anyone who has the chat application and registers for it.

**Use cases** These are the use cases for the client of the chat application. The server has access to all of these cases as well.

**Main Menu** When the client runs the chat application, the client will see the main menu, which will welcome them, At the main menu, the client will have the choice to register for the chat application, login to the chat application, or exit it. **Register/Login/Logout** The user must register in order to login, the user must login in order to send messages to those who are online, and must be able to logout if the user wants to logout.

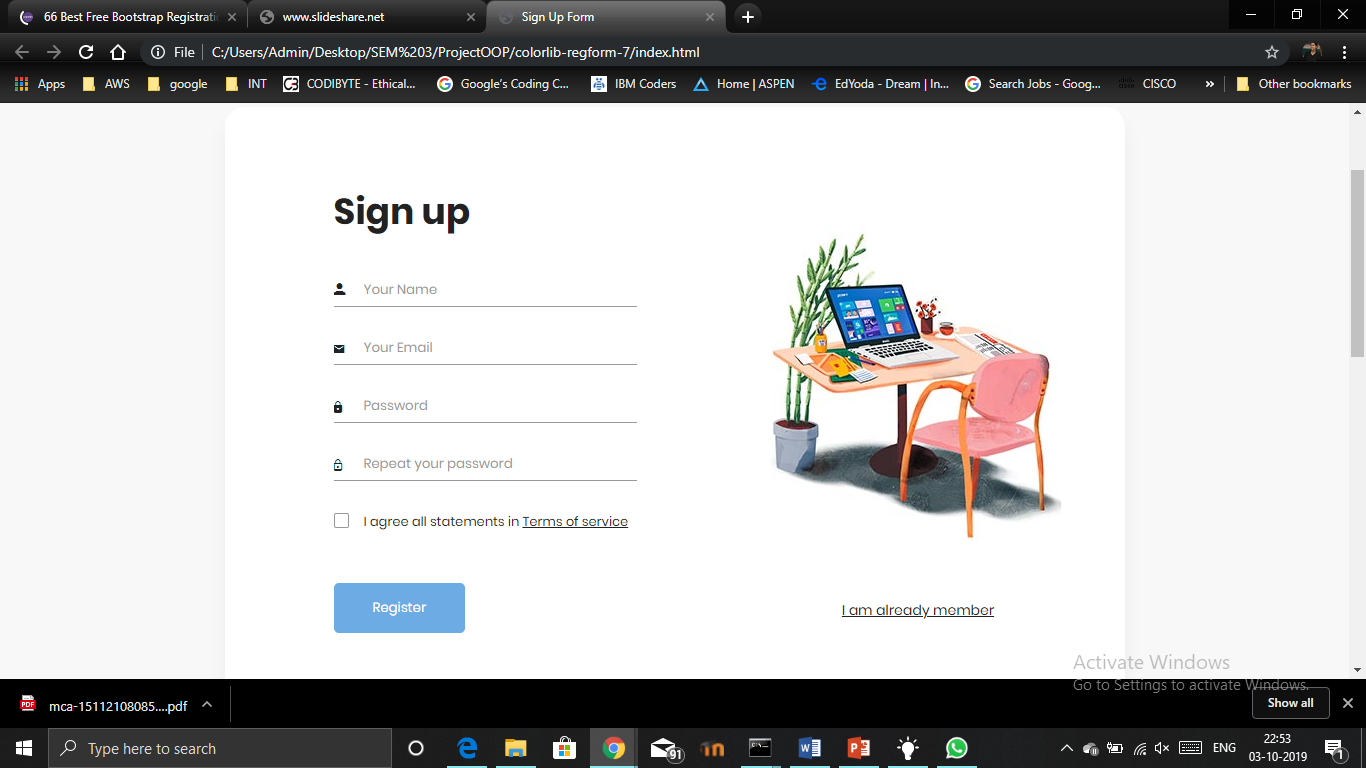
**Online menu** After the client is logged in, the client can choose to send a message, only if another client is online, check who is online, and be able to logout when the client wants to, which will be by logging out by hitting the logout button or by hitting the exit button.

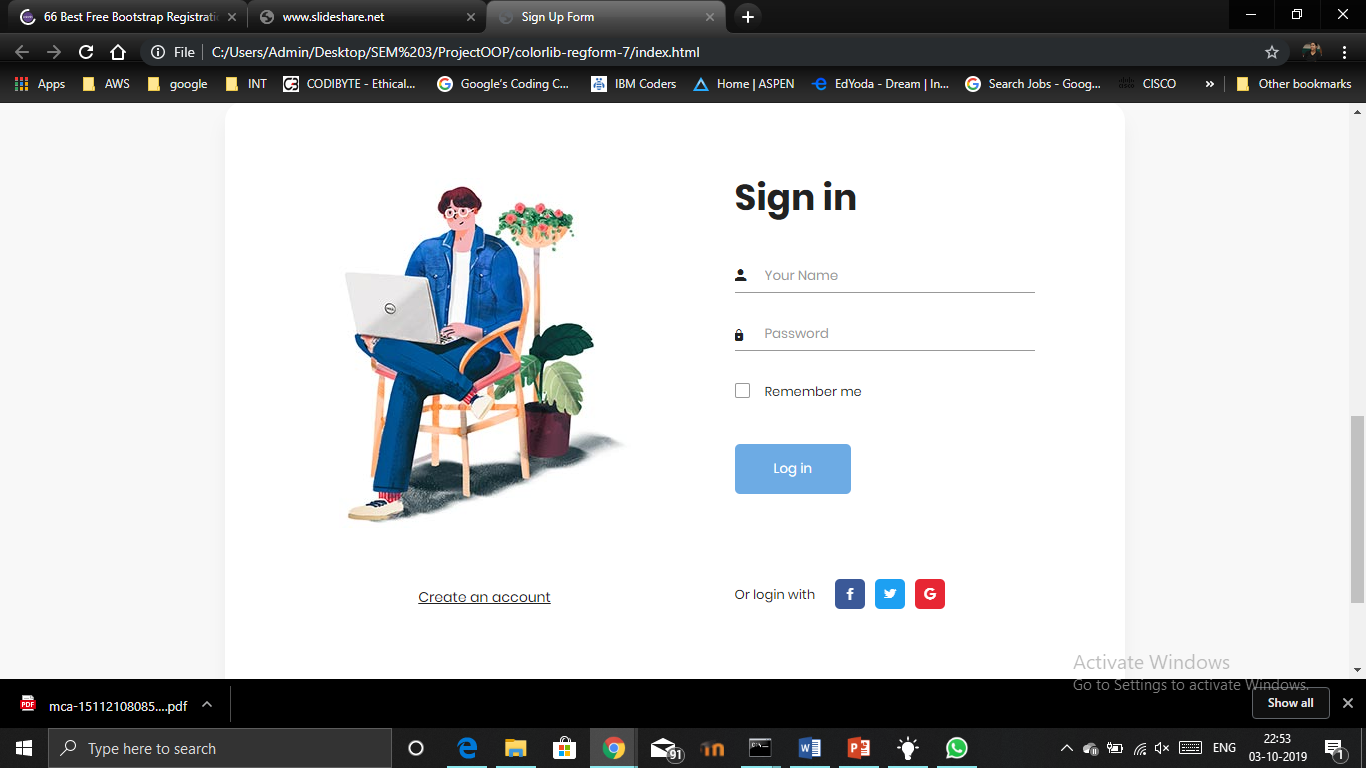
**Users Online** When the client wants to see who is online, the client clicks the “who is online” button in the online menu, and the client can choose who to message, by double clicking the name.

**Message** When the client wants to message the user, the client clicks on the name, the user can send a message to another online user. Some character limitations for typing might be put into place, only if it is necessary.

**Chat history** When the client wants to see the chat history, the user will be able to do so. The user will be able to clear the history if the user wants to.

**OUTPUT SCREENSHOTS:**





**REFERENCES: (3 References Should Be**

**There)**

1. <https://www.slideshare.net/>
2. [https://www.youtube.com](https://www.youtube.com/)
3. <https://colorlib.com/wp/free-bootstrap-registration-forms/>

**CONCLUSION:**

The project has dealt with implementing a chatting application on small scale. It has features such as text messaging, group chat, data transfer. The main objective of the project was to develop an application which provides the data transfers and hence has been implemented.