MATURI VENKATA SUBBA RAO ENGINEERING COLLEGE

(An Autonomous Institution)

(Affiliated to Osmania University & Recognized by AICTE) Nadergul, RangaReddyDist.

CERTIFICATE

Department of COMPUTER SCIENCE &ENGINEERING

This is to certify that it is a bonafide laboratory work of experiments carried out by Mr./Miss. _____bearing R.No. _____ under the laboratory course **Distributed Systems Lab** prescribed for B.E **IV/IV** Sem-7 by Osmania University in the department during the academic year **2022-2023**.

Internal Examiner ***External Examiner***

VISION AND MISSION

VISION

- To impart technical education of the highest standards, producing competent and confident engineers with an ability to use computer science knowledge to solve societalproblems.

MISSION

- To make learning process exciting, stimulating andinteresting.
- To impart adequate fundamental knowledge and soft skills tostudents.
- To expose students to advanced computer technologies in order to excel in engineering practices by bringing out the creativity instudents.
- To develop economically feasible and socially acceptablesoftware.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The Bachelor's program in Computer Science and Engineering is aimed at preparing graduates who will:-
PEO-1: Achieverecognitionthroughdemonstrationoftechnicalcompetenceforsuccessfulexecutionofsoftware projects to meet customer businessobjectives.
PEO-2: Practice life-long learning by pursuing professional certifications, higher education or research in the emerging areas of information processing and intelligent systems at a global level.
PEO-3: Contribute to society by understanding the impact of computing using a multidisciplinary and ethical approach.

PROGRAM OUTCOMES(POs)

At the end of the program the students (Engineering Graduates) will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineeringproblems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmentalconsiderations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environmentandsustainability:**Understandtheimpactoftheprofessionalengineeringsolutionsinsocietaland environmental contexts, and demonstrate the knowledge of, and need for sustainabledevelopment.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineeringpractice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community
    andwithsocietyatlarge,suchas,beingabletocomprehendandwriteeffectivereportsanddesigndocumentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principle and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinaryenvironments.
12. **Lifelong learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technologicalchange.

PROGRAM SPECIFIC OUTCOMES(PSOs)

13. (PSO-1) Demonstrate competence to build effective solutions for computational real-world problems using software and hardware across multi-disciplinarydomains.
14. (PSO-2) Adapt to current computing trends for meeting the industrial and societal needs througha holistic professional development leading to pioneering careers orentrepreneursh.

Course Name: Distributed Systems Lab

Course Code: PC 752CS

Academic Year: 2022-2023

Semester:VII

Course Objectives

- To implement client and server programs using sockets
- To learn about working of NFS
- Understanding Remote Communication and Interprocess Communication
- To use Map, reduce model for distributed processing
- To develop mobile applications

Course Outcomes

After completing this course, the student will be able to

- Write programs that communicate data between two hosts
- Configure NFS
- To implement inter process communication and remote communication
- Use distributed data processing frameworks and mobile application tool kits

INDEX

S.No.	Name of the Experiment	Experiment Date	Date of Submission	Page No.

1.Chat Server Implementation

Introduction:

Chat server is a standalone application that is made of combination of two-applications, server application (which runs on server side) and client application (which runs on client side). This application is using for chatting in LAN. To start chatting client must be connected with the server after which, messages can be broadcast between each and every client.

SERVER:

Server side application is used to get the message from any client and broadcast to each and every client. And this application is also used to maintain the list of users and broadcast this list to everyone. Firstly server program creates a new server socket by the **ServerSocket ss = new ServerSocket(Port number);**

- After creating the ServerSocket it accepts the client socket and adds this socket into an arraylist.
- After getting the client socket it creates a thread and enables the DataInputStream for this socket.
- After creating the input stream it reads the user name and adds it to arraylist and this arraylist object writes into the ObjectOutputStream of each client by using an iterator.
- After this process, it creates a new thread and creates one DataInputStream for reading the messages which is sent by the client and after reading the message it creates the DataOutputStream for each socket and writes this message in each client output stream through the iterator. If any client logs out,the server receives the client name and removes it from the arraylist. Further ,it sends this updated arraylist to all clients.

CLIENT:

In the client side, first the program creates a new socket and specifies the address and port of the server and establishes the connection with the Server.The client, then, creates a new thread and DataInputStream, ObjectInputStream and DataOutputStream for sending the user name and retrieving the list of all users and adds all the user names into its list box through the iterator.Then new thread is created for sending and receiving the messages from the server. This task is done by using DataInputStream and DataOutputStream.When the client logs out it sends its' name and message i.e. "User_Name has Logged out" and terminates the chatting.

ChatServer.java

```
import java.net.*;
import java.io.*;
public class ChatServer implements Runnable {
   private ChatServerThread clients[] = new ChatServerThread[50];
   private ServerSocket server = null;
   private Thread thread = null;
   private int clientCount = 0;
   public ChatServer(int port) {
      try {
         System.out.println("binding to port" + port + ",please wait ...");
         server = new ServerSocket(port);
         System.out.println("server started:" + server);
         thread = new Thread(this);
         thread.start();
      } catch (IOException e) {
         System.out.println("cannot bind to port" + port + ":" + e.getMessage());
      }
   }
   public void run() {
      while (thread != null) {
         try {
            System.out.println("waiting for a client...");
            addThread(server.accept());
         } catch (IOException e) {
            System.out.println("server accept error:" + e);
            if (thread != null) {
               thread.stop();
               thread = null;
            }
         }
      }
   }
   public void stop() {
      if (thread != null) {
         thread.stop();
         thread = null;
      }
   }
   private int findClient(int ID) {
      for (int i = 0; i < clientCount; i++)
         if (clients[i].getID() == ID)
            return i;
      return -1;
   }
   public synchronized void handle(int ID, String input) {
      if (input.equals("quit")) {
         clients[findClient(ID)].send("quit");
```

```
            remove(ID);
        } else
            System.out.println(ID + ":" + input);
        for (int i = 0; i < clientCount; i++)
            clients[i].send(ID + ":" + input);
    }
    public synchronized void remove(int ID) {
        int pos = findClient(ID);
        if (pos >= 0) {
            ChatServerThread closing = clients[pos];
            System.out.println("removing client thread:" + ID + "at" + pos);
            if (pos < clientCount - 1)
                for (int i = pos + 1; i < clientCount; i++)
                    clients[i - 1] = clients[i];
            clientCount--;
            try {
                closing.close();
            } catch (IOException e) {
                System.out.println("error closing thread;" + e);
            }
            closing.stop();
        }
    }
    private void addThread(Socket socket) {
        if (clientCount < clients.length) {
            System.out.println("client accepted:" + socket);
            clients[clientCount] = new ChatServerThread(this, socket);
            try {
                clients[clientCount].open();
                clients[clientCount].start();
                clientCount++;
            } catch (IOException e) {
                System.out.println("error opening thread;" + e);
            }
        } else
            System.out.println("client refused;maximum" + clients.length + "reached");
    }
    public static void main(String a[]) {
        ChatServer server = null;
        if (a.length != 1)
            System.out.println("Usage:javaChatServer Port");
        else
            server = new ChatServer(Integer.parseInt(a[0]));
    }
}
```

ChatServerThread.java

```
import java.net.*;
import java.io.*;
public class ChatServerThread extends Thread {
   private ChatServer server = null;
   private Socket socket = null;
   private DataInputStream In = null;
   private int ID = -1;
   private PrintStream Out = null;
   public ChatServerThread(ChatServer serv, Socket sock) {
      super();
      server = serv;
      socket = sock;
      ID = socket.getPort();
   }
   public void send(String msg) {
      Out.println(msg);
      Out.flush();
   }
   public int getID() {
      return ID;
   }
   public void run() {
      System.out.println("Server thread" + ID + "running");
      while (true) {
         try {
            server.handle(ID, In.readLine());
         } catch (IOException e) {
            System.out.println(ID + "error reading" + e.getMessage());
            server.remove(ID);
            stop();
         }
      }
   }
   public void open() throws IOException {
      In = new DataInputStream(socket.getInputStream());
      Out = new PrintStream(socket.getOutputStream());
   }
   public void close() throws IOException {
      if (socket != null)
         socket.close();
      if (In != null)
         In.close();
      if (Out != null)
         Out.close();
   }
}
```

ChatClient.java

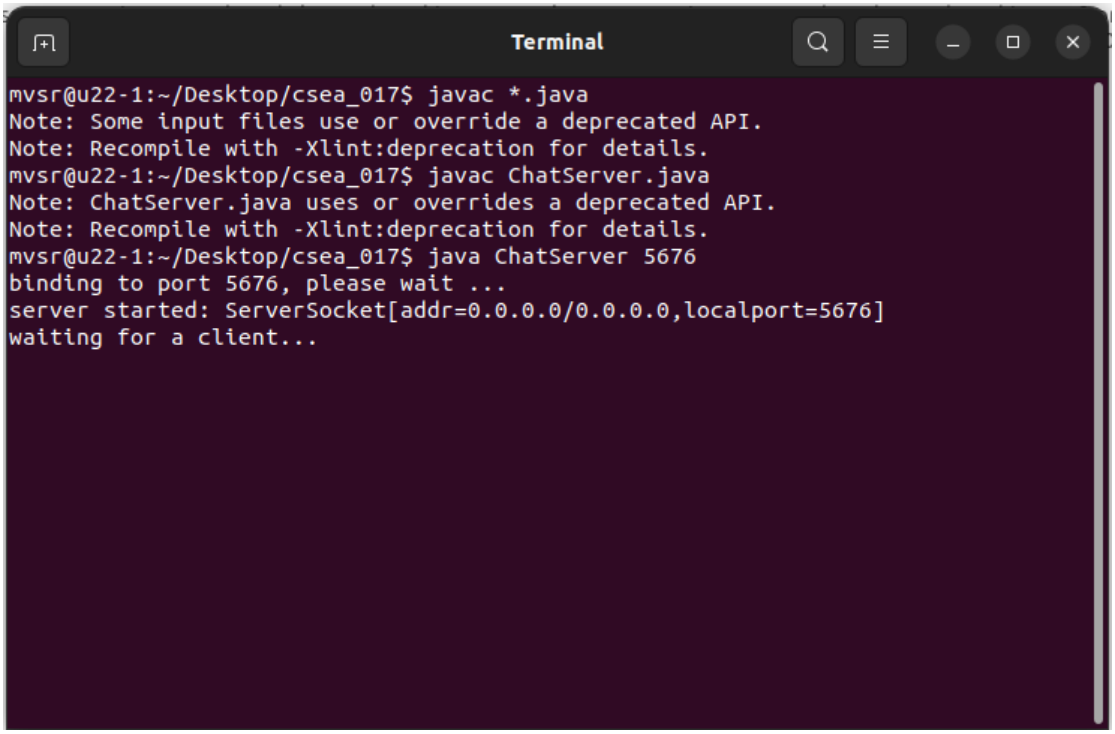
```
import java.net.*;
import java.io.*;
public class ChatClient implements Runnable {
    private ChatClientThread client = null;
    private Socket socket = null;
    private DataInputStream console = null;
    private Thread thread = null;
    private PrintStream Out = null;
    public ChatClient(String serverName, int serverPort) {
        System.out.println("Establishing connection please wait...");
        try {
            socket = new Socket(serverName, serverPort);
            System.out.println("connected" + socket.toString());
            console = new DataInputStream(System.in);
            Out = new PrintStream(socket.getOutputStream());
            if (thread == null) {
                client = new ChatClientThread(this, socket);
                thread = new Thread(this);
                thread.start();
            }
        } catch (UnknownHostException e) {
            System.out.println("Host unknown" + e.getMessage());
        } catch (IOException ioe) {
            System.out.println("Unexcepted exception" + ioe.getMessage());
        }
    }
    public void run() {
        while (thread != null) {
            try {
                Out.println(console.readLine());
                Out.flush();
            } catch (IOException e) {
                System.out.println("sending error" + e.getMessage());
                stop();
            }
        }
    }
    public void handle(String msg) {
        if (msg.equals("quit")) {
            System.out.println("good bye, press RETURN to exit...");
            stop();
        } else
            System.out.println(msg);
    }
    public void stop() {
        if (thread != null) {
            thread.stop();
```

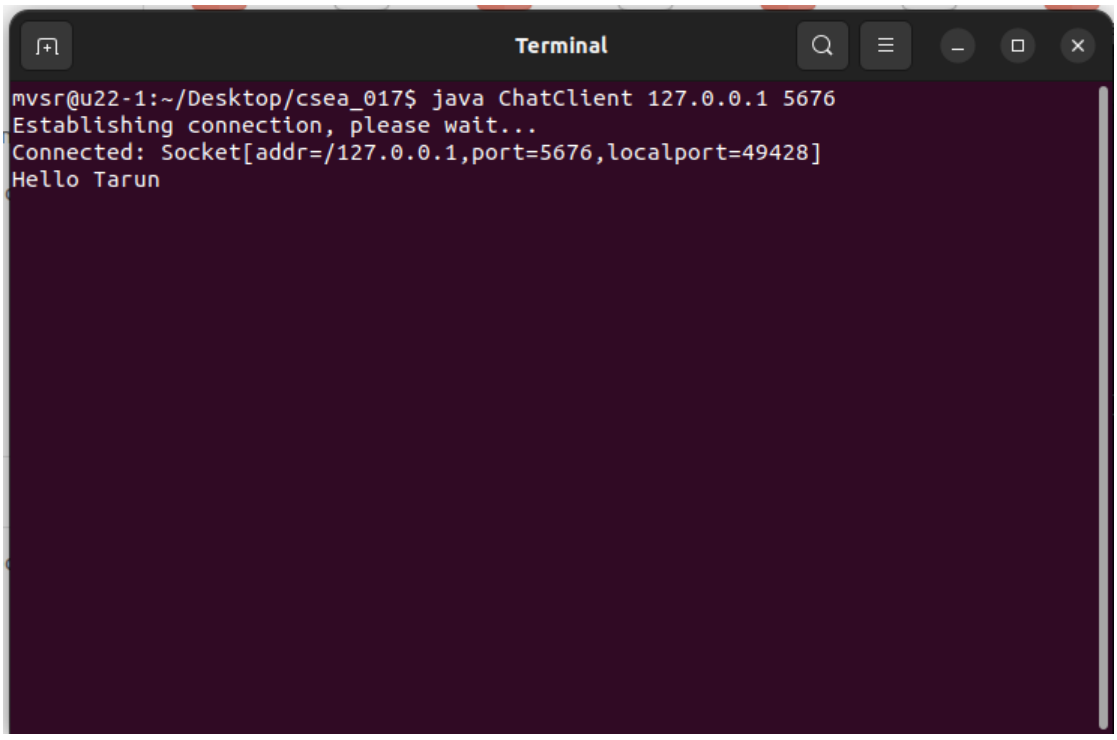
```
            thread = null;
        }
        try {
            if (console != null)
                console.close();
            if (Out != null)
                Out.close();
            if (socket != null)
                socket.close();
        } catch (IOException e) {
            System.out.println("error closing...");
        }
        client.close();
        client.stop();
    }
    public static void main(String args[]) {
        ChatClient client = null;
        if (args.length != 2)
            System.out.println("usage:javaChatClient host port");
        else
            client = new ChatClient(args[0], Integer.parseInt(args[1]));
    }
}
```

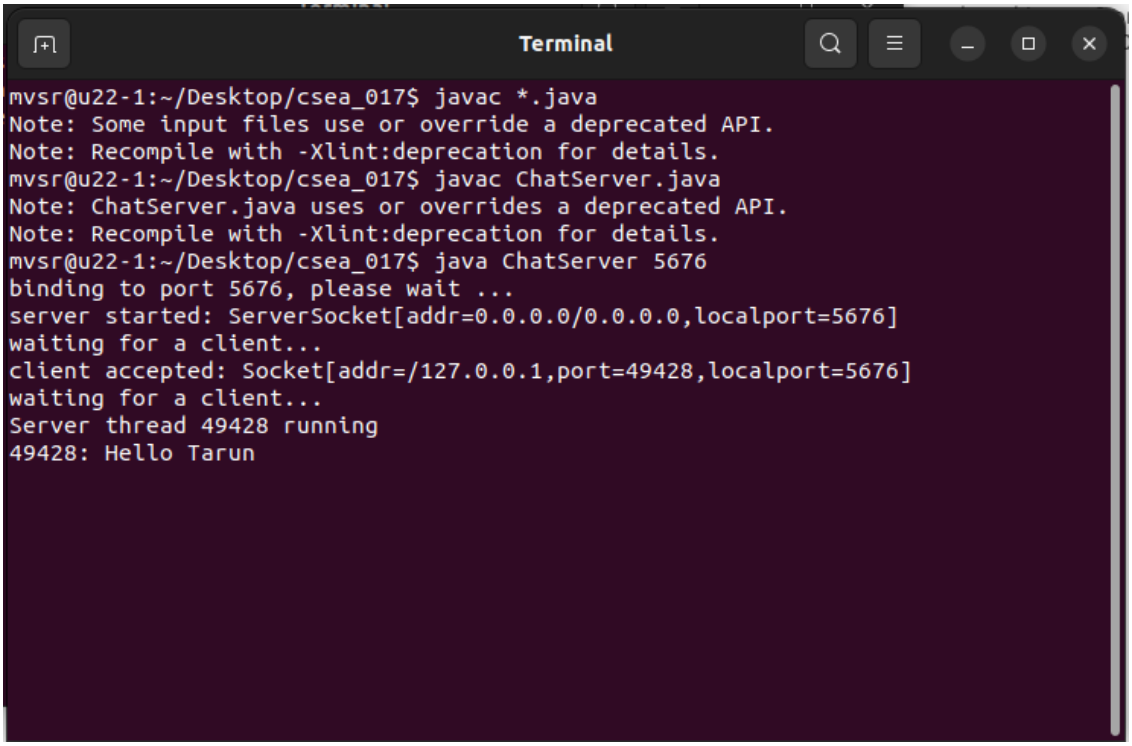
ChatClientThread.java

```
import java.net.*;
import java.io.*;
public class ChatClientThread extends Thread {
        private ChatClient client=null;
        private Socket socket=null;
        private DataInputStream In=null;
        public ChatClientThread(ChatClient cli, Socket sock){
            client=cli;
            socket=sock;
            try{
                In=new DataInputStream(socket.getInputStream());
            }
            catch (IOException e){
                System.out.println("error geting input stream:"+e);
                client.stop();
            }
            start();
        }

        public void close(){
            try{
                if(In!=null)
                In.close();
            }
            catch (IOException e){
                System.out.println("error closing input stream:"+e);
            }
        }

        public void run(){
            while(true){
                try{
                    client.handle(In.readLine());
                }
                catch(IOException e){
                    System.out.println("Listening error"+e.getMessage());
                    client.stop();
                }
            }
        }
}
```

Sample Outputs:






Terminal
mvsr@u22-1:~/Desktop/csea_017$ javac *.java
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
mvsr@u22-1:~/Desktop/csea_017$ javac ChatServer.java
Note: ChatServer.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
mvsr@u22-1:~/Desktop/csea_017$ java ChatServer 5676
binding to port 5676, please wait ...
server started: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=5676]
waiting for a client...
client accepted: Socket[addr=/127.0.0.1,port=49428,localport=5676]
waiting for a client...
Server thread 49428 running
49428: Hello Tarun

2. FTP Implementation

```
import java.net.*;
import java.io.*;
import java.util.*;
public class ftp {
   static Socket DataSocket;
   static DataInputStream ipstream;
   static PrintStream outstream;
   public static void main(String args[]) throws IOException {
      try {
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         System.out.println("Enter the FTP host address:");
         String host = br.readLine();
         DataSocket = new Socket(host, 21);
         ipstream = new DataInputStream(DataSocket.getInputStream());
         outstream = new PrintStream(DataSocket.getOutputStream());
         String s = ipstream.readLine();
         if (s.startsWith("220")) {
            System.out.println("Connected to server " + host);
            System.out.println("\n Enter Username:\t");
            String uname = br.readLine();
            outstream.print("USER" + uname + "\r\n");
            s = ipstream.readLine();
            if (s.startsWith("331")) {
               System.out.println("\n Enter password:\t");
               String pass = br.readLine();
               outstream.print("PASS" + pass + "\r\n");
               s = ipstream.readLine();
               if (s.startsWith("230")) {
                  System.out.println("Login successful!!!HAVE A NICE DAY!!!\n");
                  while (true) {
                     System.out.println("Press Enter to continue . . . ");
                     String name = br.readLine();
                     System.out.println("Enter your choice: \n");
                     System.out.println("1. Read a file \n");
                     System.out.println("2. Store a file \n");
                     System.out.println("3. List files \n");
                     System.out.println("4. Change directory \n");
                     System.out.println("5. Change to Parent Directory \n");
                     System.out.println("6. Create Directory \n");
                     System.out.println("7. Print Current Directory \n");
                     System.out.println("8. Logout \n");
                     int option = Integer.parseInt(br.readLine());
                     switch (option) {
                        case 1:
                           read();
```

```
                    break;
                case 2:
                    store();
                    break;
                case 3:
                    list();
                    break;
                case 4:
                    System.out.println("Enter Directory: \n");
                    name = br.readLine();
                    outstream.print("CWD " + name + "\r\n");
                    break;
                case 5:
                    outstream.print("CDUP" + "\r\n");
                    break;
                case 6:
                    System.out.println("Enter Directory: \n");
                    name = br.readLine();
                    outstream.print("MKD " + name + "\r\n");
                    break;
                case 7:
                    outstream.print("PWD" + "\r\n");
                    break;
                case 8:
                    outstream.print("QUIT" + "\r\n");
                    System.exit(1);
                    break;
                default:
                    System.out.println("Choose a valid option!!!\n");
                    break;
                }
            }
        }
      }
    } else {
        System.out.println("Error connecting to server" + host);
    }
  } catch (UnknownHostException e) {
    System.err.println(e);
  } catch (IOException e) {
    System.err.println(e);
  }
}
static void store() {
  try {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

```
        System.out.println("Enter Filename: \n");
        String filename = br.readLine();
        FileInputStream inputStream = new FileInputStream(filename);
        outstream.print("PASV" + "\r\n");
        String s = ipstream.readLine();
        String ip = null;
        int port = -1;
        int opening = s.indexOf('(');
        int closing = s.indexOf(')', opening + 1);
        if (closing > 0) {
           String dataLink = s.substring(opening + 1, closing);
           StringTokenizer tokenizer = new StringTokenizer(dataLink, ",");
           ip = tokenizer.nextToken() + "." + tokenizer.nextToken() + "." + tokenizer.nextToken() +
              "." + tokenizer.nextToken();
           port = Integer.parseInt(tokenizer.nextToken());
           outstream.print("STOR" + filename + "\r\n");
           Socket dataSocket = new Socket(ip, port);
           s = ipstream.readLine();
           if (!s.startsWith("150")) {
              throw new IOException("Simple FTP was not allowed to send the file:" + s);
           }
           BufferedInputStream input = new BufferedInputStream(inputStream);
           BufferedOutputStream output = new
           BufferedOutputStream(DataSocket.getOutputStream());
           byte[] buffer = new byte[4096];
           int bytesRead = 0;
           while ((bytesRead = input.read(buffer)) != -1) {
              output.write(buffer, 0, bytesRead);
           }
           output.flush();
           output.close();
           input.close();
        }
     } catch (UnknownHostException e) {
        System.err.println(e);
     } catch (IOException e) {
        System.err.println(e);
     }
  }
  static void read() {
     try {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter Filename \n");
        String filename = br.readLine();
        outstream.print("PASV" + "\r\n");
        String s = ipstream.readLine();
```

```
            System.out.println(s);
            String ip = null;
            int port = -1;
            int opening = s.indexOf('(');
            int closing = s.indexOf(')', opening + 1);
            if (closing > 0) {
               String dataLink = s.substring(opening + 1, closing);
               StringTokenizer tokenizer = new StringTokenizer(dataLink, ",");
               ip = tokenizer.nextToken() + "." + tokenizer.nextToken() + "." + tokenizer.nextToken() +
                  "." + tokenizer.nextToken();
               port = Integer.parseInt(tokenizer.nextToken()) * 256 +
                  Integer.parseInt(tokenizer.nextToken());
               outstream.print("RETR" + filename + "\r\n");
               Socket dataSocket = new Socket(ip, port);
               s = ipstream.readLine();
               if (!s.startsWith("150")) {
                  throw new IOException("Simple FTP was not allowed to send the file:" + s);
               }
               BufferedInputStream input = new BufferedInputStream(dataSocket.getInputStream());
               FileOutputStream output = new FileOutputStream("New" + filename);
               byte[] buffer = new byte[4096];
               int bytesRead = 0;
               while ((bytesRead = input.read(buffer)) != -1) {
                  output.write(buffer, 0, bytesRead);
               }
               s = ipstream.readLine();
               System.out.println("Finished!" + s);
               output.close();
               input.close();
            }
         } catch (UnknownHostException e) {
            System.err.println(e);
         } catch (IOException e) {
            System.err.println(e);
         }
      }
      static void list() {
         try {
            outstream.print("PASV" + "\r\n");
            String s = ipstream.readLine();
            System.out.println(s);
            String ip = null;
            int port = -1;
            int opening = s.indexOf('(');
            int closing = s.indexOf(')', opening + 1);
            if (closing > 0) {
```

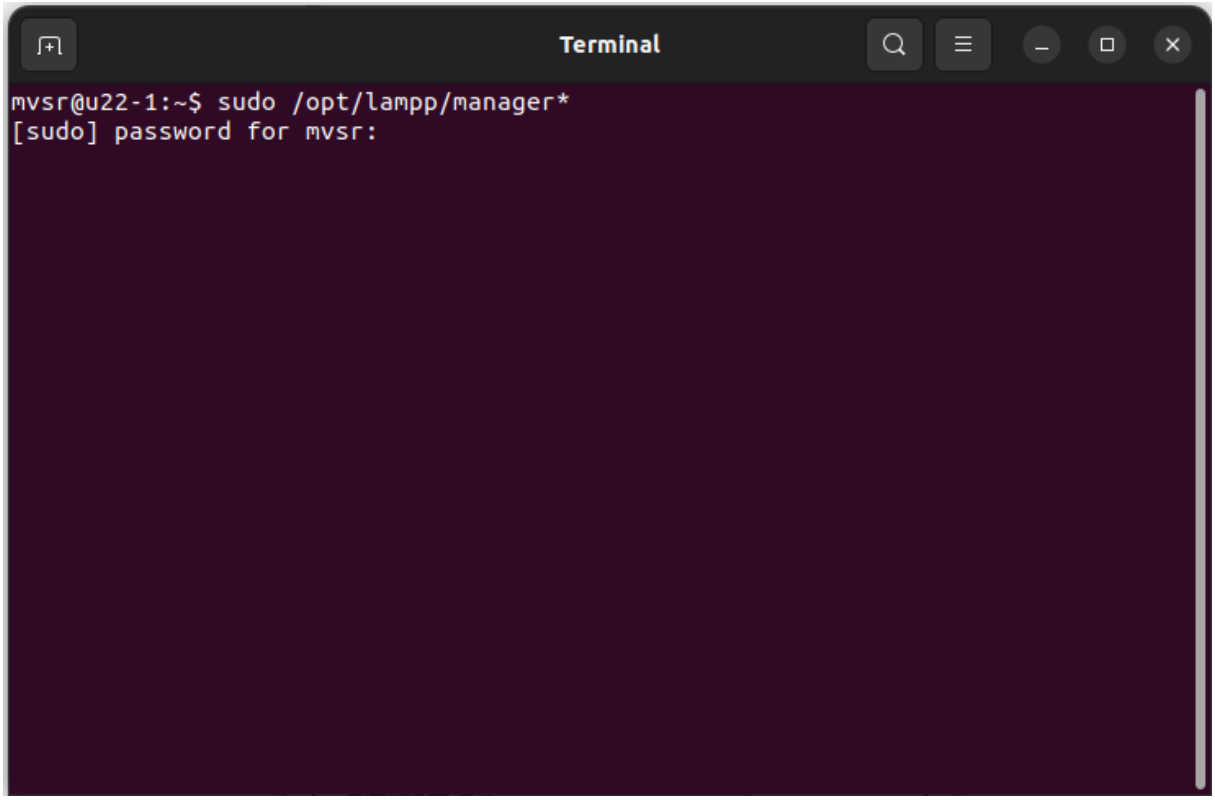
```
            String dataLink = s.substring(opening + 1, closing);
            StringTokenizer tokenizer = new StringTokenizer(dataLink, ",");
            ip = tokenizer.nextToken() + "." + tokenizer.nextToken() + "." + tokenizer.nextToken() +
               "." + tokenizer.nextToken();
            port = Integer.parseInt(tokenizer.nextToken()) * 256 +
               Integer.parseInt(tokenizer.nextToken());
            outstream.print("LIST" + "\r\n");
            Socket dataSocket = new Socket(ip, port);
            s = ipstream.readLine();
            DataInputStream input = new DataInputStream(dataSocket.getInputStream());
            String line;
            while ((line = input.readLine()) != null)
               System.out.println(line);
            input.close();
         }
      } catch (UnknownHostException e) {
         System.err.println(e);
      } catch (IOException e) {
         System.err.println(e);
      }
   }
}
```

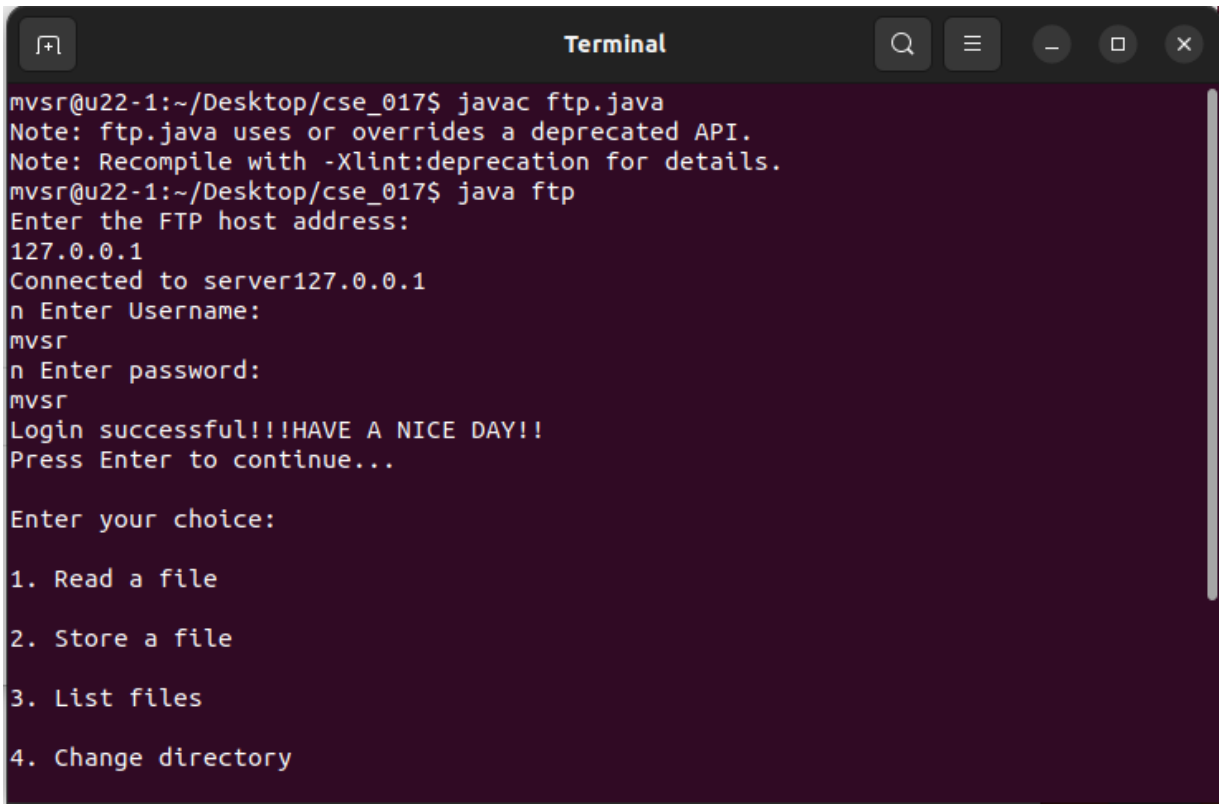
Steps of Execution:

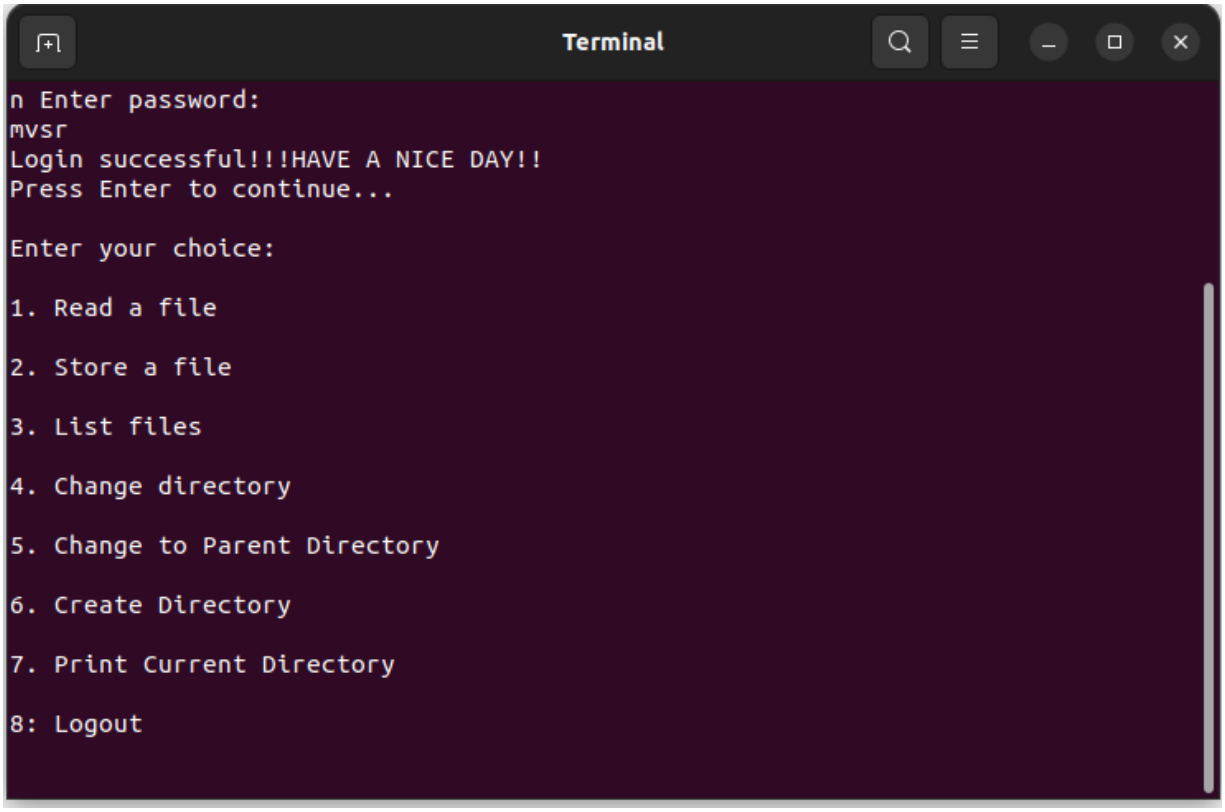
1. After compiling the java program open a new terminal and type the commands
   - sudo /opt/lampp/manager*
   → Password :mvsr
2. It connects to the lampp.
3. Select Manage servers and ProFTPD
4. Click start.
5. Now run the java program and enter FTP host address as loop back address i.e. 127.0.0.1

And rest of the steps as mentioned in the below picture.

OUTPUT:


Terminal
mvsr@u22-1:~$ sudo /opt/lampp/manager*
[sudo] password for mvsr:


Terminal
mvsr@u22-1:~/Desktop/cse_017$ javac ftp.java
Note: ftp.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
mvsr@u22-1:~/Desktop/cse_017$ java ftp
Enter the FTP host address:
127.0.0.1
Connected to server127.0.0.1
n Enter Username:
mvsr
n Enter password:
mvsr
Login successful!!!HAVE A NICE DAY!!
Press Enter to continue...
Enter your choice:
1. Read a file
2. Store a file
3. List files
4. Change directory


Terminal
n Enter password:
mvsr
Login successful!!!HAVE A NICE DAY!!
Press Enter to continue...
Enter your choice:
1. Read a file
2. Store a file
3. List files
4. Change directory
5. Change to Parent Directory
6. Create Directory
7. Print Current Directory
8: Logout

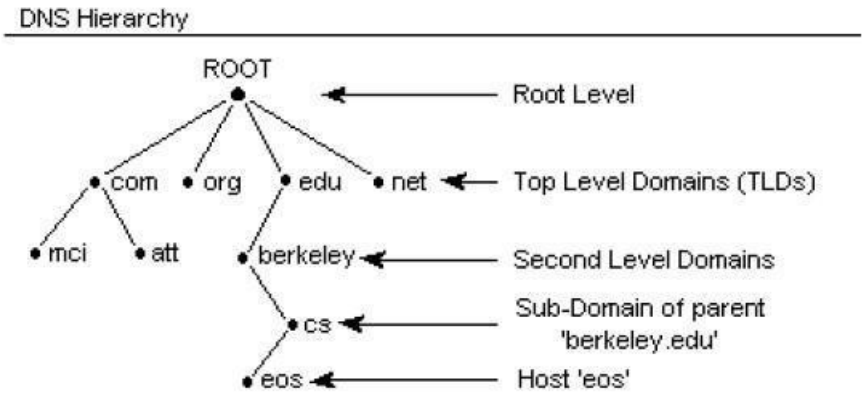
3. Domain Name Server Implementation Using Database

Introduction:

The **Domain Name System** (**DNS**) is a hierarchical naming system built on a distributed database for computers, services, or any resource connected to the Internet or a private network. The Domain Name System makes it possible to assign domain names to groups of Internet resources and users in a meaningful way, independent of each entity's physical location. Because of this, World Wide Web (WWW) hyperlinks and Internet contact information can remain consistent and constant even if the current Internet routing arrangements change.The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating iterative name servers for each domain.The Domain Name System is maintained by a distributed database system, which uses the clientserver model. The nodes of this database are the name servers. Each domain has at least one authoritative DNS server that publishes information about that domain and the name servers of any domains subordinate to it. The top of the hierarchy is served by the root name servers, the servers to query when looking up (*resolving*) a top level domain name.

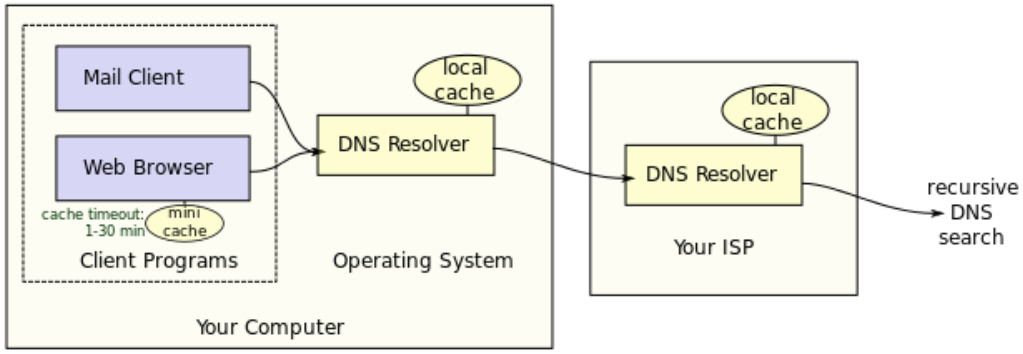
The Domain Name System also specifies the technical functionality of thedatabaseservice which is at its core. It defines the DNS protocol, a detailed specification of the data structures and data communication exchanges used in DNS, as part of theInternet Protocol Suite. Historically, other directory services preceding DNS were not scalable to large or global directories as they were originally based on text files, prominently theHOSTS.TXTresolver. DNS has been in wide use since the 1980s. The most common types of records stored in the DNS database are forDNS zoneauthority (SOA),IP addresses(A and AAAA),SMTPmail exchangers(MX),name servers(NS), pointers forreverse DNS lookups(PTR), anddomain name aliases(CNAME).

Domain name resolvers determine the domain name servers responsible for the domain name in question by a sequence of queries starting with the right-most (top-level) domain label.



DNS resolvers

The client side of the DNS is called a DNS resolver. It is responsible for initiating and sequencing the queries that ultimately lead to a full resolution (translation) of the resource sought, e.g., translation of a domain name into an IP address.



Implementation:

- Design the table with in the database as specified in the next section.
- Implement individual server for each domain with in DNS.
- Each server program is now in waiting state.
- Implement client program which accepts URL from user, divides the URL into different domain and fetch the IPs of the domains from the corresponding servers.
- Combine the IPs received from different server and display them to the user.

Table Name : **Client**

name	ipadd	port
com	10	5123
edu	20	4123
org	30	3123

Table Name: Root

name	ipadd	port
google	13	5125
yahoo	12	5124

Table Name: Yahoo

name	ipadd	port
mail	100	0
program	200	0

Table Name: Google

name	ipadd	port
mail	100	0
program	200	0

Server1.java

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.sql.Connection;
import java.util.*;
class Server1 {
   public static void main(String a[]) {
      ServerSocket sock;
      Socket client;
      DataInputStream input;
      PrintStream ps;
      String url, u, s;
      Connection con;
      Statement smt;
      ResultSet rs;
      try {
         s = u = "\0";
         Class.forName("com.mysql.cj.jdbc.Driver");
         con = DriverManager.getConnection("jdbc:mysql://localhost:3307/dns", "root", "");
         smt = con.createStatement();
         sock = new ServerSocket(5123);
         while (true) {
            client = sock.accept();
            input = new DataInputStream(client.getInputStream());
            ps = new PrintStream(client.getOutputStream());
            url = input.readLine();
            System.out.println("IN SERVER1 URL IS:" + url);
            StringTokenizer st = new StringTokenizer(url, ".");
            while (st.countTokens() > 1)
               s = s + st.nextToken() + ".";
            s = s.substring(0, s.length() - 1).trim();
            u = st.nextToken();
            rs = smt.executeQuery("select port,ipadd from root where name='" + u + "'");
            if (rs.next()) {
               ps.println(Integer.parseInt(rs.getString(1)));
               ps.println(Integer.parseInt(rs.getString(2)));
               ps.println(s);
            } else {
               ps.println("Illegal address pleasr check the spelling again");
               con.close();
            }
         }
      } catch (Exception e) {
         System.err.println(e);
      }
   }
}
```

Server2.java

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.sql.Connection;
import java.util.*;
class Server2 {
   public static void main(String a[]) {
      ServerSocket sock;
      Socket client;
      DataInputStream input;
      PrintStream ps;
      String url, u, s;
      Connection con;
      Statement smt;
      ResultSet rs;
      try {
         s = u = "\0";
         Class.forName("com.mysql.cj.jdbc.Driver");
         con = DriverManager.getConnection("jdbc:mysql://localhost:3307/dns", "root", "");
         smt = con.createStatement();
         sock = new ServerSocket(5124);
         while (true) {
            client = sock.accept();
            input = new DataInputStream(client.getInputStream());
            ps = new PrintStream(client.getOutputStream());
            url = input.readLine();
            System.out.println("IN SERVER2 URL IS:" + url);
            StringTokenizer st = new StringTokenizer(url, ".");
            while (st.countTokens() > 1)
               s = s + st.nextToken() + ".";
            s = s.substring(0, s.length() - 1).trim();
            u = st.nextToken();
            rs = smt.executeQuery("select port,ipadd from yahoo where name='" + u + "'");
            if (rs.next()) {
               ps.println(rs.getString(1));
               ps.println(rs.getString(2));
               ps.println(s);
            } else {
               ps.println("Illegal address pleasr check the spelling again");
               con.close();
            }
         }
      } catch (Exception e) {
         System.err.println(e);
      }
   }
}
```

Server3.java

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.sql.Connection;
import java.util.*;
class Server3 {
   public static void main(String a[]) {
      ServerSocket sock;
      Socket client;
      DataInputStream input;
      PrintStream ps;
      String url, u, s;
      Connection con;
      Statement smt;
      ResultSet rs;
      try {
         s = u = "\0";
         Class.forName("com.mysql.cj.jdbc.Driver");
         con = DriverManager.getConnection("jdbc:mysql://localhost:3307/dns", "root", "");
         smt = con.createStatement();
         sock = new ServerSocket(5125);
         while (true) {
            client = sock.accept();
            input = new DataInputStream(client.getInputStream());
            ps = new PrintStream(client.getOutputStream());
            url = input.readLine();
            System.out.println("IN SERVER3 URL IS:" + url);
            StringTokenizer st = new StringTokenizer(url, ".");
            while (st.countTokens() > 1)
               s = s + st.nextToken() + ".";
            s = s.substring(0, s.length() - 1).trim();
            u = st.nextToken();
            rs = smt.executeQuery("select port,ipadd from google where name='" + u + "'");
            if (rs.next()) {
               ps.println(rs.getString(1));
               ps.println(rs.getString(2));
               ps.println(s);
            } else {
               ps.println("Illegal address pleasr check the spelling again");
               con.close();
            }
         }
      } catch (Exception e) {
         System.err.println(e);
      }
   }
}
```

Client1.java

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.sql.Connection;
import java.util.*;
class Client1 {
   public static void main(String a[]) {
      Socket clisock;
      DataInputStream input;
      PrintStream ps;
      String url, ip, s, u, p, str;
      int pno = 5123;
      Connection con;
      Statement smt;
      ResultSet rs;
      boolean status = true;
      try {
         ip = s = p = u = "\0";
         System.out.println("enter name to resolve");
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         url = br.readLine();
         Class.forName("com.mysql.cj.jdbc.Driver");
         con = DriverManager.getConnection("jdbc:mysql://localhost:3307/dns", "root", "");
         smt = con.createStatement();
         while (status) {
            s = "\0";
            System.out.println("IN CLIENT URL IS:" + url);
            StringTokenizer st = new StringTokenizer(url, ".");
            if (st.countTokens() == 1) {
               status = false;
            }
            while (st.countTokens() > 1)
               s = s + st.nextToken() + ".";
```

```
            s = s.substring(0, s.length() - 1).trim();
            u = st.nextToken();
            System.out.println("u=" + u);
            rs = smt.executeQuery("select port,ipadd from client where name='" + u + "'");
            if (rs.next()) {
               p = rs.getString(1);
               pno = Integer.parseInt(p);
               str = rs.getString(2);
               url = s;
               ip = str + "." + ip;
            } else {
               System.out.println("pno=" + pno);
               clisock = new Socket("127.0.0.1", pno);
               input = new DataInputStream(clisock.getInputStream());
               ps = new PrintStream(clisock.getOutputStream());
               ps.println(url);
               p = input.readLine();
               pno = Integer.parseInt(p);
               str = input.readLine();
               url = input.readLine();
               ip = str + "." + ip;
               smt.executeUpdate("insert into client values('" + u + "','" + str + "','" + p + "')");
            }
            System.out.println("ip=" + ip);
         }
         ip = ip.substring(0, ip.length() - 1).trim();
         System.out.println("ip address is:" + ip);
         con.close();
      } catch (Exception e) {
         System.err.println(e);
      }
   }
}
```

Steps to Download xampp Manager for UBUNTU

1. Download xampp for ubuntu 64 bit
2. After installation go to terminal
   Cd Downloads
3.To give Executable Permission
chmod +x ./xa (press tab) – displays complete file name
4. sudo ./xa (press tab) and enter to install
give mvsr as a pswd

After installation

5.In one of the terminals, type the command **sudo /opt/lampp/manager***

→ Password :mvsr

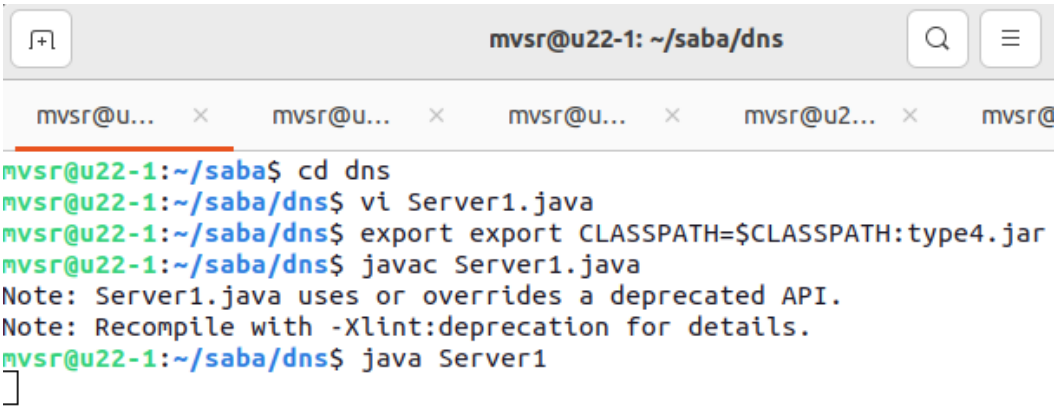
6.Then the xamp server will be opened, click start all.
7. Go to https://locallhost.me/phpmyadmin in web browser and create 4 tables Client, Root,Yahoo,Google as shown in the "implementation" section above.
8.Put your database name as your roll number and make sure to replace **dns** (database name in the above programs) with your roll number.

Download mysqljdbc driver
Go to terminal set the path as rename the mysqljdbc driver name as type 4.jar
1.export export CLASSPATH=$CLASSPATH :type4.jar
javac Server1.java
java Server1

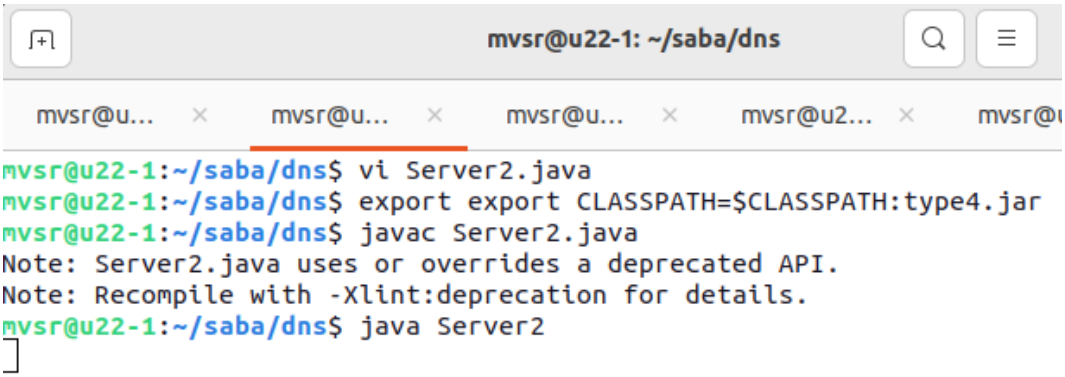
→ Enter name to resolve : google.com
→ 13.10

Sample Outputs:

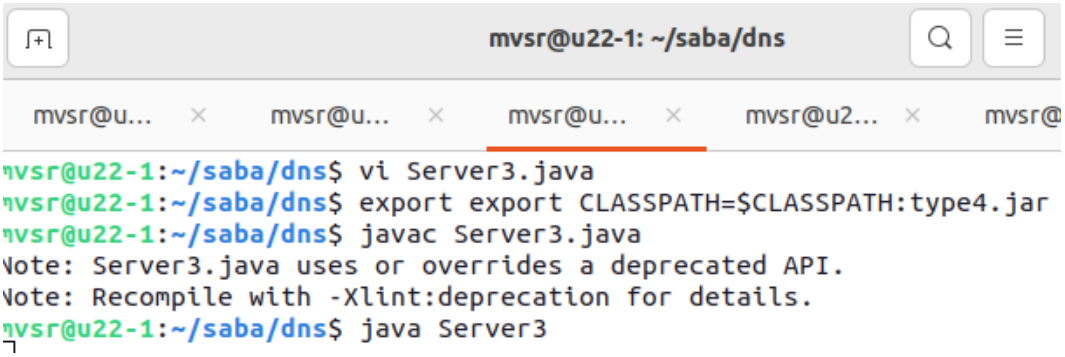
Server1



Server2



Server3



Client

4.Implementation of DNS using RMI

Introduction

This is a brief introduction to Java Remote Method Invocation (RMI). Java RMI is a mechanism that allows one to invoke a method on an object that exists in another address space. The other address space could be on the same machine or a different one. The RMI mechanism is basically an object-oriented RPC mechanism.

There are three processes that participate in supporting remote method invocation.

1. The *Client* is the process that is invoking a method on a remote object.
2. The *Server* is the process that owns the remote object. The remote object is an ordinary object in the address space of the server process.
3. The *Object Registry* is a name server that relates objects with names. Objects are *registered* with the Object Registry. Once an object has been registered, one can use the Object Registry to obtain access to a remote object using the name of the object.

RMI Stubs and Skeletons

- RMI uses stub and skeleton objects to provide the connection between the client and the remote object
- A stub is a proxy for a remote object which is responsible for forwarding method invocations from the client to the server where the actual remote object implementation resides
- A client's reference to a remote object, therefore, is actually a reference to a local stub. The client has a local copy of the stub object.
- A skeleton is a server-side object which contains a method that dispatches calls to the actual remote object implementation
- A remote object has an associated local skeleton object to dispatch remote calls to it.
- The skeleton object is automatically provided on the server side.
- A method can get a reference to a remote object
  - by looking up the remote object in some directory service. RMI provides a simple directory service called the RMI registry for this purpose
  - by receiving the remote object reference as a method argument or return value.

Steps To Develop an RMI Application

1. Design and implement the components of your distributed application
2. Define the remote interface(s)
3. Implement the remote object(s)
4. Implement the client(s)
5. Compile sources and generate stubs (and skeletons)
6. Make required classes network accessible.

7. Run the application.

DNSServer

```
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;
public class DNSServer extends UnicastRemoteObject implements DNSServerIntf {
   public DNSServer() throws RemoteException {
      super();
   }
   public String DNS(String s1) throws RemoteException {
      if (s1.equals("www.osmania.ac.in"))
         return "50.32.24.29";
      if (s1.equals("www.mvsrec.edu.in"))
         return "90.82.44.89";
      if (s1.equals("www.jntu.ac.in"))
         return "150.32.64.20";
      if (s1.equals("www.yahoo.com"))
         return "88.39.124.129";
      else
         return "No Info about this address";
   }
   public static void main(String args[]) throws Exception {
      Registry r = LocateRegistry.createRegistry(8145);
      r.rebind("mvsrserver", new DNSServer());
      System.out.println("server started...");
   }
}
```

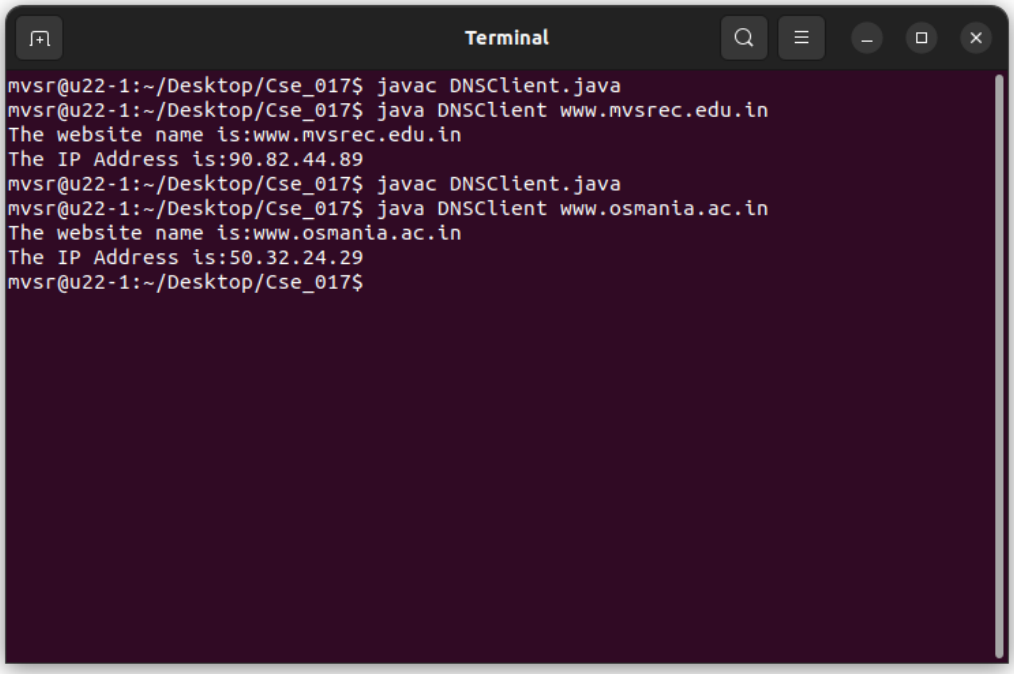
DNSClient

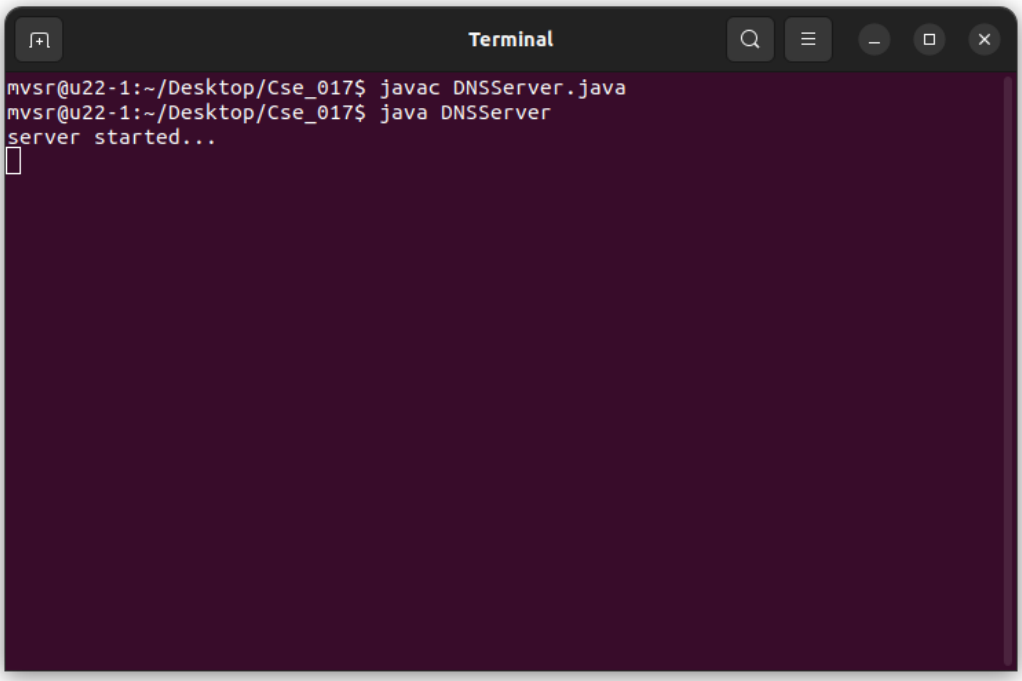
```
import java.rmi.*;
import java.rmi.registry.*;
public class DNSClient {
   public static void main(String args[]) throws Exception {
      Registry r = LocateRegistry.getRegistry("localhost", 8145);
      DNSServerIntf d = (DNSServerIntf) r.lookup("mvsrserver");
      String str = args[0];
      System.out.println("The website name is:" + str);
      System.out.println("The IP Address is:" + d.DNS(str));
   }
}
```

DNSServerIntf

```
import java.rmi.*;
public interface DNSServerIntf extends Remote {
   public String DNS(String s1) throws RemoteException;
}
```

Sample Outputs:





5.Understanding Working of NFS

NFS is a protocol for remote access to a file system developed by Sun

– It does not implement a file system per se

– Remote access is transparent to applications

• File system and OS independent

• Client/server architecture

– Client file system requests are forwarded to a remote server; NFS follows the remote access model

– Requests are implemented as remote procedure calls (RPCs)

Servers:

File sharing Server's:

1. NFS: Network File System: These servers are used in LAN
2. FTP: File Transfer Protocol: These servers are used in WAN & LAN
3. Samba Server: These servers are used in Linux & Window heterogeneous networks.

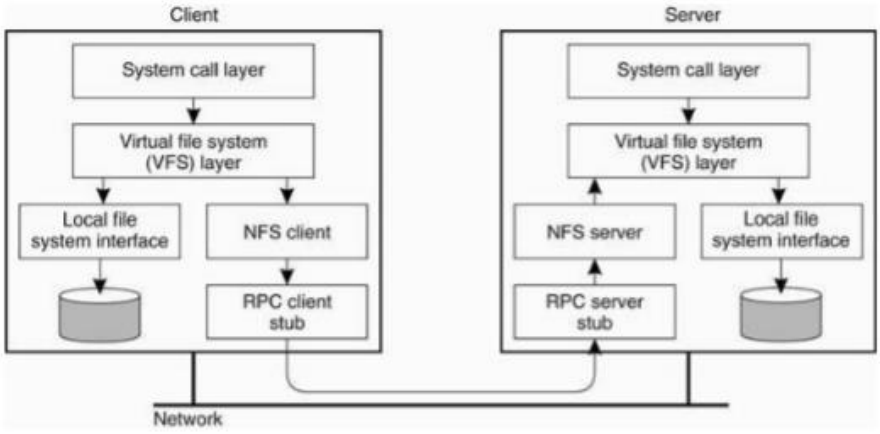
Note:

1. NFS is used in LAN because bandwidth for NFS is more than compared with FTP and Samba.
2. STP is used in WAN because FTP requires very less bandwidth when compared to NFS.

NFS (Network File System):-

Configuration Steps:

1) Copy all the shared files to a single location
   - /nfsshare aa bb cc
2) Create new file inside the share directory
   - aa bb cc
3) Packages:
   i) nfs-utils: to un-install the nfs-utils 60
   ii) portmap (please do not uninstall portmap
4)services
   - nfsportmap
5) Daemons:
   - nfs d quota d mount d
6) Main Configuration File: /etc/exports



Experiment:

Make a directory named nfsshare with files aa bb cc:

```
#mkdir /nfsshare
#cd /nfsshare
#touch aa bb cc
#lsaa
bb
cc
```

Check if the nfs-utils package is installed or not:

```
#rpm –q nfs-utils (Prints a message whether package is installed or not)
#rpm –q portmap
```

To reinstall the package first remove it with the following commands:

```
#servicesnfs stop
#rpm –e nfs-utils
#rm –rf /var/lib/nfs/xtab-------- remove
```

If package is not installed then there are two ways to install:

1. Download from FTP and install

```
#ping the server
#rpm –ivhftp://192.168.0.250:/pub.RedHat/RPMS/nfs* --force –aid
```

2. Install from CD

```
#mount /dev/cdrom/mnt
#cd /mnt
#ls
#cd Fedora
#cd RPMS

#rpm –ivhnfs-utils* --force –aid
#rpm –ivhportmap* --force –aid
```

After installing the nfs-utils package create file as below:

```
#vi /etc/exports
/var/ftp/pub 192.168.0.0./24(ro,sync)
/nfsshare 192.168.0.0.24(rw,sync)
```

Note:

in vi-editor write this content (/nfsshare server ipaddress and no. of systems that are connected in network)

After installing services enter the command to restart

```
#servicesnfs restart
```

Note:

execute this command twice because first it will show failed second time it will show ok.

Access to NFS share from client:

#mount –t nfs 10.10.12.114:/nfsshare/mnt

Note:

In client machine enter Server Ipaddress

Note:

1) #ping 192.168.0.3 –b

Broadcasts the address in the network only.

2) #ssh 192.168.0.8

Connects the PC to another PC just like Terminal connection in Windows.

3) In NFS all files & Directory are by default in read only mode.

Common KVM Switch: Using a KVM switch a monitor, keyboard, and a mouse can be connected to two computers.

5. Implement a word count application which counts the number of occurrences of each word a large collection of documents Using Map Reduce model

Workflow of MapReduce consists of 5 steps:

1. Splitting – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').
2. Mapping – as explained above.
3. Intermediate splitting – the entire process in parallel on different clusters. In order to group them in "Reduce Phase" the similar KEY data should be on the same cluster.
4. Reduce – it is nothing but mostly group by phase.
5. Combining – The last phase where all the data (individual result set from each cluster) is combined together to form a result.

Hadoop must be installed on your system with the Java SDK.

Steps

1. Open Eclipse> File > New > Java Project >(Name it –Samplewordcount) > Finish.
2. Right Click > New > Package (Name it - PackageDemo) > Finish.
3. Right Click on Package > New > Class (Name it - WordCount).
4. Add Following Reference Libraries:
    1. Right Click on Project > Build Path> Add External
        1. /usr/lib/hadoop-0.20/hadoop-core.jar
        2. Usr/lib/hadoop-0.20/lib/Commons-cli-1.2.jar

The program consists of three classes:

- Driver class (Public, void, static, or main; this is the entry point).
- The Map class which extends the public class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the Map function.
- The Reduce class which extends the public class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the Reduce function.

6. Make a jar file

Right Click on Project> Export> Select export destination as Jar File > next> Finish.

7. Take a text file and move it into HDFS format:

To move this into Hadoop directly, open the terminal and enter the following commands:

[training@localhost~]$hadoop fs -put wordcountFilewordCountFile

8. Run the jar file:

(Hadoop jar jarfilename.jar packageName.ClassNamePathToInputTextFilePathToOutputDirectry)
[training@localhost~]$hadoop jar MRProgramsDemo.jar PackageDemo.WordCountwordCountFile MRDir1

9. Open the result:
[training@localhost~]$hadoop fs -ls MRDir1
Found 3 items
-rw-r--r-- 1 training supergroup 0 2016-02-23 03:36 /user/training/MRDir1/_SUCCESS drwxr-xr-x - training supergroup 0 2016-02-23 03:36 /user/training/MRDir1/_logs -rw-r--r-- 1 training supergroup 20 2016-02-23 03:36 /user/training/MRDir1/part-r-00000
[training@localhost~]$hadoop fs -cat MRDir1/part-r-00000
BUS 7
CAR 4
TRAIN 6

WordCount.java

```
package PackageDemo;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;


public class WordCount {

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

        Configuration c = new Configuration();

        String[] files = new GenericOptionsParser(c, args).getRemainingArgs();

        Path input = new Path(files[0]);
```

```
    Path output = new Path(files[1]);
    Job j = new Job(c, "wordcount");
    j.setJarByClass(WordCount.class);
    j.setMapperClass(MapForWordCount.class);
    j.setReducerClass(ReduceForWordCount.class);
    j.setOutputKeyClass(Text.class);
    j.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(j, input);
    FileOutputFormat.setOutputPath(j, output);
    System.exit(j.waitForCompletion(true) ? 0 : 1);
}

public static class MapForWordCount extends Mapper < LongWritable, Text, Text, IntWritable > {
    public void map(LongWritable key, Text value, Context con) throws IOException,
    InterruptedException {
        String line = value.toString();
        String[] words = line.split(",");
        for (String word: words) {
            Text outputKey = new Text(word.toUpperCase().trim());
            IntWritableoutputValue = new IntWritable(1);
            con.write(outputKey, outputValue);
        }
    }
}
public static class ReduceForWordCount extends Reducer < Text, IntWritable, Text, IntWritable > {
    public void reduce(Text word, Iterable < IntWritable > values, Context con) throws IOException,
    InterruptedException {
        int sum = 0;
        for (IntWritable value: values) {
            sum += value.get();
```

```
            }
            con.write(word, new IntWritable(sum));
        }
    }
}
```

7. Develop an application using 3-tier architectures

React Native (Sample code)

```
// App.js
import {io} from 'socket.io-client'
import React, { useState, useEffect, useRef } from 'react';
import { View, StyleSheet, Text, TouchableOpacity, FlatList, TextInput, Image } from 'react-native';
import { MaterialIcons } from '@expo/vector-icons';
import * as ImagePicker from 'expo-image-picker';
import * as Notifications from 'expo-notifications';
import LoadingScreen from './LoadingScreen'; // Adjust the path accordingly

const logo = require('./assets/logo.png');

export default function App() {
  const [currentGroup, setCurrentGroup] = useState('Valorant');
  const [messages, setMessages] = useState([]);
  const [newMessage, setNewMessage] = useState("");
  const [selectedImage, setSelectedImage] = useState(null);
  const [isLoading, setIsLoading] = useState(true);
  const scrollViewRef = useRef();
  const [socket,setSocket]=useState(null);

  useEffect(()=>{
    if(socket && scrollViewRef)
    socket.on("newMsgRec",async (msgs)=>{
      const {text,stream,type}=msgs;
      console.log(msgs,"msgs");
      setMessages((prevMessages) => [
        ...prevMessages,
        {
```

```
        text: text,
        group: stream,
        sender: type,
      },
    ]);
    scrollViewRef.current.scrollToEnd({ animated: true });
  });
  if(socket && scrollViewRef)
  socket.on("newImgRec",async (img)=>{
    console.log(img,"img")
    setMessages((prevMessages)=>{[
      ...prevMessages,
      img]
    })
    scrollViewRef.current.scrollToEnd({ animated: true });
  })

},[socket])

useEffect(() => {
  // Simulating an asynchronous operation (e.g., fetching data) for 2 seconds
  const loadingTimer = setTimeout(() => {
    setIsLoading(false);
  }, 2000);

  //const soc = io('http://localhost:3001', {transports: ['websocket']})
  const soc = io('http://192.168.74.135:3001', { transports: ['websocket'] })

  setSocket(soc);
```

```
  soc.on('connect',()=>{
    console.log('connected')
  })

  // Clear the timer if the component is unmounted
  return () => clearTimeout(loadingTimer);
}, []);

const switchGroup = (group) => {
  setCurrentGroup(group);
};

const sendNotification = async () => {
  try {
    const notificationContent = {
      title: 'Players Needed',
      body: 'We need more players! Join the game.',
    };

    // Add a system message to the messages array
    setMessages((prevMessages) => [
      ...prevMessages,
      {
        text: 'Players Needed! Join the game.',
        group: currentGroup,
        sender: 'ping',
      },
    ]);
    const pingdata = {text: 'Players Needed! Join the game.',
    stream: currentGroup,
```

```
  type: 'ping'}

  socket.emit("newMsg",pingdata);

  // Send notifications to all users in the current group
  await Promise.all(
    messages
      .filter((msg) => msg.group === currentGroup && msg.sender !== 'me')
      .map(async (user) => {
        try {
          const notification = await Notifications.scheduleNotificationAsync({
            content: notificationContent,
            to: user.sender,
            trigger: null, // Explicitly set trigger to null for immediate notification
          });

          if (notification && notification.data) {
            const { requestor } = notification.data;
            console.log(`Notification sent to ${requestor}`);
          } else {
            console.log('Notification sent, but requestor information not available.');
          }
        } catch (error) {
          console.error('Error sending notification:', error);
        }
      })
  );

  console.log('Notifications sent successfully!');
} catch (error) {
```

```
    console.error('Error sending notification:', error);
  }
};



const pickImage = async () => {
  try {
    const permissionResult = await ImagePicker.requestMediaLibraryPermissionsAsync();

    if (permissionResult.granted === false) {
      alert('Permission to access camera roll is required!');
      return;
    }

    const result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.All,
      allowsEditing: true,
      aspect: [4, 3],
      quality: 1,
    });



    if (!result.canceled) {
      // Use the "assets" array instead of "uri"
      const selectedAsset = result.assets[0];
      setSelectedImage(selectedAsset.uri);
    }
  } catch (error) {
    console.error('Error picking image:', error);
  }
```

```
};

const sendMessage = () => {
  if (newMessage.trim()!=="") {
    setMessages((prevMessages) => [
      ...prevMessages,
      {
        text: newMessage,
        image: selectedImage, // Add the image URI to the message
        group: currentGroup,
        sender: 'me',
      },
    ]);

    const textdata={text: newMessage,
    stream: currentGroup,
    type: 'text'}
    socket.emit("newMsg",textdata);
    setNewMessage("");
    setSelectedImage(null); // Clear the selected image after sending

    // Scroll to the end of the list when a new message is sent
    scrollViewRef.current.scrollToEnd({ animated: true });
  }
  else if (selectedImage!==null){
    setMessages((prevMessages) => [
      ...prevMessages,
      {
        text: newMessage,
        image: selectedImage, // Add the image URI to the message
```

```
        group: currentGroup,
        sender: 'me',
      },
    ]);
    const temp = {
      text: newMessage,
      image: selectedImage, // Add the image URI to the message
      group: currentGroup,
      sender: 'me',
    };
    socket.emit("newImg",temp);
    setNewMessage("");
    setSelectedImage(null);
    scrollViewRef.current.scrollToEnd({ animated: true });
  }
};

const finishLoading = () => {
  setIsLoading(false);
};

// Add a new style for system messages
const systemMessageStyle = {
  backgroundColor: '#3498db', // Set the background color for system messages
  alignSelf: 'center', // Center-align the system messages
  color: 'white', // Set the text color for system messages
  borderRadius: 2, // Add some border-radius for a rounded appearance
  padding: 8, // Add padding to the system messages
  marginBottom: 8, // Adjust the margin at the bottom
};
```

```
return (
  <View style={styles.container}>
    {isLoading ? (
      <LoadingScreen onFinishLoading={finishLoading} />
    ) : (
      <View style={styles.container}>
        <View style={styles.header}>
          <Image source={logo} style={styles.logo} />
          <View style={styles.buttonRow}>
            <TouchableOpacity
              style={[styles.groupButton, currentGroup === 'Valorant' && styles.activeButton]}
              onPress={() => switchGroup('Valorant')}
            >
              <Text style={styles.buttonText}>Valorant</Text>
            </TouchableOpacity>
            <TouchableOpacity
              style={[styles.groupButton, currentGroup === 'CSGO' && styles.activeButton]}
              onPress={() => switchGroup('CSGO')}
            >
              <Text style={styles.buttonText}>CS:GO</Text>
            </TouchableOpacity>
            <TouchableOpacity
              style={[styles.groupButton, currentGroup === 'RocketLeague' && styles.activeButton]}
              onPress={() => switchGroup('RocketLeague')}
            >
              <Text style={styles.buttonText}>R.L</Text>
            </TouchableOpacity>
          </View>
        </View>
```

```
<View style={styles.messagesContainer}>
  <FlatList
    ref={scrollViewRef}
    data={messages.filter((msg) => msg.group === currentGroup)}
    keyExtractor={(item, index) => index.toString()}
    renderItem={({ item }) => (
      <View
        style={[
          styles.messageContainer,
          item.sender === 'me' ? styles.myMessageContainer : styles.otherMessageContainer,
          item.sender === 'ping' && systemMessageStyle, // Apply system message style
        ]}
      >
        {item.image ? (
          <View style={styles.imageContainer}>
            <Image source={{ uri: item.image }} style={styles.messageImage} />
            {item.sender !== 'ping' && (
              <MaterialIcons name="attachment" size={16} color="white" style={styles.attachmentIcon} />
            )}
          </View>
        ) : (
          <Text style={styles.messageText}>{item.text}</Text>
        )}
      </View>
    )}
  />
</View>

<View style={styles.inputContainer}>
```

```
          <TextInput
            style={styles.input}
            // placeholder="Type your message..."
            value={newMessage}
            onChangeText={(text) => setNewMessage(text)}
          />
          <TouchableOpacity style={styles.sendButton} onPress={sendMessage}>
            <MaterialIcons name="send" size={24} color="white" />
          </TouchableOpacity>
          <TouchableOpacity style={styles.sendButton} onPress={pickImage}>
            <MaterialIcons name="image" size={24} color="white" />
          </TouchableOpacity>
          <TouchableOpacity style={styles.sendButton} onPress={sendNotification}>
            <MaterialIcons name="notifications" size={24} color="white" />
          </TouchableOpacity>
        </View>
      </View>
    )}
  </View>
 );
}
```

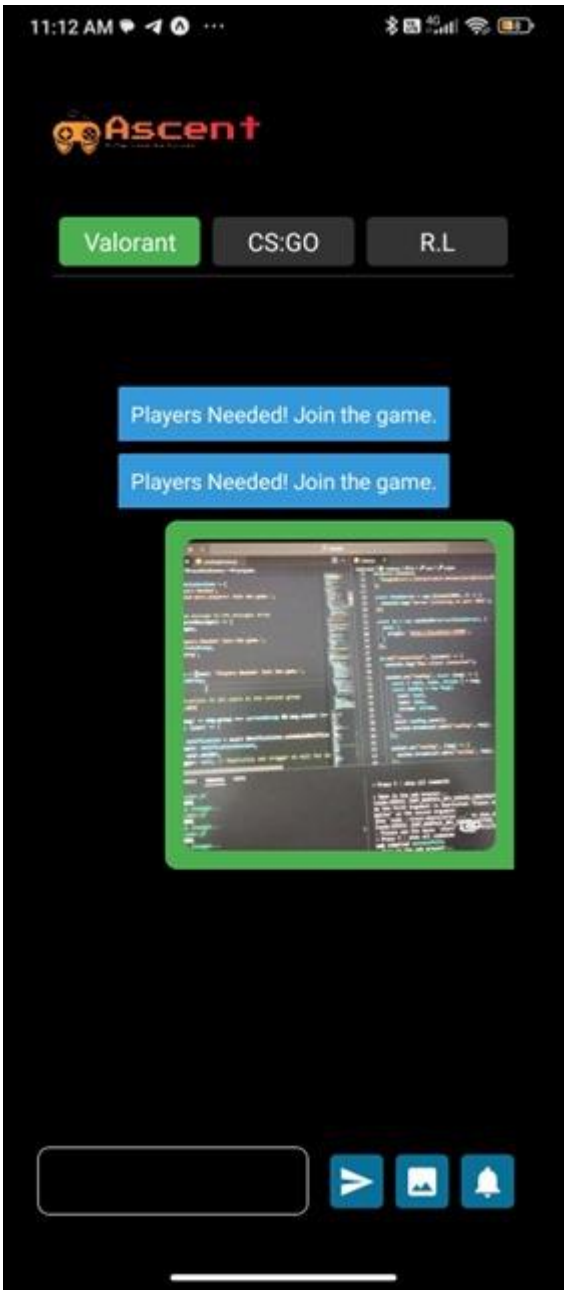
Node.js

```
const express = require("express");
const mongoose = require("mongoose");
const socketIO = require("socket.io");
const cors = require("cors");
const Msg = require("./models/Msg");

// App - express & socketIO
const app = express();
```

```
app.use(express.json());
app.use(cors());
mongoose.connect(
  "mongodb+srv://miniproject:miniproject@cluster0.7rtoqyt.mongodb.net/Ascent?retryWrites=true&w=majority"
);
const httpServer = app.listen(3001, () => {
  console.log(`Server listening on port 3001`);
});
const io = new socketIO.Server(httpServer, {
  cors: {
    origin: "*",
  }
});
io.on("connection", (socket) => {
  console.log("New client connected");
  socket.on("newMsg", async (msg) => {
    const { text, type, stream } = msg;
    const newMsg = new Msg({
      text: text,
      type: type,
      stream: stream,
    });
    await newMsg.save();
    console.log(msg,"msg");
    socket.broadcast.emit("newMsgRec", msg);
  });
  socket.on("newImg", (img) => {
    console.log(img,"img");
    socket.broadcast.emit("newImgRec", img);
```

```
  });
  socket.on("disconnect", () => {
    console.log(socket.id + " disconnected");
  });
});
app.post("/newAccount", async (req, res) => {
  try {
    const data = await Msg.find().sort({ createdAt: 1 });
    return res.send(data);
  } catch (err) {
    return res.status(500).json({ error: err.message });
  }
});
app.get("/hi", async (req, res) => {
  const newMsg = new Msg({
    text: "endhi bro",
    type: "text",
    stream: "valo",
  });
  res.send(await newMsg.save());
});
```

OutPut :


11:12 AM
Ascent
Valorant
CS:GO
R.L
Players Needed! Join the game.
Players Needed! Join the game.