MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR(M.P.)

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

Skill Based Mini

Project Report on

"CHAT APPLICATION"

Submitted By:

Aman Chourasiya (0901CS223D02)

Aman Gupta (0901CS223D03)

Nishant Shrivastava (0901CS223D08)

Faculty Mentor:

Dr. Rajni Ranjan Singh Makwana, Assistant Professor, CSE

Submitted to:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCEGWALIOR - 474005 (MP) est. 1957

JAN-JUNE 2023

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

CERTIFICATE

This is certified that **Aman Chourasiya (0901CS223D02),Aman Gupta (0901CS223D03), Nishant Shrivastava (0901CS223D08)** has submitted the project report titled CHAT APPLICATION under the mentorship of **Dr. Rajni Ranjan Singh Makwana**, in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering from Madhav Institute of Technology and Science, Gwalior.

[illegible] 22/4/23

Dr. Rajni Ranjan Singh
Faculty Mentor Assistant Professor
Computer Science and Engineering

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

DECLARATION

I hereby declare that the work being presented in this project report, for the partial fulfillment of requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering at Madhav Institute of Technology & Science, Gwalior is an authenticated and original record of my work under the mentorship of **Dr. Rajni Ranjan Singh, Assistant Professor,** Department of CSE I declare that I have not submitted the matter embodied in this report for the awardof any degree or diploma anywhere else.

[illegible] Nishant

Aman.

Aman Gupta (0901CS223D03),
Aman Chourasiya (0901CS223D02)
Nishant Shrivastava (0901CS223D08)
2nd Year,
Computer Science and Engineering

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

ACKNOWLEDGEMENT

The full semester project has proved to be pivotal to my career. I am thankful to my institute, **Madhav Institute of Technology and Science** to allow me to continue my disciplinary/interdisciplinary project as a curriculum requirement, under the provisions of the Flexible Curriculum Scheme (basedon the AICTE Model Curriculum 2018), approved by the Academic Council of the institute. I extendmy gratitude to the Director of the institute, **Dr.R. K. Pandit** and Dean Academics, **Dr. Manjaree Pandit** for this.

I would sincerely like to thank my department, **Department of Computer Science and Engineering, for allowing** me to explore this project. I humbly thank **Dr. Manish Dixit**,Professor and Head, Department of Computer Science and Engineering, for his continued support during the course of this engagement, which eased the process and formalities involved.

I am sincerely thankful to my faculty mentors. I am grateful to the guidance of **Dr. RajniRanjan Singh**, Assistant Professor, Department of CSE, for his continued support and guidance throughout the project. I am also very thankful to the faculty and staff of the department.

[illegible] Nishant
[illegible]

Aman Gupta(0901CS223D03)
Aman Chourasiya(0901CS223D02)
Nishant Shrivastava (0901CS223D08)
2nd Year
Computer Science and Engineering

ABSTRACT

Providing a useful suggestion of products to online users to increase their consumption on websites is the goal of many companies nowadays People usually select or purchase a new product based on some friend’s recommendations, comparison of similar products or feedbacks from other users. In order to do all these tasks automatically, a recommender system must be implemented. The recommender systems are tools that provide suggestions that best suit the client’s needs, even when they are not aware of it. That offers of personalized content are based on past behavior and it hooks the customer to keep coming back to the website. In this report, a **CHAT APPLICATION** will be built.

The main types of recommender algorithm are Titles, Cast, Directors and Description. All of them will be introduced in this report. We will select the algorithms that best fit to the data and we will implement them.

INTRODUCTION

The rise of the internet and its impact on modern-day communication has been tremendous. The ability to communicate with anyone in the world has made life more comfortable, and businesses have leveraged the technology to expand their operations. One of the most popular modes of communication today is messaging applications. With the increased use of smartphones and other mobile devices, people have access to a wide range of messaging applications, and these applications have become an integral part of our daily lives. In this project, we will develop a chat application using socket programming. We will explore the basic concepts of networking, multithreading, and the practical use of these concepts in developing a chat application.

Understanding the Basic Concept of Networking : -

Networking is the process of connecting different devices together to share resources and information. The internet is the largest network in the world, and it is made up of millions of devices that are connected together. In order to understand networking, it is important to understand the different layers that make up the internet protocol stack. The internet protocol stack is a conceptual model that describes the different layers of communication that take place between devices connected to the internet.

VARIOUS LAYER OF INTERNET PROTOCOL STACK

The internet protocol stack is made up of the following layers:

1. **Physical layer:** This layer defines the physical characteristics of the communication medium used to transmit data, such as cables, wireless signals, or optical fibers.

2. **Data link layer:** This layer is responsible for transmitting data packets over a communication medium, and it ensures that data is transmitted reliably and without errors.

3. **Network layer:** This layer is responsible for routing data packets between different devices on a network.

4. **Transport layer:** This layer is responsible for ensuring that data is transmitted reliably between devices, and it provides services such as error detection and correction, flow control, and congestion control.

5. **Application layer:** This layer provides services that allow applications to communicate with each other over a network. Examples of application layer protocols include HTTP, FTP, and SMTP.

MULTITHREADING

Multithreading is a programming technique that allows a program to perform multiple tasks simultaneously. In a multithreaded program, each task is executed in a separate thread of execution. This allows the program to be more efficient and responsive, as multiple tasks can be performed concurrently. In the context of a chat application, multithreading is essential as it allows multiple clients to connect to the server and exchange messages simultaneously.

Use of Networking and Multithreading: -

In order to develop a chat application using socket programming, we will need to use both networking and multithreading. Socket programming is a technique that allows applications to communicate with each other over a network using sockets. A socket is a software endpoint that allows two applications to communicate with each other over a network.

In our chat application, we will have two types of users: the client and the server. The client is the user who wants to connect to the chat application and exchange messages with other users. The server is the central component of the chat application, and it is responsible for managing the communication between clients.

The chat application will work as follows:

1. The server will start listening for incoming connections from clients.

2. When a client connects to the server, the server will create a new thread to handle the communication with the client.

3. The client will send a message to the server indicating that it wants to join the chat.

4. The server will add the client to the list of connected clients and send a

message to all other clients indicating that a new user has joined the chat.

5. Clients can now exchange messages with each other.

6. When a client disconnects from the chat, the server will remove the client from the list of connected clients and send a message to all other clients indicating that a user has left the chat.

STREAM

Streams are a way of transferring data between two points in a network. In socket programming, a stream is a sequence of bytes that is transmitted between a client and a server. A stream can be thought of as a channel through which data flows between two points.

In socket programming, there are two types of streams: input streams and output streams. An input stream is used to receive data from a client, and an output stream is used to send data to a client.

The Java programming language provides classes for working with streams. The java.io package contains classes for working with input and output streams. The following are some of the classes provided by the java.io package:

1. **InputStream -** This class is used to read data from a stream.

2. **OutputStream -** This class is used to write data to a stream.

3. **DataInputStream -** This class is used to read primitive data types from a stream.

4. **DataOutputStream -** This class is used to write primitive data types to a stream.

5. **ObjectInputStream -** This class is used to read objects from a stream.

6. **ObjectOutputStream -** This class is used to write objects to a stream.

SOCKET PROGRAMMING

In socket programming, input and output streams are used to send and receive data between a client and a server. When a client sends a message to the server, the message is sent as a sequence of bytes through an output stream. The server then receives the message as a sequence of bytes through an input stream.
Similarly, when the server sends a message to a client, the message is sent as a sequence of bytes through an output stream. The client then receives the message as a sequence of bytes through an input stream.

PURPOSE OF SOCKET PROGRAMMING : -

The purpose of this project is to develop a console-based chat application using socket programming in Java. The application will allow multiple clients to connect to a server and send messages to each other in real-time. This project will cover the basic concepts of networking, multithreading, and the practical use of these concepts in developing a chat application.

VARIOUS USED CONCEPTS IN PROJECT

Object-Oriented Programming (OOPs) : - OOPs is a programming paradigm based on the concept of "objects", which can contain data and code. The four main concepts of OOPs are:

1. **Encapsulation -** This is the process of hiding the internal details of an object from the outside world. In Java, encapsulation is achieved by using private access modifiers.

2. **Inheritance -** This is the process of creating a new class from an existing class. The new class inherits the properties and methods of the existing class. In Java, inheritance is achieved by using the "extends" keyword.

3. **Polymorphism -** This is the ability of an object to take on many forms. In Java, polymorphism is achieved by using method overriding and method overloading.

4. **Abstraction -** This is the process of hiding the implementation details of an object from the outside world. In Java, abstraction is achieved by using abstract classes and interfaces.

Networking: - Networking is the process of connecting two or more devices together to share resources and information. In Java, networking is achieved by using the java.net package. The java.net package provides classes for working with network sockets, URLs, and other network-related tasks.

Multithreading: - Multithreading is the ability of a program to perform multiple tasks at the same time. In Java, multithreading is achieved by creating threads. A thread is a separate execution path within a program.

APPROACH OF PROJECT

The chat application will consist of a client and a server. The client will be able to connect to the server and send messages to other clients. The server will be responsible for handling the connections between clients and forwarding messages**.**

The approach for developing the chat application will be as follows: -

1. Create a Server class that listens for incoming connections from clients.
2. When a client connects to the server, create a new thread to handle the connection.
3. Create a Client class that connects to the server and sends messages.
4. When a message is received by the server, forward the message to all connected clients.
5. Implement error handling for when clients disconnect from the server or when there are connection issues.

Console-Based Application : - The chat application will be a console-based application, meaning that it will run in the command line interface. The user will be able to interact with the application by typing commands and messages into the console.

CODING OF PROJECT

CODING OF THIS PROJECT IS DONE IN TWO CLASSES. THESE CLASSES ARE AS FOLLOWS : -

1. SERVER CLASS
2. CLIENT CLASS

SERVER CLASS

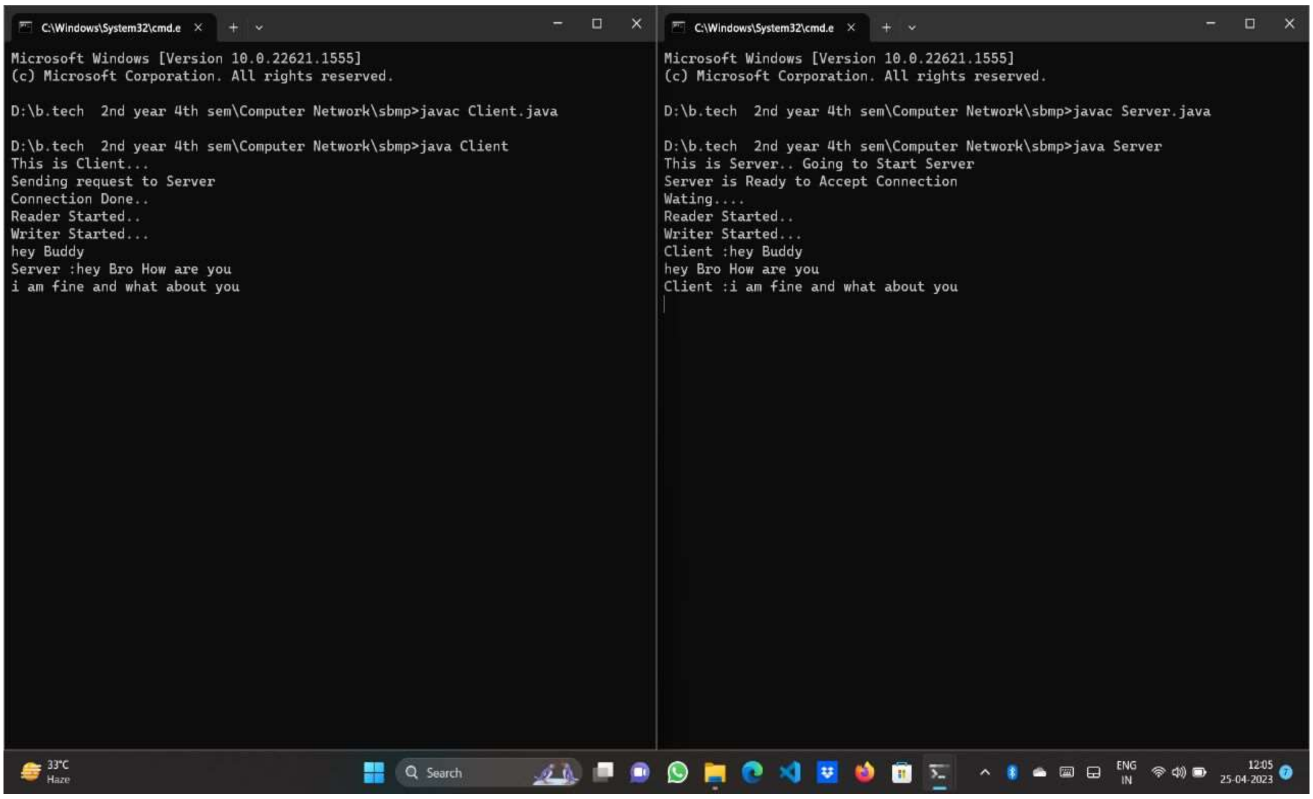
```
import java.net.*;
import java.io.*;
class Server
{
   ServerSocket server;
   Socket socket;
   BufferedReader br;
   PrintWriter out;



   //constructor..
   public Server()
   {
      try {
         server = new ServerSocket(7777);
         System.out.println("Server is Ready to Accept Connection");
         System.out.println("Wating....");
         socket = server.accept();

         br = new BufferedReader(new InputStreamReader (socket.getInputStream()));

         out = new PrintWriter(socket.getOutputStream());

         startReading();
         startWriting();
      }
      catch (Exception e) {
        e.printStackTrace();
    }

}

public void startReading()
{
    //thread - read karke data rahega
    Runnable r1=()->{

        System.out.println("Reader Started..");

        try{
        while(true)
        {

            String msg = br.readLine();
            if(msg.equals("Exit"))
            {
              System.out.println("Client Terminated the Chat");

              socket.close();
              break;
            }

            System.out.println("Client :"+msg);

        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    };

    new Thread(r1).start();
}

public void startWriting()
```

```
    {
        //thread - data user lega and the send karega client tak
        Runnable r2=()->
        {
            System.out.println("Writer Started...");

            try{
             while(true)
             {

                    BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));
                    String content = br1.readLine();
                    out.println(content);
                    out.flush();
             }
           }
            catch(Exception e)
           {
              e.printStackTrace();
            }

        };

        new Thread(r2).start();
    }

    public static void main(String[] args)
    {
        System.out.println("This is Server.. Going to Start Server");
        new Server();
    }
}
```

CLIENT CLASS

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

public class Client {
    Socket socket;
    BufferedReader br;
    PrintWriter out;

    public Client() {
        try {
            System.out.println("Sending request to Server");
            socket = new Socket("192.168.29.136", 7777);
            System.out.println("Connection Done..");

            br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            out = new PrintWriter(socket.getOutputStream());

            startReading();
            startWriting();
        }

        catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void startReading() {
        // thread - read karke data rahega
        Runnable r1 = () -> {
            System.out.println("Reader Started..");

            try {
                while (true) {

                    String msg = br.readLine();
                    if (msg.equals("Exit")) {
                        System.out.println("Server Terminated the Chat");
                        break;
                    }

                    System.out.println("Server :" + msg);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        };

        new Thread(r1).start();
    }

    public void startWriting() {
        // thread - data user lega and the send karega client tak
        Runnable r2 = () -> {
            System.out.println("Writer Started...");
            try {
                while (true) {

                    BufferedReader br1 = new BufferedReader(new
InputStreamReader(System.in));
                    String content = br1.readLine();
                    out.println(content);
                    out.flush();
                }
            }

            catch (Exception e) {
                e.printStackTrace();
            }

        };
        new Thread(r2).start();
    }

    public static void main(String[] args) {
        System.out.println("This is Client...");
        new Client();
    }
}
```

IMAGES OF PROJECT



```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

D:\b.tech  2nd year 4th sem\Computer Network\sbmp>javac Client.java

D:\b.tech  2nd year 4th sem\Computer Network\sbmp>java Client
This is Client...
Sending request to Server
Connection Done..
Reader Started..
Writer Started...
hey Buddy
Server :hey Bro How are you
i am fine and what about you
```

```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

D:\b.tech  2nd year 4th sem\Computer Network\sbmp>javac Server.java

D:\b.tech  2nd year 4th sem\Computer Network\sbmp>java Server
This is Server.. Going to Start Server
Server is Ready to Accept Connection
Wating....
Reader Started..
Writer Started...
Client :hey Buddy
hey Bro How are you
Client :i am fine and what about you
```

CONCLUSION

In conclusion, this project has covered the basic concepts of networking, multithreading, and the practical use of these concepts in developing a chat application using socket programming in Java. The approach for developing the chat application was to create a console-based application that allows multiple clients to connect to a server and send messages to each other in real-time.

Socket programming is a powerful technique for developing network applications. The ability to communicate with other devices over a network has made it possible to develop applications that can be used by millions of people around the world. In this project We have also discussed the use of streams in socket programming and how they are used to transfer data between a client and a server. Overall, this project has provided a good introduction to socket programming and the various techniques used in developing network applications.

REFRENCES

WWW.YOUTUBE.COM
WWW.VSCODE.COM
WWW.WIKIPEDIA.COM
WWW.W3SCHOOL.COM
WWW.GEEKSFORGEEKS.COM