

Roll No. **42**

Exam Seat No. _____

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Hashu Advani Memorial Complex, Collector's Colony, R. C.
Marg, Chembur, Mumbai – 400074. Contact No. 02261532532



Since 1962

CERTIFICATE

Certified that Mr. **CHRISTY PHILIP [ROLL NO: 42]** of **SYMCA-2B** has satisfactorily completed a course of the necessary experiments in **MCAL32 - Distributed System and Cloud Computing Lab** under the supervision of **Mr. Sunny Nahar** in the Institute of Technology in the academic year **2022- 2023.**

Principal

Head of Department

Lab In-charge

Subject Teacher



**V.E.S. Institute of Technology, Collector Colony,
Chembur, Mumbai**

Department of M.C.A

MCAL32 - DISTRIBUTED SYSTEM AND CLOUD COMPUTING LAB INDEX

Sr. No	Contents	Date Of Preparation	Date Of Submission	Marks	Sign
1	Remote Process Communication.	22/08/2022	29/08/2022	10	
2	Remote Procedure call using datagram (Calculator)	22/08/2022	29/08/2022	10	
3	Remote Procedure call using datagram (Date and Time)	29/08/2022	05/09/2022	10	
4	Remote Method Invocation (Stubs & Skeletons)	05/09/2022	14/09/2022	10	
5	Remote Method Invocation (Equation Solver)	05/09/2022	14/09/2022	10	
6	Implementation of Remote Method Communication using JDBC and RMI.	14/09/2022	28/09/2022	10	
7	Implementation of mutual exclusion using the Token ring algorithm.	28/09/2022	12/10/2022	10	
8	Implementation of Storage as a Service using Google Docs.	12/10/2022	19/10/2022	10	
9	Application using Google App Engine.	19/10/2022	02/11/2022	10	
10	Identity Access Management in Amazon Web Services.	02/11/2022	16/11/2022	10	

PRACTICAL 1

Remote Process Communication

AIM: - To develop a multi-client chat server application where multiple clients chat with each other concurrently, where the messages sent by different clients are first communicated to the server and then the server, on behalf of the source client, communicates the messages to the appropriate destination client.

THEORY: -

A remote procedure call is an interprocess communication technique that is used for client-server based applications. It is also known as a subroutine call or a function call.

A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

The sequence of events in a remote procedure call are given as follows –

The client stub is called by the client.

The client stub makes a system call to send the message to the server and puts the parameters in the message.

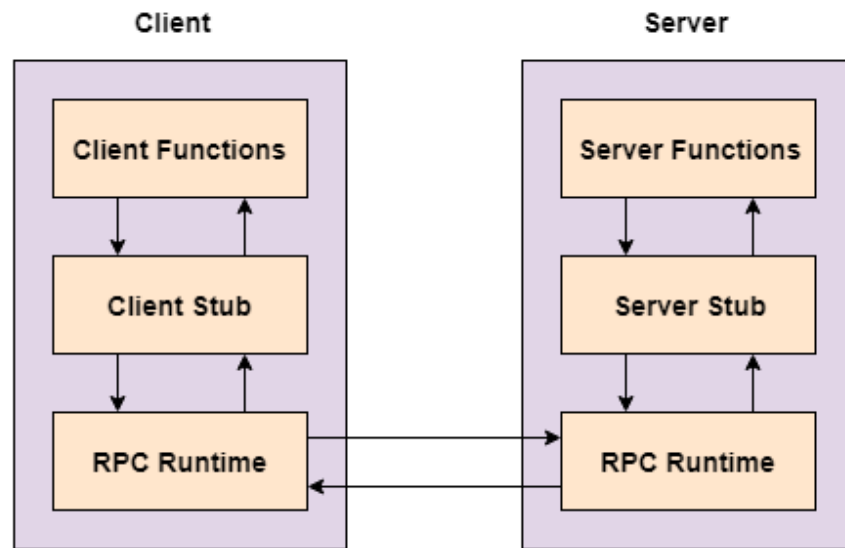
The message is sent from the client to the server by the client's operating system.

The message is passed to the server stub by the server operating system.

The parameters are removed from the message by the server stub.

Then, the server procedure is called by the server stub.

A diagram that demonstrates this is as follows –



Advantages of Remote Procedure Call

Some of the advantages of RPC are as follows –

- 1) Remote procedure calls support process oriented and thread oriented models.
- 2) The internal message passing mechanism of RPC is hidden from the user.
- 3) The effort to re-write and re-develop the code is minimum in remote procedure calls.
- 4) Remote procedure calls can be used in distributed environment as well as the local environment.
- 5) Many of the protocol layers are omitted by RPC to improve performance.

Disadvantages of Remote Procedure Call

Some of the disadvantages of RPC are as follows –

- 1) The remote procedure call is a concept that can be implemented in different ways. It is not a standard.
- 2) There is no flexibility in RPC for hardware architecture. It is only interaction based.
- 3) There is an increase in costs because of remote procedure call.

Client.java

```
package prac1;

import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) {
        String name = "empty";
        String reply = "empty";
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your name (Please enter your name to join the chat): ");
        reply = sc.nextLine();
        name = reply;
        try (Socket socket = new Socket("localhost", 5000)) {
            PrintWriter cout = new PrintWriter(socket.getOutputStream(), true);
            ThreadClient threadClient = new ThreadClient(socket);
            new Thread(threadClient).start(); // start thread to receive message
            cout.println(reply + ": has joined chat-room.");
            do {
                String message = (name + " : ");
                reply = sc.nextLine();
                if (reply.equals("logout")) {
                    cout.println("logout");
                    break;
                }
                cout.println(message + reply);
            } while (!reply.equals("logout"));
        } catch (Exception e) {
            System.out.println(e.getStackTrace());
        }
    }
}
```

ThreadClient.java

```
package prac1;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.net.SocketException;
public class ThreadClient implements Runnable {
    private Socket socket;
    private BufferedReader cin;
    public ThreadClient(Socket socket) throws IOException {
        this.socket = socket;
        this.cin = new BufferedReader(new
        InputStreamReader(socket.getInputStream()));
    }
    @Override
    public void run() {
        try {
            while (true) {
                String message = cin.readLine();
                System.out.println(message);
            }
        } catch (SocketException e) {
            System.out.println("You left the chat-room");
        } catch (IOException exception) {
            System.out.println(exception);
        } finally {
            try {
                cin.close();
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
    }
}
```

Server.java

```
package prac1;

import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.HashMap;
public class Server {
    public static void main(String[] args) {
        ArrayList<Socket> clients = new ArrayList<>();
        HashMap<Socket, String> clientNameList = new HashMap<Socket, String>();
        try (ServerSocket serversocket = new ServerSocket(5000)) {
            System.out.println("Server is started...");
            while (true) {
                Socket socket = serversocket.accept();
                clients.add(socket);
                ThreadServer ThreadServer = new ThreadServer(socket, clients, clientNameList);
                ThreadServer.start();
            }
        } catch (Exception e) {
            System.out.println(e.getStackTrace());
        }
    }
}
```

ThreadServer.java

```
package prac1;
import java.io.*;
import java.net.Socket;
import java.net.SocketException;
import java.util.ArrayList;
import java.util.HashMap;
public class ThreadServer extends Thread {
    private Socket socket;
    private ArrayList<Socket> clients;
    private HashMap<Socket, String> clientNameList;
    public ThreadServer(Socket socket, ArrayList<Socket> clients, HashMap<Socket,
String> clientNameList) {
        this.socket = socket;
```

```

this.clients = clients;
this.clientNameList = clientNameList;
}
@Override
public void run() {
try {
BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
while (true) {
    String outputString = input.readLine();
    if (outputString.equals("logout")) {
        throw new SocketException();
    }
    if (!clientNameList.containsKey(socket)) {
        String[] messageString = outputString.split(":", 2);
        clientNameList.put(socket, messageString[0]);
        System.out.println(messageString[0] + messageString[1]);
        showMessageToAllClients(socket, messageString[0] +
        messageString[1]);
    } else {
        System.out.println(outputString);
        showMessageToAllClients(socket, outputString);
    }
}
} catch (SocketException e) {
    String printMessage = clientNameList.get(socket) + " left the chat room";
    System.out.println(printMessage);
    showMessageToAllClients(socket, printMessage);
    clients.remove(socket);
    clientNameList.remove(socket);
} catch (Exception e) {
    System.out.println(e.getStackTrace());
}
}

private void showMessageToAllClients(Socket sender, String outputString) {
    Socket socket;
    PrintWriter printWriter;
    int i = 0;
    while (i < clients.size()) {
        socket = clients.get(i);

```



```

        i++;
        try {
            if (socket != sender) {
                printWriter = new PrintWriter(socket.getOutputStream(), true);
                printWriter.println(outputString);
            }
        } catch (IOException ex) {
            System.out.println(ex);
        }
    }
}
}
}
}
}

```

OUTPUT: -

Server

```

Server is started...
christy has joined chat-room.
ronal has joined chat-room.
christy : hi
ronal : hello
christy left the chat room
ronal left the chat room

```

Client

```

Enter your name (Please enter your name to join the chat):
Raju
Christy has joined chat-room.
Hi
Christy : Hi

```

```

Enter your name (Please enter your name to join the chat):
Christy
Raju : Hi
Hi
Raju : How are you?

```

CONCLUSION: -

From this practical I have learned to implement Remote Procedure Call.

PRACTICAL 2

Remote Procedure call using Datagram

AIM: - To implement a Server calculator containing ADD (), MUL (), SUB (), DIV (), etc

THEORY: -

In distributed computing, RPC is when a computer program causes a procedure to execute in a different address space, which is coded as if it were a normal (local) procedure call, without the programmer explicitly coding the details for the remote interaction. This is a form of client–server interaction, typically implemented via a request–response message-passing system. RPC is an interprocess communication technique that is used for client-server-based applications. A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

A common pattern of communication used by application programs structured as a client/server pair is the request/reply message transaction: A client sends a request message to a server, and the server responds with a reply message, with the client blocking to wait for the reply. A transport protocol that supports the request/reply paradigm is much more than a UDP message going in one direction followed by a UDP message going in the other direction. It needs to deal with correctly identifying processes on remote hosts and correlating requests with responses. It may also need to overcome some or all of the limitations of the underlying network outlined in the problem statement at the beginning of this chapter.

While TCP overcomes these limitations by providing a reliable byte-stream service, it doesn't perfectly match the request/reply paradigm either. This section describes a third category of transport protocol, called RPC, that more closely matches the needs of an application involved in a request/reply message exchange.

The sequence of events in a remote procedure call are given as follows:

1. The client stub is called by the client.
2. The client stub makes a system call to send the message to the server and puts the parameters in the message.
3. The message is sent from the client to the server by the client's operating system.
4. The message is passed to the server stub by the server operating system.
5. The parameters are removed from the message by the server stub.
6. Then, the server procedure is called by the server stub.

DATAGRAM

A datagram is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed. Datagrams play a vital role as an alternative. They are bundles of information passed between machines. Once the datagram has been released to its intended target, there is no assurance that it will arrive or even that someone will be there to catch it. Likewise, when the datagram is received, there is no assurance that it hasn't been damaged in transit or that whoever sent it is still there to receive a response and it is crucial point to note.

A datagram is a basic transfer unit associated with a packet-switched network. Datagrams are typically structured in header and payload sections. Datagrams provide a connectionless communication service across a packet-switched network. The delivery, arrival time, and order of arrival of datagrams need not be guaranteed by the network.

Java implements datagrams on top of the UDP (User Datagram Protocol) protocol by using two classes:

- **DatagramPacket** object is the data container.
- **DatagramSocket** is the mechanism used to send or receive the DatagramPackets.

DatagramSocket defines four public constructors. They are shown here:

- DatagramSocket () throws SocketException
- DatagramSocket (int port) throws SocketException
- DatagramSocket (int port, InetAddress ipAddress) throws SocketException
- DatagramSocket (SocketAddress address) throws SocketException

SocketAddress is an abstract class that is implemented by the concrete class InetSocketAddress. InetSocketAddress encapsulates an IP address with a port number. All can throw a SocketException if an error occurs while creating the socket. DatagramSocket defines many methods. Two of the most important are send () and receive (), which are shown here:

- void send (DatagramPacket packet) throws IOException
- void receive (DatagramPacket packet) throws IOException

The send () method sends packet to the port specified by packet. The receive method waits for a packet to be received from the port specified by packet and returns the result.

Client-Side Programming

1. Establish a Socket Connection

To connect to another machine, we need a socket connection. A socket connection means the two machines have information about each other's network location (IP Address) and TCP port. The java.net.Socket class represents a Socket. To open a socket:

```
Socket socket = new Socket("127.0.0.1", 5000)
```

First argument – IP address of Server. (127.0.0.1 is the IP address of localhost, where code will run on single stand-alone machine).

Second argument – TCP Port. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

2. Communication

To communicate over a socket connection, streams are used to both input and output the data.

3. Closing the connection

The socket connection is closed explicitly once the message to server is sent

Server Programming

1. Establish a Socket Connection

To write a server application two sockets are needed.

- A ServerSocket which waits for the client requests (when a client makes a new Socket())
- A plain old Socket socket to use for communication with the client.

2. Communication

getOutputStream() method is used to send the output through the socket.

3. Close the Connection

After finishing, it is important to close the connection by closing the socket as well as input/output streams.

CODE: -

Server.java

```
package practical2;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.StringTokenizer;
class Server {
public static void main(String[] args)
throws IOException
{
DatagramSocket ds = new DatagramSocket(1234);
byte[] buf = null;
DatagramPacket DpReceive = null;
DatagramPacket DpSend = null;
while (true) {
buf = new byte[65535];
DpReceive = new DatagramPacket(buf, buf.length);
ds.receive(DpReceive);
String inp = new String(buf, 0, buf.length);
inp = inp.trim();
System.out.println("Equation Received:- "+ inp);
if (inp.equals("close")) {
System.out.println("Client sent closing");
break;
}
int result;
StringTokenizer st = new StringTokenizer(inp);
int oprnd1 = Integer.parseInt(st.nextToken());
String operation = st.nextToken();
```

```

int oprnd2 = Integer.parseInt(st.nextToken());
// Perform the required operation
if (operation.equals("+"))
result = oprnd1 + oprnd2;
else if (operation.equals("-"))
result = oprnd1 - oprnd2;
else if (operation.equals("*"))
result = oprnd1 * oprnd2;
else if (operation.equals("%"))
result = oprnd1 % oprnd2;
else
result = oprnd1 / oprnd2;
System.out.println("Sending the result...");
String res = Integer.toString(result);
buf = res.getBytes();
int port = DpReceive.getPort();
DpSend = new DatagramPacket(
buf, buf.length, InetAddress.getLocalHost(), port);
ds.send(DpSend);
} } }

```

Client.java

```

package practical2;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class Client {
public static void main(String args[])
throws IOException
{
Scanner sc = new Scanner(System.in);
DatagramSocket ds = new DatagramSocket();
InetAddress ip = InetAddress.getLocalHost();
byte buf[] = null;
while (true) {

```

```

System.out.print("Enter the equation in the format:");
String inp = sc.nextLine();
buf = new byte[65535];
buf = inp.getBytes();
DatagramPacket DpSend = new DatagramPacket( buf, buf.length, ip, 1234);
ds.send(DpSend);
if (inp.equals("bye"))
break;
buf = new byte[65535];
DatagramPacket DpReceive = new DatagramPacket(buf, buf.length);
ds.receive(DpReceive);
System.out.println("Answer = "+ new String(buf, 0, buf.length));
}
}
}

```

OUTPUTS: -

Server

```

Equation Received:- 10 + 15
Sending the result...
Equation Received:- 10 - 5
Sending the result...
Equation Received:- 10 * 5
Sending the result...
Equation Received:- 10 / 5
Sending the result...
Equation Received:- 15 % 5
Sending the result...

```

Client

```

Enter the equation in the format:10 + 15
Answer = 25
Enter the equation in the format:

```

```

Enter the equation in the format:10 - 5
Answer = 5
Enter the equation in the format:

```

```
Enter the equation in the format:10 / 5
Answer = 2
```

```
Enter the equation in the format:10 % 18
Answer = 0
Enter the equation in the format:
```

```
Enter the equation in the format:10 * 5
Answer = 50
Enter the equation in the format:
```

CONCLUSION: -

From this practical I have learned to implement Remote Procedure Call using Datagram.

PRACTICAL 3

AIM: - To implement a Date Time Server containing date () and time ().

THEORY: -

In distributed computing, RPC is when a computer program causes a procedure to execute in a different address space, which is coded as if it were a normal (local) procedure call, without the programmer explicitly coding the details for the remote interaction. This is a form of client–server interaction, typically implemented via a request–response message-passing system. RPC is an interprocess communication technique that is used for client-server-based applications. A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

A common pattern of communication used by application programs structured as a client/server pair is the request/reply message transaction: A client sends a request message to a server, and the server responds with a reply message, with the client blocking to wait for the reply. A transport protocol that supports the request/reply paradigm is much more than a UDP message going in one direction followed by a UDP message going in the other direction. It needs to deal with correctly identifying processes on remote hosts and correlating requests with responses. It may also need to overcome some or all of the limitations of the underlying network outlined in the problem statement at the beginning of this chapter.

While TCP overcomes these limitations by providing a reliable byte-stream service, it doesn't perfectly match the request/reply paradigm either. This section describes a third category of transport protocol, called RPC, that more closely matches the needs of an application involved in a request/reply message exchange.

The sequence of events in a remote procedure call are given as follows:

1. The client stub is called by the client.

2. The client stub makes a system call to send the message to the server and puts the parameters in the message.
3. The message is sent from the client to the server by the client's operating system.
4. The message is passed to the server stub by the server operating system.
5. The parameters are removed from the message by the server stub.
6. Then, the server procedure is called by the server stub.

Datagram

A datagram is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed. Datagrams play a vital role as an alternative. They are bundles of information passed between machines. Once the datagram has been released to its intended target, there is no assurance that it will arrive or even that someone will be there to catch it. Likewise, when the datagram is received, there is no assurance that it hasn't been damaged in transit or that whoever sent it is still there to receive a response and it is crucial point to note.

A datagram is a basic transfer unit associated with a packet-switched network. Datagrams are typically structured in header and payload sections. Datagrams provide a connectionless communication service across a packet-switched network. The delivery, arrival time, and order of arrival of datagrams need not be guaranteed by the network.

Java implements datagrams on top of the UDP (User Datagram Protocol) protocol by using two classes:

- DatagramPacket object is the data container.
- DatagramSocket is the mechanism used to send or receive the DatagramPackets.

DatagramSocket defines four public constructors. They are shown here:

- DatagramSocket () throws SocketException
- DatagramSocket (int port) throws SocketException
- DatagramSocket (int port, InetAddress ipAddress) throws SocketException
- DatagramSocket (SocketAddress address) throws SocketException

SocketAddress is an abstract class that is implemented by the concrete class

InetSocketAddress. InetSocketAddress encapsulates an IP address with a port number. All

can throw a `SocketException` if an error occurs while creating the socket. `DatagramSocket` defines many methods. Two of the most important are `send ()` and `receive ()`, which are shown here:

- void `send (DatagramPacket packet)` throws `IOException`
- void `receive (DatagramPacket packet)` throws `IOException`

The `send ()` method sends packet to the port specified by `packet`. The `receive` method waits for a packet to be received from the port specified by `packet` and returns the result.

Client-Side Programming

1. Establish a Socket Connection

To connect to another machine, we need a socket connection. A socket connection means the two machines have information about each other's network location (IP Address) and TCP port. The `java.net.Socket` class represents a `Socket`. To open a socket:

```
Socket socket = new Socket("127.0.0.1", 5000)
```

First argument – IP address of Server. (127.0.0.1 is the IP address of localhost, where code will run on single stand-alone machine).

Second argument – TCP Port. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

2. Communication

To communicate over a socket connection, streams are used to both input and output the data.

3. Closing the connection

The socket connection is closed explicitly once the message to server is sent

Server Programming

1. Establish a Socket Connection

To write a server application two sockets are needed.

- A `ServerSocket` which waits for the client requests (when a client makes a new `Socket()`)
- A plain old `Socket socket` to use for communication with the client.

2. Communication

`getOutputStream()` method is used to send the output through the socket.

3. Close the Connection

After finishing, it is important to close the connection by closing the socket as well as input/output streams.

CODE: -

Server.java

```
package practical3;

import java.util.*;
import java.net.*;
import java.text.SimpleDateFormat;
public class Server {
    DatagramSocket ds;
    DatagramPacket dp;
    String str,methodName,result;
    int val1,val2;
    Server()
    {
        try
        {
            ds=new DatagramSocket (1200);
            byte b[]=new byte[4096];
            while(true)
            {
                dp=new DatagramPacket(b,b.length);
                ds.receive(dp);
                str=new String(dp.getData(),0,dp.getLength());
                if(str.equalsIgnoreCase("q"))
                {
                    System.exit(1);
                }
                else
                {
                    StringTokenizer st = new StringTokenizer(str," ");
                    int i=0;
                    while(st.hasMoreTokens())
                    {
                        String token=st.nextToken();
                        methodName=token;
                    }
                }
                Calendar c = Calendar.getInstance();
                SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
                Date d = c.getTime();
                InetAddress ia = InetAddress.getLocalHost();
```

```

if(methodName.equalsIgnoreCase("date"))
{
result="" + dateFormat.format(d);
}
else if(methodName.equalsIgnoreCase("time"))
{
result= "" + d.getHours() + " : " +d.getMinutes() + " : " +d.getSeconds();
}
byte b1[]=result.getBytes();
DatagramSocket ds1 = new DatagramSocket();
DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
System.out.println("result : "+result+"\n");
ds1.send(dp1);
}}
catch (Exception e)
{
e.printStackTrace();
}}
public static void main(String[] args)
{
new Server();
}}

```

Client. java

```

package practical3;

import java.io.*;
import java.net.*;
public class Client {
Client()
{
try
{
InetAddress ia = InetAddress.getLocalHost();
DatagramSocket ds = new DatagramSocket();
byte b1[]=new byte[50];
DatagramSocket ds1 = new DatagramSocket(1300);
System.out.println("\nRPC Client\n");

```

```

while (true)
{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String str = br.readLine();
byte b[] = str.getBytes();
DatagramPacket dp = new DatagramPacket(b,b.length,ia,1200);
ds.send(dp);
dp = new DatagramPacket(b1,b1.length);
ds1.receive(dp);
String s = new String(dp.getData(),0,dp.getLength());
System.out.println("\nResult = " + s );
}}
catch (Exception e)
{
e.printStackTrace();
}}
public static void main(String[] args)
{
new Client();
}}

```

OUTPUT: -

Client

```

date

Result = 14/09/2022
time

Result = 21 : 29 : 27

```

Server

```

result : 14/09/2022

result : 21 : 29 : 27

```

CONCLUSION: -

From this practical I have learned to implement a Date Time Server containing date () and time ().

PRACTICAL 4

Remote Method Invocation (Stubs & Skeletons)

AIM: - To find date and time using Remote Method Invocation (Use stubs and skeletons)

THEORY: -

RMI (Remote Method Invocation):

The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM. The RMI provides remote communication between the applications using two objects stub and skeleton. RMI uses stub and skeleton object for communication with the remote object. A remote object is an object whose method can be invoked from another JVM.

Stub:

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine(JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

Skeleton:

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

Step: -

- 1) Write the code in the Netbeans.
- 2) Then goto the folder in which code is saved open the cmd
- 3) run javac *.java to compile all files together
- 4) then run the interface by the command (rmic Interface name)
- 5) rmi registry 5000 command
- 6) Open two command prompt one for server and ne for client

CODE: -**DateTime.java**

```
import java.rmi.*;
public interface DateTime extends Remote {
    public String date(int op) throws RemoteException;
}
```

DateTimeRemote.java

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;
import java.text.SimpleDateFormat;
public class DateTimeRemote extends UnicastRemoteObject implements DateTime {
    DateTimeRemote() throws RemoteException {
        super();
    }
    @Override
    public String date(int op) {
        Date dateObj = new Date();
        if (op == 1) {
            SimpleDateFormat df = new SimpleDateFormat("MM/dd/YYYY");
            String formattedDate = df.format(dateObj);
            return formattedDate;
        } else if (op == 2) {
            SimpleDateFormat df = new SimpleDateFormat("hh:mm:ss");
            String formattedTime = df.format(dateObj);
            return formattedTime;
        }
        return "a";
    }
}
```



```
}
```

MyClient.java

```
import java.rmi.*;
public class MyClient {
    public static void main(String args[]) {
        try {
            DateTime stub=(DateTime)Naming.lookup("rmi://localhost:5000/narender");
            System.out.println("Current Date: " + stub.date(1) + "\n" + "Current Time:" +
            stub.date(2));
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

MyServer.java

```
import java.rmi.*;
import java.rmi.registry.*;
public class MyServer {
    public static void main(String args[]) {
        try {
            DateTime stub = new DateTimeRemote();
            Naming.rebind("rmi://localhost:5000/narender",stub);
        } catch (Exception e) {
            System.out.print("exception on server");
            System.out.println(e);
        }
    }
}
```

OUTPUT: -

1) First cmd should be opened and direct to path in which netbeans code is saved

2) Run javac command

```
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Philip>cd C:\Users\Philip\OneDrive\Documents\NetBeansProjects\Prac4

C:\Users\Philip\OneDrive\Documents\NetBeansProjects\Prac4>cd src

C:\Users\Philip\OneDrive\Documents\NetBeansProjects\Prac4\src>javac *.java
```

3) run the rmic command with the interface name

```
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\Prac4\src>rmic DateTimeRemote
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
```

4) run the rmiregistry command

```
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\Prac4\src>rmiregistry 5000
```

5) Open two terminals one for client and one for server

Server

```
C:\Users\Philip>cd C:\Users\Philip\OneDrive\Documents\NetBeansProjects\Prac4\src

C:\Users\Philip\OneDrive\Documents\NetBeansProjects\Prac4\src>java MyServer
```

6) Client side run the client

```
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\Prac4\src>java MyClient
Current Date: 09/27/2022
Current Time:09:33:44
```

CONCLUSION: -

From this practical I have learned to implement date and time using Remote Method Invocation (Use stubs and skeletons).

PRACTICAL 5

Remote Method Invocation (RMI)

AIM: - Implementation of equation solver using Remote Method Invocation (RMI).

THEORY: -

RMI (Remote Method Invocation):

The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM. The RMI provides remote communication between the applications using two objects stub and skeleton. RMI uses stub and skeleton object for communication with the remote object. A remote object is an object whose method can be invoked from another JVM.

Stub:

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine(JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

Skeleton:

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

Steps: -

- 1) Write the code in the Netbeans.
- 2) Then goto the folder in which code is saved open the cmd
- 3) run `javac *.java` to compile all files together
- 4) then run the interface by the command (`rmic Interface name`)
- 5) `rmi registry 5000` command
- 6) Open two command prompt one for server and ne for client

CODE: -

EquationSolverInterface.ava

```
import java.rmi.*;
import java.util.ArrayList;
public interface EquationSolverInterface extends Remote {
    public int EvaluateEquation(String equation, ArrayList<Integer> params) throws
    RemoteException;
}
```

EquationSolverClass.java

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;
public class EquationSolverClass extends UnicastRemoteObject implements
EquationSolverInterface {
    EquationSolverClass() throws RemoteException {
        super();
    }
    @Override
    public int EvaluateEquation(String equation, ArrayList<Integer> params) {
        System.out.print("Equation recieved: " + equation);
        int a = 0, b = 0, c = 0;
        HashMap<String, Integer> map = new HashMap<>();
        if (params.size() == 2) {
            params.add(c);
        }
        map.put("a^2-b^2", (params.get(0) - params.get(1)) * (params.get(0) + params.get(1)));
        map.put("a^2+b^2", (int) (Math.pow((params.get(0) - params.get(1)), 2)+ 2 * params.get(0) *
        params.get(1)));
    }
}
```

```

map.put("(a+b)^2", (int) (Math.pow(params.get(0), 2) + 2 * params.get(0) * params.get(1)+
Math.pow(params.get(1), 2)));
map.put("(a-b)^2", (int) (Math.pow(params.get(0), 2) - 2 * params.get(0) * params.get(1)+
Math.pow(params.get(1), 2)));
map.put("(a+b+c)^2", (int) (Math.pow(params.get(0), 2) + Math.pow(params.get(1), 2)+
Math.pow(params.get(2), 2) + 2 * params.get(0) * params.get(1)+ 2 * params.get(1) *
params.get(2) + 2 * params.get(0) * params.get(2)));
map.put("(a-b-c)^2", (int) (Math.pow(params.get(0), 2) + Math.pow(params.get(1), 2)+
Math.pow(params.get(2), 2) - 2 * params.get(0) * params.get(1) + 2 * params.get(1)*
params.get(2)- 2 * params.get(0) * params.get(1)));
map.put("a^3-b^3", (int) ((params.get(0) - params.get(1))* (Math.pow(params.get(0), 2) +
params.get(0) * params.get(1) +Math.pow(params.get(1), 2))));
map.put("a^3+b^3", (int) ((params.get(0) + params.get(1)) * (Math.pow(params.get(0), 2)-
params.get(0) * params.get(1) + Math.pow(params.get(1), 2))));
map.put("(a+b)^3", (int) (Math.pow(params.get(0), 3) + 3 * Math.pow(params.get(0), 2)*
params.get(1) + 3 * params.get(0) * Math.pow(params.get(1), 2) +Math.pow(params.get(1), 3)));
map.put("(a-b)^3", (int) (Math.pow(params.get(0), 3) - 3 * Math.pow(params.get(0), 2)*
params.get(1) + 3 * params.get(0) * Math.pow(params.get(1), 2) -Math.pow(params.get(1), 3)));
int answer = map.get(equation);
System.out.print(answer);
return answer;
}
}

```

MyServer.java

```

import java.rmi.*;
public class MyServer {
public static void main(String args[]) {
try {
System.out.print("Server started, waiting for client to enter equation");
EquationSolverInterface stub = new EquationSolverClass();
Naming.rebind("rmi://localhost:5000/narenderEquationSolver", stub);
} catch (Exception e) {
System.out.print("exception on server");
System.out.println(e);
}
}
}
}

```

MyClient.java

```
import java.rmi.*;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner; // Import the Scanner class
import java.util.Set;
import java.util.TreeSet;
public class MyClient {
    public static void main(String args[]) {
        while (true) {
            try {
                Scanner sc = new Scanner(System.in);
                System.out.print("Enter equation: ");
                String equation = sc.nextLine();
                Set<String> tree = new TreeSet<>();
                ArrayList<Integer> params = new ArrayList<Integer>();
                for (char c : equation.toCharArray()) {
                    if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z')) {
                        tree.add(Character.toString(c));
                    }
                }
                int n = tree.size();
                Iterator value = tree.iterator();
                while (value.hasNext()) {
                    System.out.println("Enter value for " + value.next());
                    int temp = sc.nextInt();
                    params.add(temp);
                }
                EquationSolverInterface stub = (EquationSolverInterface)
                    Naming.lookup("rmi://localhost:5000/narenderEquationSolver");
                System.out.println("Answer: " + stub.EvaluateEquation(equation, params));
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }
}
```

OUTPUT: -

1) Goto the directory in which NetBeans code is there from there open cmd.

2) Compile all the codes by `javac *.java`

```
C:\Users\Philip>cd C:\Users\Philip\OneDrive\Documents\NetBeansProjects\newprac5\src
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\newprac5\src>javac *.java
```

3) run the interface using `rmic` interface name

```
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\newprac5\src>rmic EquationSolverClass
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
```

4) run the `rmiregistry` command

```
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\Prac4\src>rmiregistry 5000
```

5) Open 2 cmd one for server and another for client

Server

```
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\newprac5\src>java MyServer
Server started, waiting for client to enter equationEquation recieved: (a+b)^2900Equation recieved: (a-b)^31000
```

6) Now in client side run the equation

Client

```
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\newprac5\src>java MyClient
Enter equation: (a+b)^2
Enter value for a
10
Enter value for b
20
Answer: 900
Enter equation: (a-b)^3
Enter value for a
20
Enter value for b
10
Answer: 1000
```

CONCLUSION: -

From this practical I have learned to implement equation solver using Remote Method Invocation (RMI).

PRACTICAL 6

AIM: - Implementation of Remote Method Communication using JDBC and RMI.

THEORY: -

Java Database Connectivity (JDBC): JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. Java JDBC Architecture is

- 1) JDBC Application- The JDBC application is in the topmost position of the JDBC architecture. JDBC application is the data processing application that intends to access the data from the different data storage units.
- 2) JDBC API- The JDBC API plays a significant role in the JDBC architecture. The JDBC API ensures that a stable connection is established between the data storage unit and the JDBC application.
- 3) JDBC Manager- The JDBC manager takes care of selecting the appropriate driver manager to ensure that the driver software chosen supports the data storage unit's requirements to offer an uninterrupted connection.
- 4) JDBC Drivers-The JDBC driver is the crucial unit in the JDBC architecture. JDBC driver is the first layer of JDBC architecture that has direct contact with the data storage units.
- 5) Data Storage Units-Data storage units are the base of JDBC. The data storage unit is the place where all the data is kept accessible for the JDBC Applications.

Steps to Connect Java JDBC

Step 1: Import the database

Step 2A: Add the mysql connector to project

Step 2B: Loading the drivers- In order to begin with, you first need to load the driver or register it before using it in the program. Registration is to be done once in your program.

Class.forName()- Here we load the driver's class file into memory at the runtime. No need of using new or create objects. The following example uses Class.forName() to load the

MYSQL driver as shown below as follows:

```
Class.forName("com.mysql.jdbc.Driver");
```

Step 3: Establish a connection using the Connection class object After loading the driver, establish connections via as shown below as follows:

```
Connection con = DriverManager.getConnection(url,user,password)
```

- 1) user: Username from which your SQL command prompt can be accessed.
- 2) password: password from which the SQL command prompt can be accessed.
- 3) con: It is a reference to the Connection interface.
- 4) url- Uniform Resource Locator. For eg. jdbc:mysql://localhost:3306/mysql

Step 4: Create a statement- Once a connection is established you can interact with the database. The JDBCStatement, CallableStatement, and PreparedStatement interfaces define the methods that enable you to send SQL commands and receive data from your database.

Use of JDBC Statement is as follows:

```
Statement st = con.createStatement();
```

Step 5: Execute the query

Now comes the most important part i.e executing the query. The query here is an SQL Query.

Now we know we can have multiple types of queries. Some of them are as follows:

- 1) executeQuery() - The executeQuery() method of the Statement interface is used to execute queries of retrieving values from the database. This method returns the object of ResultSet that can be used to get all the records of a table.
- 2) executeUpdate(sql query)- The executeUpdate(sql query) method of the Statement interface is used to execute queries of updating/inserting.

Step 6: Closing the connections- So finally we have sent the data to the specified location and now we are on the verge of completing of our task. By closing the connection, objects of Statement and ResultSet will be closed automatically. The close() method of the Connection interface is used to close the connection. It is as shown below as follows:

```
con.close();
```

RMI (Remote Method Invocation):

The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM. The RMI provides remote communication between the applications using two objects stub and skeleton. RMI uses stub and skeleton object for communication with the remote object. A remote object is an object whose method can be invoked from another JVM.

Stub: The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object.

When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM)
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally returns the value to the caller.

Skeleton: The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshalls) the result to the caller.

The 6 steps to write

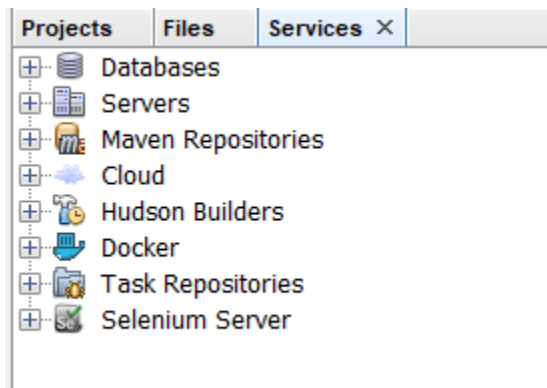
1. Create the remote interface
2. Provide the implementation of the remote interface
3. Compile the implementation class and create the stub and skeleton objects using the rmic tool
4. Start the registry service by rmiregistry tool
5. Create and start the remote application
6. Create and start the client application

A) Using MySQL create a Library database. Create table Book (Book_id, Book_name, Book_author) and retrieve the Book information from the Library database.

MySQL Installation Video: - <https://youtu.be/BOUMR85B-V0>

STEPS: -

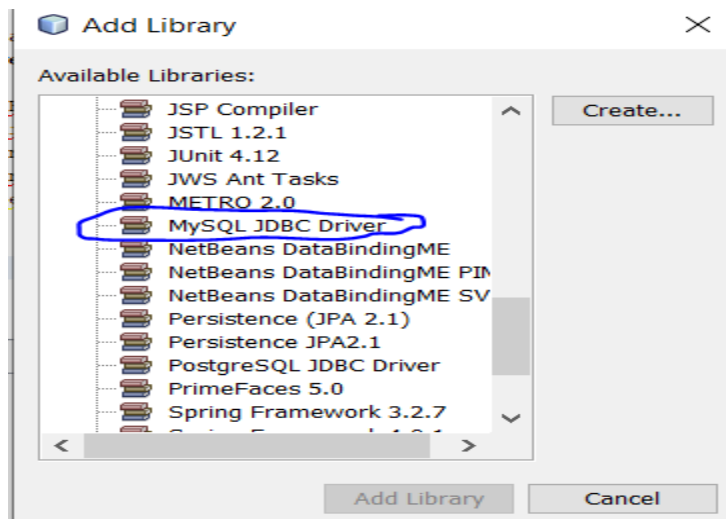
- 1) Create the database in MySQL 5.7 for that follow below steps and images
- 2) Copy paste the code given below in Netbeans
- 3) Establish the connection between SQL and Netbeans for that come to Services
 - Right click on the Databases New Connection
 - Add the connector and provide required data



Refer below youtube link for Step 3

<https://youtu.be/-nup6K1z4Po>

- 4) Also add the library in the Project



- 5) Now the code needs to be compiled
- 6) Open 3 cmds one for server and one for client, refer output section.

DATABASE CREATION: -

create database library_data;

```
mysql> create database library_data;  
Query OK, 1 row affected (0.00 sec)
```

show databases;

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| electric_bill |  
| library |  
| library_data |  
| mysql |  
| performance_schema |  
| sakila |  
| sys |  
| world |  
+-----+  
9 rows in set (0.00 sec)
```

use library_data;

```
mysql> use library_data;  
Database changed
```

create table books (
books_id int NOT NULL AUTO_INCREMENT,
books_name varchar(55) NOT NULL,
books_author varchar(55) NOT NULL,
PRIMARY KEY(books_id)
);

```
mysql> create table books (  
-> books_id int NOT NULL AUTO_INCREMENT,  
-> books_name varchar(55) NOT NULL,  
-> books_author varchar(55) NOT NULL,  
-> PRIMARY KEY(books_id)  
-> );  
Query OK, 0 rows affected (0.03 sec)
```

```
insert into books(books_name,books_author) values('HALF GIRLFRIEND','CHETAN BHAGAT');
```

```
insert into books(books_name,books_author) values('WINGS OF FIRE','APJ ABDUL KALAM');
```

```
insert into books(books_name,books_author) values('DISCOVERY OF INDIA','PANDIT JAWAHARLAL NEHRU');
```

```
mysql> insert into books(books_name,books_author) values('HALF GIRLFRIEND','CHETAN BHAGAT');
Query OK, 1 row affected (0.01 sec)

mysql> insert into books(books_name,books_author) values('WINGS OF FIRE','APJ ABDUL KALAM');
Query OK, 1 row affected (0.01 sec)

mysql> insert into books(books_name,books_author) values('DISCOVERY OF INDIA','PANDIT JAWAHARLAL NEHRU');
Query OK, 1 row affected (0.00 sec)
```

```
SELECT * FROM books;
```

```
mysql> SELECT * FROM books;
+-----+-----+-----+
| books_id | books_name      | books_author      |
+-----+-----+-----+
| 1        | HALF GIRLFRIEND | CHETAN BHAGAT     |
| 2        | WINGS OF FIRE   | APJ ABDUL KALAM   |
| 3        | DISCOVERY OF INDIA | PANDIT JAWAHARLAL NEHRU |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

CODE: -

IDB.java

```
/**
 *
 * @author Philip
 */
import java.rmi.*;

public interface IDB extends Remote {
    public String getData() throws RemoteException;
    public String getData(int id) throws RemoteException;
}
```

DBImpl.java

```
/**
 *
 * @author Philip
 */
import java.rmi.*;
import java.rmi.server.*;
import java.sql.*;
public class DBImpl extends UnicastRemoteObject implements IDB {
    String str, str1;
    public DBImpl() throws RemoteException {
    }
    public String getData() {
        String URL = "jdbc:mysql://localhost:3306/library_data";
        String UName = "root";
        String Pass = "password";
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection(URL, UName, Pass);
            Statement s = con.createStatement();
            ResultSet rs = s.executeQuery("select * from books");
            ResultSetMetaData rsmd = rs.getMetaData();
            str = "";
            str1 = "";
            for (int i = 1; i <= rsmd.getColumnCount(); i++) {
                str1 = str1 + rsmd.getColumnName(i) + "\t";
            }
            System.out.println();
            while (rs.next()) {
                for (int i = 1; i <= rsmd.getColumnCount(); i++) {
                    str = str + rs.getString(i) + "\t";
                }
                str = str + "\n";
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return (str1 + "\n" + str);
    }
    @Override
```

```

public String getData(int id) throws RemoteException {
    String URL = "jdbc:mysql://localhost:3306/library_data";
    String UName = "root";
    String Pass = "password";
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection(URL, UName, Pass);
        PreparedStatement ps = con.prepareStatement("select * from books where books_id=?");
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        ResultSetMetaData rsmd = rs.getMetaData();
        str = "";
        str1 = "";
        for (int i = 1; i <= rsmd.getColumnCount(); i++) {
            str1 = str1 + rsmd.getColumnName(i) + "\t";
        }
        System.out.println();
        while (rs.next()) {
            for (int i = 1; i <= rsmd.getColumnCount(); i++) {
                str = str + rs.getString(i) + "\t";
            }
            str = str + "\n";
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return (str1 + "\n" + str);
}
}

```

Server.java

```

/**
 *
 * @author Philip
 */
import java.rmi.Naming;
public class Server {
    public static void main(String[] args) {
        try {
            DBImpl di = new DBImpl();

```



```

Naming.rebind("rmi://127.0.0.1/DBServer6B", di);
System.out.println("Server Registered.");
} catch (Exception e1) {
e1.printStackTrace();
}
}
}

```

Client.java

```

/**
 *
 * @author Philip
 */
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.rmi.Naming;
public class Client {
public static void main(String[] args) {
String ch1 = "", res = "";
try {
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String url = "rmi://127.0.0.1/DBServer6B";
System.out.println("Retriving Books Information....");
while (true) {
System.out.print("Enter choice:\n1. Get All Books\n2. Get Book By Id ");
ch1 = br.readLine();
if (ch1.equals("1")) {
IDB id = (IDB) Naming.lookup(url);
res = id.getData();
System.out.println(res);
} else if (ch1.equals("2")) {
BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));
IDB id1 = (IDB) Naming.lookup(url);
res = id1.getData(Integer.parseInt(br1.readLine()));
System.out.println(res);
} else {
System.out.println("Please select an option");
}
}
} catch (Exception e) {

```

```
e.printStackTrace();
}
}
}
```

OUTPUT: -

```
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\dsc6b\src>start rmiregistry
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\dsc6b\src>
```

Run the file in netbeans after above execution of cmd by right clicking and choosing

Server.java

```
run:
Server Registered.
```

Client.java

```
run:
Retriving Books Information....
Enter choice:
1. Get All Books
2. Get Book By Id 1
books_id      books_name      books_author
1      HALF GIRLFRIEND CHETAN BHAGAT
2      WINGS OF FIRE    APJ ABDUL KALAM
3      DISCOVERY OF INDIA      PANDIT JAWAHARLAL NEHRU

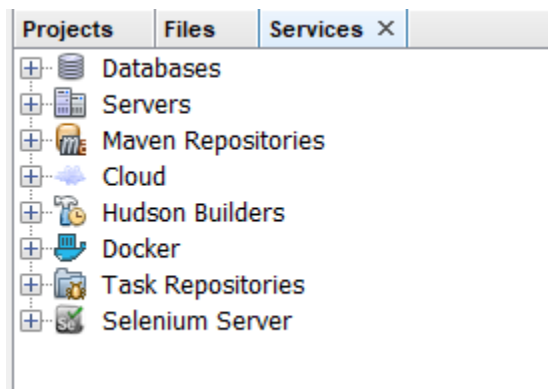
Enter choice:
1. Get All Books
2. Get Book By Id 2
1
books_id      books_name      books_author
1      HALF GIRLFRIEND CHETAN BHAGAT

Enter choice:
1. Get All Books
2. Get Book By Id |
```

B) Using MySQL create Electric_Bill database. Create table Bill. (bill_id, consumer_name, bill_due_date, bill_amount) and retrieve the Bill information from the Electric_Bill database.

STEPS: -

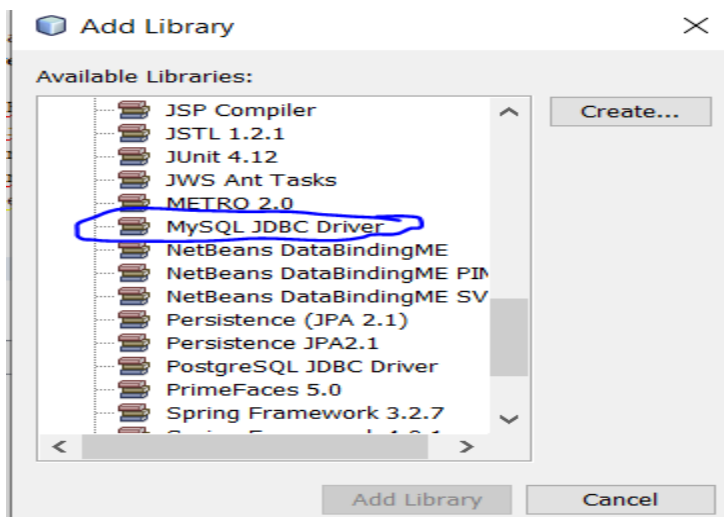
- 1) Create the database in MySQL 5.7 for that follow below steps and images
- 2) Copy paste the code given below in Netbeans
- 3) Establish the connection between SQL and Netbeans for that come to Services
 - Right click on the Databases New Connection
 - Add the connector and provide required data



Refer below youtube link for Step 3

<https://youtu.be/-nup6K1z4Po>

- 4) Also add the library in the Project



- 5) Now the code needs to be compiled
- 6) Open 3 cmds one for server and one for client, refer output section.

DATABASE CREATION: -

create database Electric_Bill;

```
mysql> create database Electric_Bill;  
Query OK, 1 row affected (0.01 sec)
```

show databases;

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| electric_bill |  
| library |  
| mysql |  
| performance_schema |  
| sakila |  
| sys |  
| world |  
+-----+  
8 rows in set (0.01 sec)
```

use electric_bill;

```
mysql> use electric_bill;  
Database changed
```

```
create table Bill(  
  bill_id int not null auto_increment,  
  consumer_name varchar(55) not null,  
  bill_due_date varchar(55) not null,  
  bill_amount int not null,  
  PRIMARY KEY(bill_id)  
);
```

```
mysql> create table Bill(
-> bill_id int not null auto_increment,
-> consumer_name varchar(55) not null,
-> bill_due_date varchar(55) not null,
-> bill_amount int not null,
-> PRIMARY KEY(bill_id)
-> );
Query OK, 0 rows affected (0.08 sec)
```

```
INSERT INTO BILL(consumer_name, bill_due_date, bill_amount) values("JOHN THOMAS"
,"20-08-2022",3000);
```

```
INSERT INTO BILL(consumer_name, bill_due_date, bill_amount) values("RAJESH TIWARI"
,"10-10-2022",3500);
```

```
INSERT INTO BILL(consumer_name, bill_due_date, bill_amount) values("RAJIV SHEIKH"
,"15-06-2022",5500);
```

```
mysql> INSERT INTO BILL(consumer_name, bill_due_date, bill_amount) values("JOHN THOMAS" ,"20-08-2022",3000);
Query OK, 1 row affected (0.02 sec)
```

```
mysql> INSERT INTO BILL(consumer_name, bill_due_date, bill_amount) values("RAJESH TIWARI" ,"10-10-2022",3500);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> INSERT INTO BILL(consumer_name, bill_due_date, bill_amount) values("RAJIV SHEIKH" ,"15-06-2022",5500);
Query OK, 1 row affected (0.00 sec)
```

```
select * from bill;
```

```
mysql> select * from bill;
```

bill_id	consumer_name	bill_due_date	bill_amount
1	JOHN THOMAS	20-08-2022	3000
2	RAJESH TIWARI	10-10-2022	3500
3	RAJIV SHEIKH	15-06-2022	5500

```
3 rows in set (0.00 sec)
```

CODE: -

IDB.java

```
import java.rmi.*;

public interface IDB extends Remote {
    public String getData(int id) throws RemoteException;
}
```

DBImpl.java

```
/**
 *
 * @author Philip
 */

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;

public class DBImpl extends UnicastRemoteObject implements IDB {
    String str, str1;
    public DBImpl() throws RemoteException {
    }
    @Override
    public String getData(int id) throws RemoteException {
        String URL = "jdbc:mysql://localhost:3306/Electric_Bill";
        String UName = "root";
        String Pass = "password";
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection(URL, UName, Pass);
            PreparedStatement ps = con.prepareStatement("select * from bill where bill_id=?");
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            ResultSetMetaData rsmd = rs.getMetaData();
            str = "";
            str1 = "";
            for (int i = 1; i <= rsmd.getColumnCount(); i++) {
```

```

str1 = str1 + rsmd.getColumnName(i) + "\t";
}
System.out.println();
while (rs.next()) {
for (int i = 1; i <= rsmd.getColumnCount(); i++) {
str = str + rs.getString(i) + "\t";
}
str = str + "\n";
}
} catch (Exception e) {
e.printStackTrace();
}
return (str1 + "\n" + str);
}
}

```

Server.java

```

/**
 *
 * @author Philip
 */
import java.rmi.Naming;
public class Server {
    public static void main(String[] args) {
try {
DBImpl di = new DBImpl();
Naming.rebind("rmi://127.0.0.1/DBServerBill", di);
System.out.println("Server Registered.");
} catch (Exception e1) {
e1.printStackTrace();
}
}
}
}

```

Client.java

```

/**
 *
 * @author Philip
 */
import java.io.BufferedReader;

```

```

import java.io.InputStreamReader;
import java.rmi.Naming;
public class Client {
public static void main(String[] args) {
String res = "";
try {
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String url = "rmi://127.0.0.1/DBServerBill";
System.out.println("Retriving Bill Information....");
while (true) {
System.out.print("Enter choice:\n1. Get Bill By Id ");
BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));
IDB id1 = (IDB) Naming.lookup(url);
res = id1.getData(Integer.parseInt(br1.readLine()));
System.out.println(res);
}
} catch (Exception e) {
e.printStackTrace();
}
}
}

```

OUTPUT: -

```

C:\Users\Philip\OneDrive\Documents\NetBeansProjects\newdsc6b\src>javac *.java
C:\Users\Philip\OneDrive\Documents\NetBeansProjects\newdsc6b\src>

```

After executing the above command in cmd run the file in Netbeans by right clicking the file

Server.java

```

run:
Server Registered.

```


Client.java

```
run:
Retriving Bill Information....
Enter choice:
1. Get Bill By Id 1
bill_id consumer_name    bill_due_date    bill_amount
1      JOHN THOMAS      20-08-2022      3000

Enter choice:
1. Get Bill By Id 2
bill_id consumer_name    bill_due_date    bill_amount
2      RAJESH TIWARI    10-10-2022      3500

Enter choice:
1. Get Bill By Id 3
bill_id consumer_name    bill_due_date    bill_amount
3      RAJIV SHEIKH     15-06-2022      5500

Enter choice:
1. Get Bill By Id |
```

CONCLUSION: -

From this practical I have learned to implement Remote Method Communication using JDBC and RMI.

PRACTICAL 7

AIM: - Implementation of mutual exclusion using the Token ring algorithm.

THEORY: -

In computer science, mutual exclusion is a property of concurrency control, which is instituted for the purpose of preventing race conditions. It is the requirement that one thread of execution never enters a critical section while a concurrent thread of execution is already accessing said critical section, which refers to an interval of time during which a thread of execution accesses a shared resource or shared memory.

The shared resource is a data object, which two or more concurrent threads are trying to modify (where two concurrent read operations are permitted but, no two concurrent write operations or one read and one write are permitted, since it leads to data inconsistency). Mutual exclusion algorithm ensures that if a process is already performing write operation on a data object [critical section] no other process/thread is allowed to access/modify the same object until the first process has finished writing upon the data object [critical section] and released the object for other processes to read and write upon.

The problem which mutual exclusion addresses is a problem of resource sharing: how can a software system control multiple processes' access to a shared resource, when each process needs exclusive control of that resource while doing its work? The mutual-exclusion solution to this makes the shared resource available only while the process is in a specific code segment called the critical section. It controls access to the shared resource by controlling each mutual execution of that part of its program where the resource would be used.

A successful solution to this problem must have at least these two properties:

- It must implement mutual exclusion: only one process can be in the critical section at a time.
- It must be free of deadlocks: if processes are trying to enter the critical section, one of them must eventually be able to do so successfully, provided no process stays in the critical section permanently.

Deadlock freedom can be expanded to implement one or both of these properties:

- Lockout-freedom guarantees that any process wishing to enter the critical section will be able to do so eventually. This is distinct from deadlock avoidance, which requires that some waiting process be able to get access to the critical section, but does not require that every process gets a turn. If two processes continually trade a resource between them, a third process could be locked out and experience resource starvation, even though the system is not in deadlock. If a system is free of lockouts, it ensures that every process can get a turn at some point in the future.
- A k-bounded waiting property gives a more precise commitment than lockout-freedom. Lockout-freedom ensures every process can access the critical section eventually: it gives no guarantee about how long the wait will be. In practice, a process could be overtaken an arbitrary or unbounded number of times by other higher-priority processes before it gets its turn. Under a k-bounded waiting property, each process has a finite maximum wait time. This works by setting a limit to the number of times other processes can cut in line, so that no process can enter the critical section more than k times while another is waiting

.Every process's program can be partitioned into four sections, resulting in four states. Program execution cycles through these four states in order the cycle of sections of a single process

Non-Critical Section

Operation is outside the critical section; the process is not using or requesting the shared resource.

Trying

The process attempts to enter the critical section.

Critical Section

The process is allowed to access the shared resource in this section.

Exit

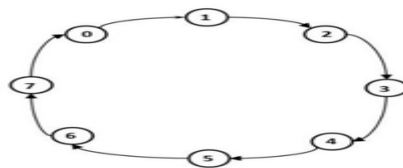
The process leaves the critical section and makes the shared resource available to other processes.

If a process wishes to enter the critical section, it must first execute the trying section and wait until it acquires access to the critical section. After the process has executed its critical section and is finished with the shared resources, it needs to execute the exit section to release them for other processes' use. The process then returns to its non-critical section.

TOKEN RING ALGORITHM: -

- In this algorithm it is assumed that all the processes in the system are organized in a logical ring. The figure blow describes the structure.

- The ring positions may be allocated in numerical order of network addresses and is unidirectional in the sense that all messages are passed only in clockwise or anti-clockwise direction.
- When a process sends a request message to current coordinator and does not receive a reply within a fixed timeout, it assumes the coordinator has crashed. It then initializes the ring and process P_i is given a token.
- The token circulates around the ring. It is passed from process k to $k+1$ in point to point messages. When a process acquires the token from its neighbor it checks to see if it is attempting to enter a critical region. If so the process enters the region does all the execution and leaves the region. After it has exited it passes the token along the ring. It is not permitted to enter a second critical region using the same token.
- If a process is handed the token by its neighbor and is not interested in entering a critical region it just passes along. When no processes want to enter any critical regions the token just circulates at high speed around the ring.
- Only one process has the token at any instant so only one process can actually be in a critical region. Since the token circulates among the process in a well-defined order, starvation cannot occur.
- Once a process decides it wants to enter a critical region, at worst it will have to wait for every other process to enter and leave one critical region.
- The disadvantage is that if the token is lost it must be regenerated. But the detection of lost token is difficult. If the token is not received for a long time it might not be lost but is in use.



CODE: -

TokenServer.java

```

import java.net.*;
import java.io.*;
class TokenServer

```

```

{
public static DatagramSocket ds;
public static DatagramPacket dp;
public static void main(String[] args)throws Exception
{
try
{
ds=new DatagramSocket(1000);
}
catch(Exception e)
{
e.printStackTrace();
}
while(true)
{
byte buff[]=new byte[1024];
ds.receive(dp=new DatagramPacket(buff,buff.length));
String str=new String(dp.getData(),0,dp.getLength());
System.out.println("Message from " +str);
}
}
}

```

TokenClient1.java

```

import java.net.*;
import java.io.*;
class TokenClient1
{
public static DatagramSocket ds;
public static DatagramPacket dp;
public static BufferedReader br;
static int cp=100;
public static void main(String[] args)throws Exception
{
boolean hasToken;

try
{
ds=new DatagramSocket(100);
}

```

```

catch(Exception e)
{
e.printStackTrace();
}
hasToken=true;
while(true)
{
if(hasToken==true)
{
System.out.println("Do you want to enter data...(yes/no):");
br=new BufferedReader(new InputStreamReader(System.in));
String ans=br.readLine();
if(ans.equalsIgnoreCase("yes"))
{
System.out.println("ready to send");
System.out.println("sending");
System.out.println("Enter the data");
br=new BufferedReader(new
InputStreamReader(System.in));
String str="Client-1====> "+br.readLine();
byte buff[]=new byte[1024];
buff=str.getBytes();
ds.send(new
DatagramPacket(buff,buff.length,InetAddress.getLocalHost(),1000));
System.out.println("now sending");
}
else if(ans.equalsIgnoreCase("no"))
{
System.out.println("I am busy state");
//sending msg to client-2
String msg="Token";
byte bf1[]=new byte[1024];
bf1=msg.getBytes();
ds.send(new
DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),200));
hasToken=false;
//receivngmsg from client-2
byte bf2[]=new byte[1024];
ds.receive(dp=new
DatagramPacket(bf2,bf2.length));

```

```

String clientmsg=new
String(dp.getData(),0,dp.getLength());
System.out.println("The data is "+clientmsg);

if(clientmsg.equals("Token"))
hasToken=true;
System.out.println("I am leaving busy state");
}
}
else
{
System.out.println("Entering in receive mode.");
byte bf[]=new byte[1024];
ds.receive(dp=new DatagramPacket(bf,bf.length));
String clientmsg1=new String(dp.getData(),0,dp.getLength());
System.out.println("The data is "+clientmsg1);
if(clientmsg1.equals("Token"));
{
hasToken=true;
}
}
}
}
}
}
}

```

TokenClient2.java

```

import java.net.*;
import java.io.*;
class TokenClient2
{
static DatagramSocket ds;
static DatagramPacket dp;
static BufferedReader br;
public static void main(String[] args)throws Exception
{
try
{
ds=new DatagramSocket(200);
}
catch(Exception e)

```

```

{
e.printStackTrace();
}
boolean hasToken=true;
while(true)
{
//System.out.println("Entering if");
if(hasToken==true)
{
System.out.println("Do you want to enter data(Yes/No):");
br=new BufferedReader(new InputStreamReader(System.in));
String str=br.readLine();
if(str.equalsIgnoreCase("yes"))
{
System.out.println("Enter Data; ");
br=new BufferedReader(new
InputStreamReader(System.in));
String msg="Client-2==>"+br.readLine();
byte bf1[]=new byte[1024];
bf1=msg.getBytes();
ds.send(new
DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),1000));
System.out.println("Data sent");
}
else
{
//send to client 1.
String clientmsg="Token";
byte bf2[] = new byte[1024];
bf2=clientmsg.getBytes();
ds.send(new
DatagramPacket(bf2,bf2.length,InetAddress.getLocalHost(),100));
hasToken=false;
}
}
else
{
try
{
byte buff[]=new byte[1024];

```



```

System.out.println("Entering in receiving mode.");
ds.receive(dp=new DatagramPacket(buff,buff.length));
String clientmsg1=new
String(dp.getData(),0,dp.getLength());
System.out.println("The data is "+clientmsg1);
if(clientmsg1.equals("Token"))
hasToken=true;
}
catch(Exception e)
{
e.printStackTrace();
}
}
}
}
}
}
}
}

```

OUTPUT: -

Server

```

C:\Users\User\Documents\NetBeansProjects\dsc\src>javac *.java

C:\Users\User\Documents\NetBeansProjects\dsc\src>java TokenServer
Message from Client-1==> How are you?
Message from Client-2==>I am fine

```

TokenClient1

```

C:\Users\User\Documents\NetBeansProjects\dsc\src>java TokenClient1
Do you want to enter data...(yes/no):
yes
ready to send
sending
Enter the data
How are you?
now sending
Do you want to enter data...(yes/no):

```

TokenClient2

```

C:\Users\User\Documents\NetBeansProjects\dsc\src>java TokenClient2
Do you want to enter data(Yes/No):
yes
Enter Data;
I am fine
Data sent
Do you want to enter data(Yes/No):

```

CONCLUSION: -

From this practical I have learned the Implementation of mutual exclusion using the Token ring algorithm.

PRACTICAL 8

AIM: - Implementation of Storage as a Service using Google Docs.

THEORY: -

Google Drive:

Google Drive is a cloud-based storage solution that allows you to save files online and access them anywhere from any smartphone, tablet, or computer. Using a cloud storage service like Google Drive has plenty of advantages, such as easier file sharing and having a remote location to back up your files. But when compared to competitors like DropBox and Apple's iCloud service, Google Drive's popularity is built on useful collaborative tools and built-in integrations with Google's suite of products and services. If you have a Google account, you already have 15 GB of free storage on Google Drive. So how to take advantage of all that space? Our guide covers all the basics, from how to use Google Drive to upload and access files on any device, to all the tools that make collaboration with others a breeze.

Benefits of Google drive:

1) Ability to Access Files from Everywhere: -

The most significant advantage of using Google Drive to store your data is that you can access it anywhere. Google servers allow you to sign in using your Gmail account and access your data anytime. What you need is an internet connection and a device that can go online. This ensures you never lose your files and important data even if your computer, smartphone, or other gadget is stolen or damaged.

2) Ability to edit files: -

The ability to edit files in Google Drive is an added advantage that you will not find with other data storage options. You get instant access to various suite editing tools, including Google Spreadsheets, Google Docs and Google Slides and many others. The access to edit your files allows making any changes you wish to before saving them again.

3) Compatibility with most devices: -

You are not limited to using a single device to access Google Drive. You can use different gadgets like an Android phone, iPad, Mac, iPhone or a PC to access your stored data anytime from anywhere. This makes Google Drive beneficial and highly efficient.

4) Quick Files Search: -

Searching a specific file in Google Drive is made easy by the search feature that allows you to filter the file type and use keywords to find the file. Even a word on an image included in your file can help you locate it fast. With the ability to store thousands of documents, it may be impossible to find a file by scanning through them all. Thanks to the quick file search options, you can locate your file within seconds.

5) Ability to view different file types: -

This is a crucial pro of Google Drive. You do not have to install different file types to view documents on your device; Google can open many types of files, including Adobe, HD video, Photoshop, and Illustrator.

6) Easy sharing: -

It takes a few clicks to share files and data with other people. This makes it easy for people to work as a group. You need to set up permissions to allow your collaborators to edit, view or comment on specific files. It is a great feature that also ensures Google Drive is safe because only those permitted can access your files.

7) Open discussion: -

If you have a project that requires participants to carry out discussions, Google Drive makes it possible. It has a comments feature where users can make comments or reply to others and get feedback instantly. A group of people can work on a project without the need for other forms of communication e, thanks to the comments feature that enables them to collaborate smoothly and achieve their common goal.

8) Free Storage space of up to 15 GB: -

When you sign for a Google account, you get 15 GB of storage space for free. It is shared across Google Photos, Gmail and Google Drive. If you have plenty of files and data to store, you can upgrade at the cost of \$2.49 for 25 GB, \$4.99 for 100 GB or \$49.99 for 1 TB. The plans are monthly, and choose what suits you depending on your needs.

9) Excellent User interface: -

Easy to navigate interface is another great advantage of Google Drive. Even first-time users do not get lost because this online storage option is easy to manage. You can manoeuvre through the files and data and locate whatever you are looking for very fast. The user interface is friendly and made for anyone who wishes to store their information using cloud computing.

10) Affordability and great usability: -

Google Drive comes with a wide range of features that encourages everyone, including new users, to browse through it effortlessly. It is free, and you have up to 15 GB of storage space. This makes it affordable and great for users who do not have much to store and looking for free storage.

Steps to create Google drive account:

1. Search google drive in chrome browser and visit the first link.
2. Select individuals and then go to drive.
3. Next, Create new account for myself
4. Fill all the necessary personal details.
5. Choose any username and password for the account.
6. Once your account is created, we can access all free services of google drive.

Steps to create Google Doc:

1. To create a new doc, click on new and select google doc.
2. Select a new blank document.
3. Type text, add images, add header and many more features are available in google doc.
4. We can download doc in any format.
5. Share doc is also available in google doc

6. All docs automatically save into our google drive folder we can access it anytime and anywhere.

Steps to create Google Slides:

1. To create a new slide,click on new and select Google Slides.
2. Select Blank presentation.
3. Choose anytheme, type text and manyfeatures are available in Google slides.
4. All slides are saved in my drive and we can access it anywhere and anytime.

CONCLUSION: -

From this practical I have learned the implementation of Storage as a Service using Google Docs.

PRACTICAL 9

Application using Google App Engine

AIM: - To develop applications using Google App Engine by using Eclipse IDE. Find the position of a target value within a sorted integer array.(Binary Search)

THEORY: -

Platform as a Service – PaaS:

Platform as a Service (PaaS) provides a runtime environment. It allows programmers to easily create, test, run, and deploy web applications. PaaS specifically provides a platform for customers to develop, run, and manage applications without building and maintaining the cloud infrastructure required to develop and launch an app.

PaaS can be delivered in three different formats.

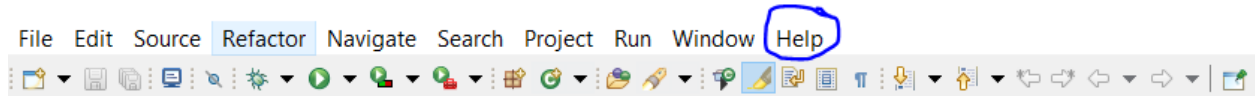
- First is as a cloud service from the provider. In this configuration, the customer controls software deployment with minimal configuration options. The Platform as a Service provider supplies the networking, servers, storage, operating system (OS), middleware (e.g., Java runtime, .NET runtime, integration, etc.), database and other services to host the consumer's application.
- The second PaaS configuration can be run as a private service (software or appliance) behind a firewall.
- The third PaaS configuration can be run as software deployed on public infrastructure as a service such as AWS.

Google App Engine:

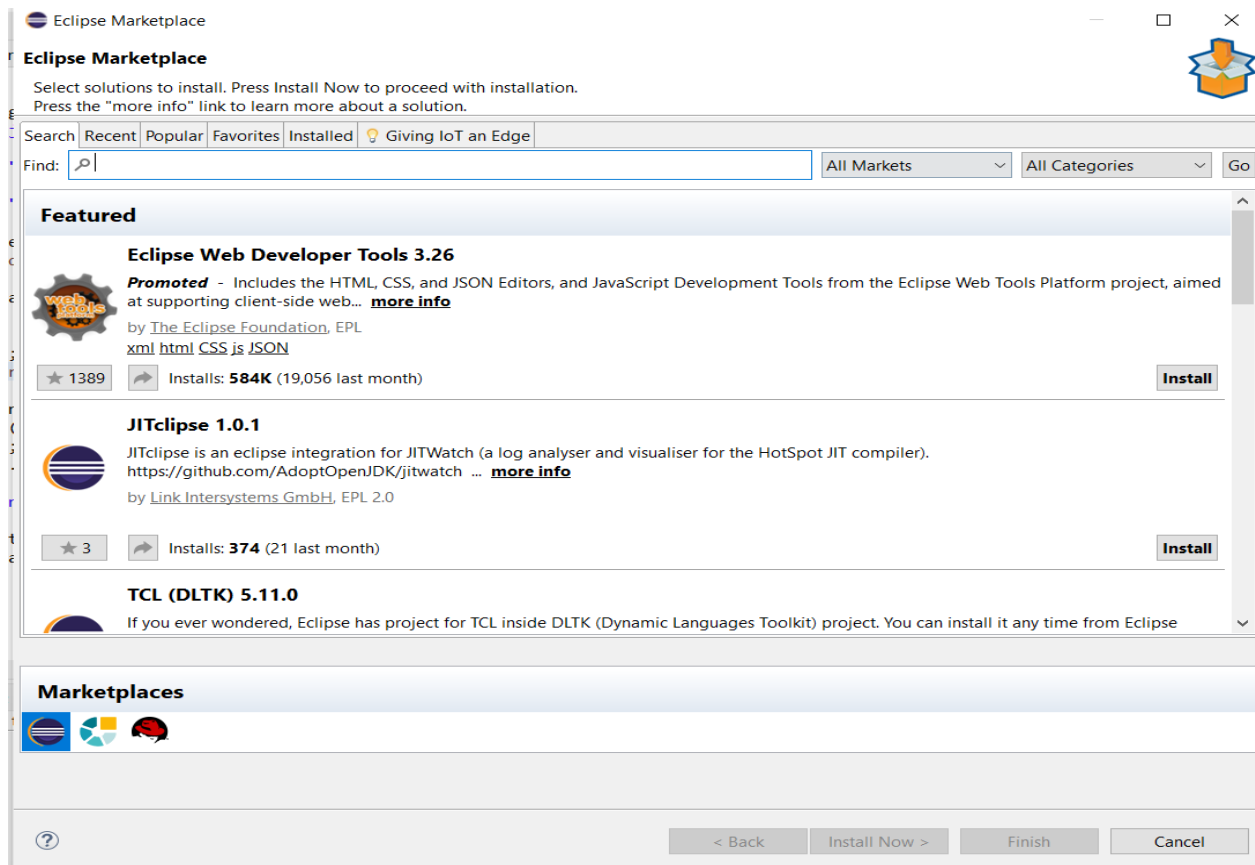
Google App Engine is a Platform as a Service (PaaS) product that provides Web app developers and enterprises with access to Google's scalable hosting and tier 1 Internet service. The App Engine requires that apps be written in Java or Python, store data in Google BigTable and use the Google query language. Non-compliant applications require modification to use App Engine. Google App Engine provides more infrastructure than other scalable hosting services such as Amazon Elastic Compute Cloud (EC2). The App Engine also eliminates some system administration and developmental tasks to make it easier to write scalable applications. Google App Engine is free up to a certain amount of resource usage. Users exceeding the per-day or per-minute usage rates for CPU resources, storage, number of API calls or requests and concurrent requests can pay for more of these resources.

STEPS: -

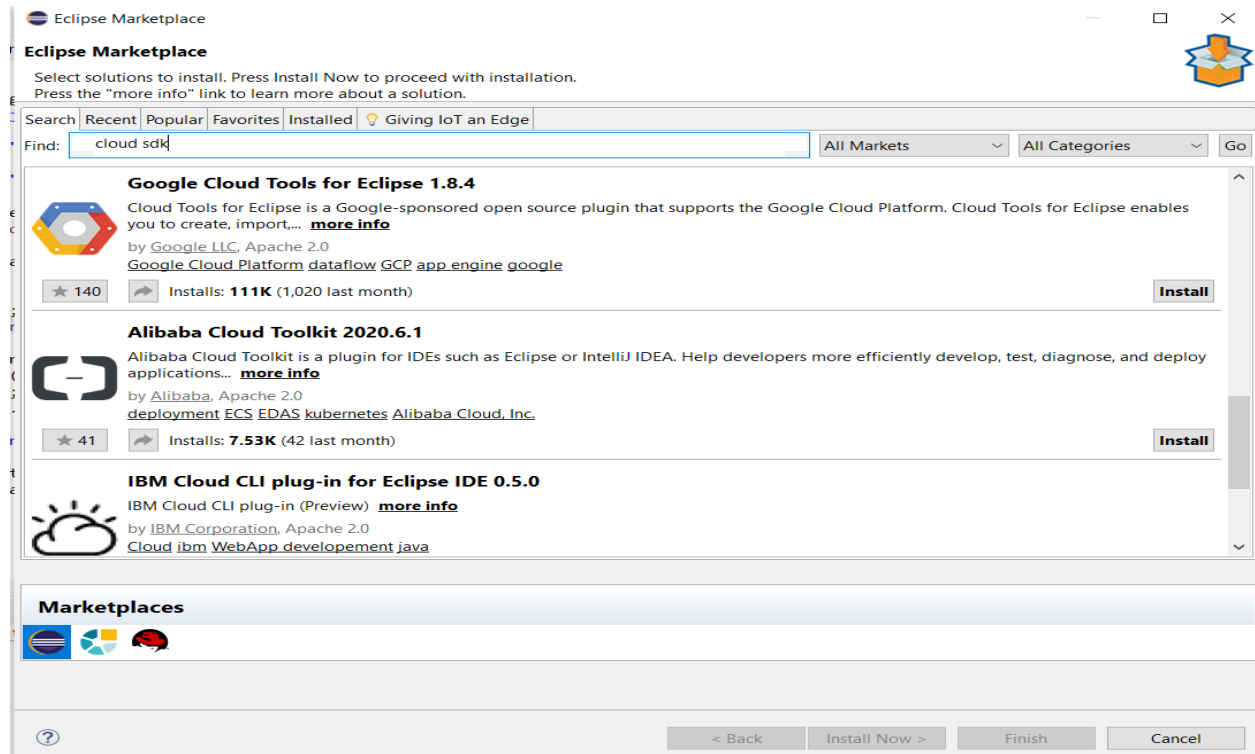
1) Open Eclipse go to Help which is circled in below image



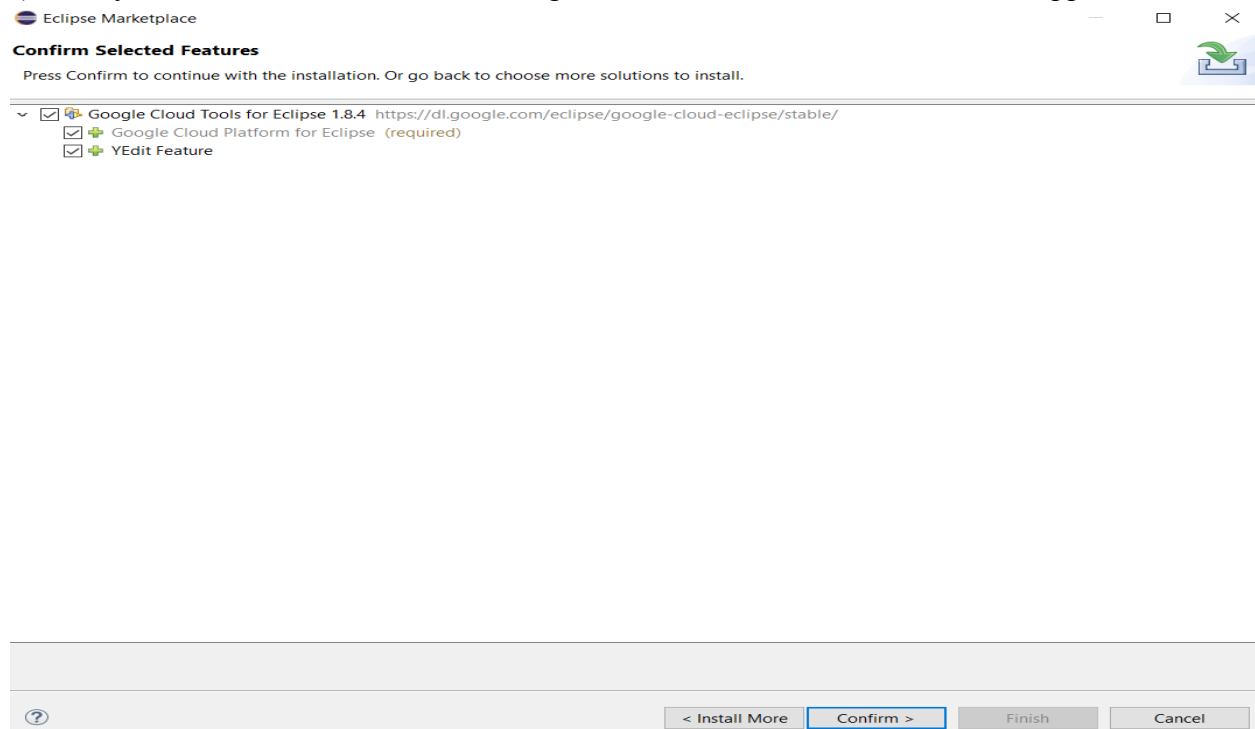
2) Choose Eclipse Marketplace below window appears



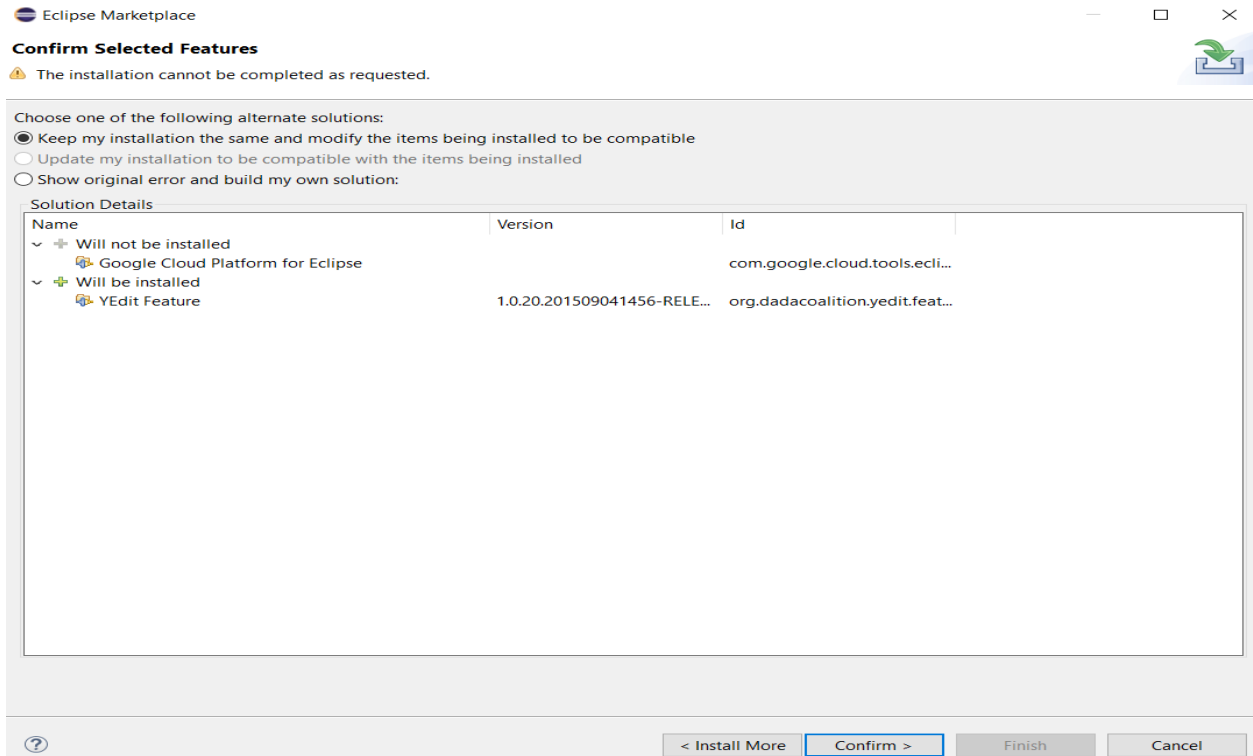
3) Search Cloud Sdk→Many option appears→Scroll and find(Google Cloud Tools for Eclipse)



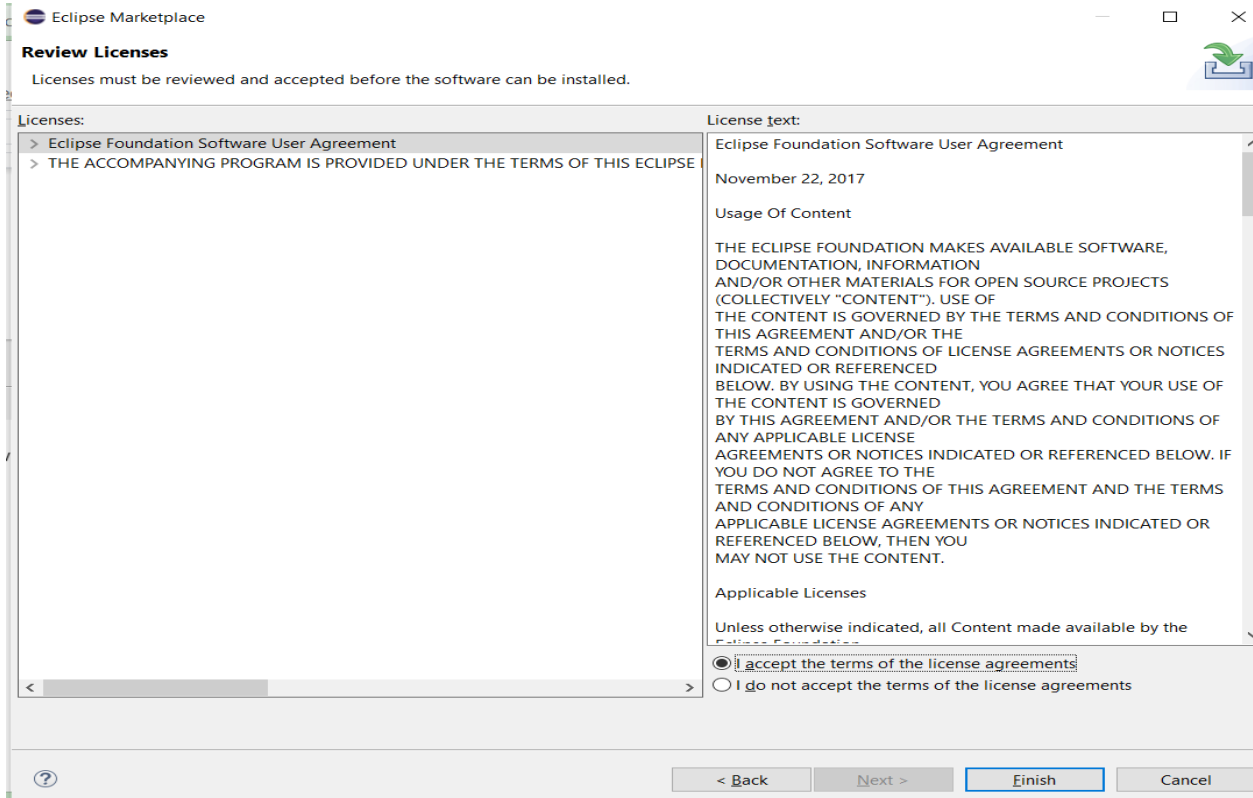
4) After you click Install Select the below given values when the below window appears



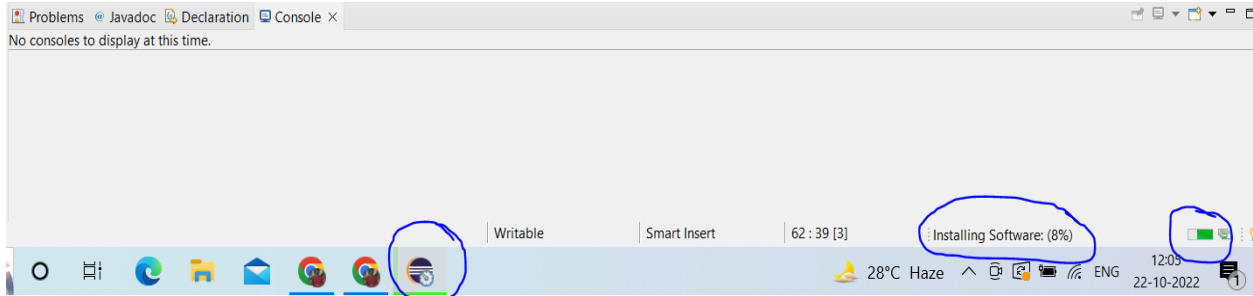
5) Click Confirm installation starts after installation below window appears



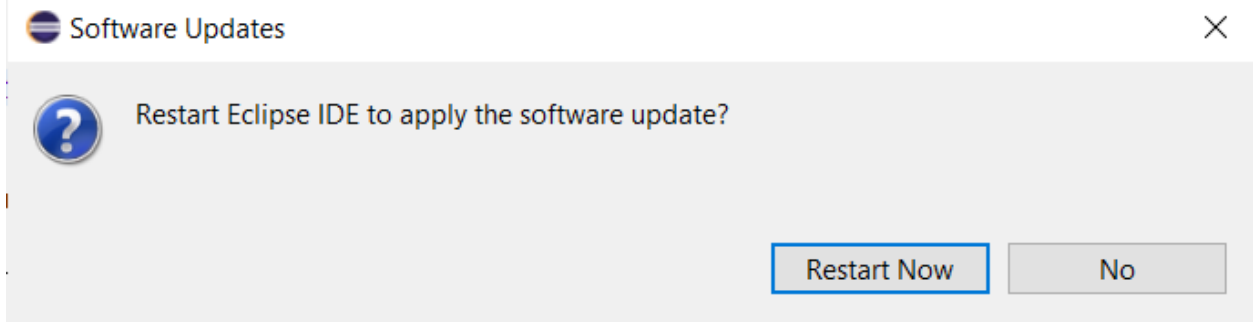
6) Click Confirm below image i.e. license window appears



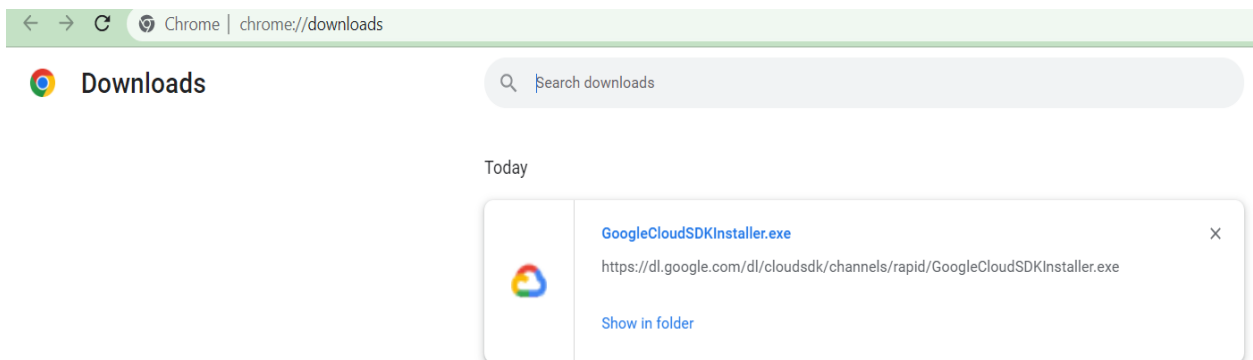
7) After accepting, choose Finish. The installation will begin as shown in below image



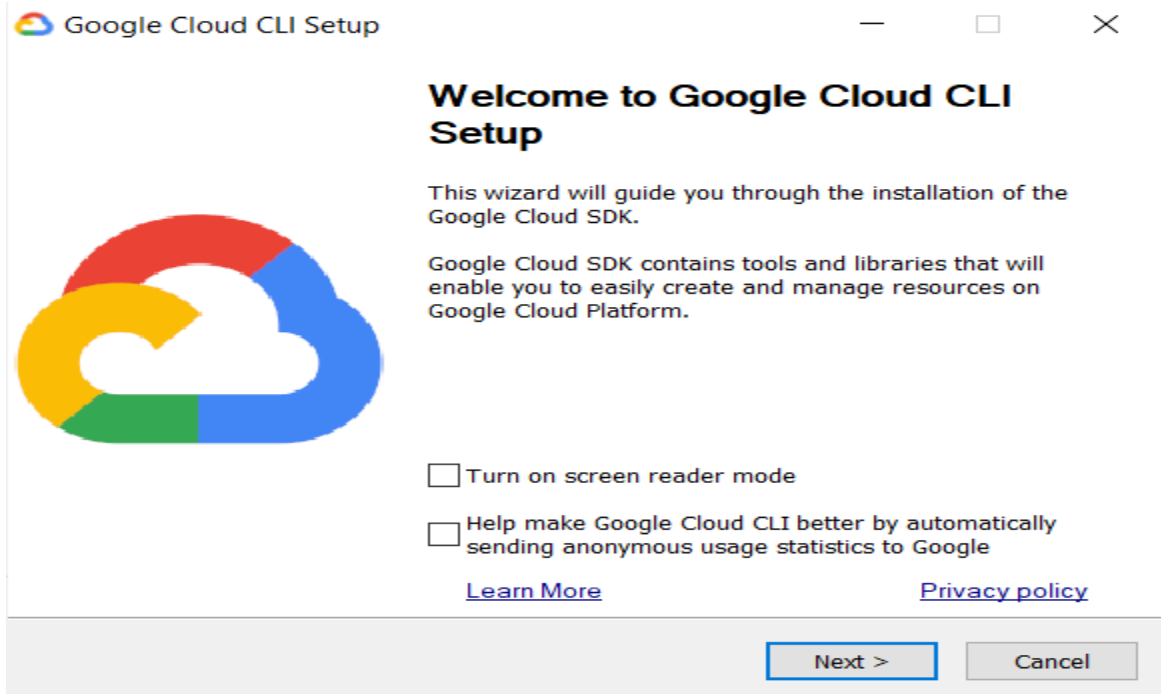
8) The Restart option will be shown after installation choose Restart Now.



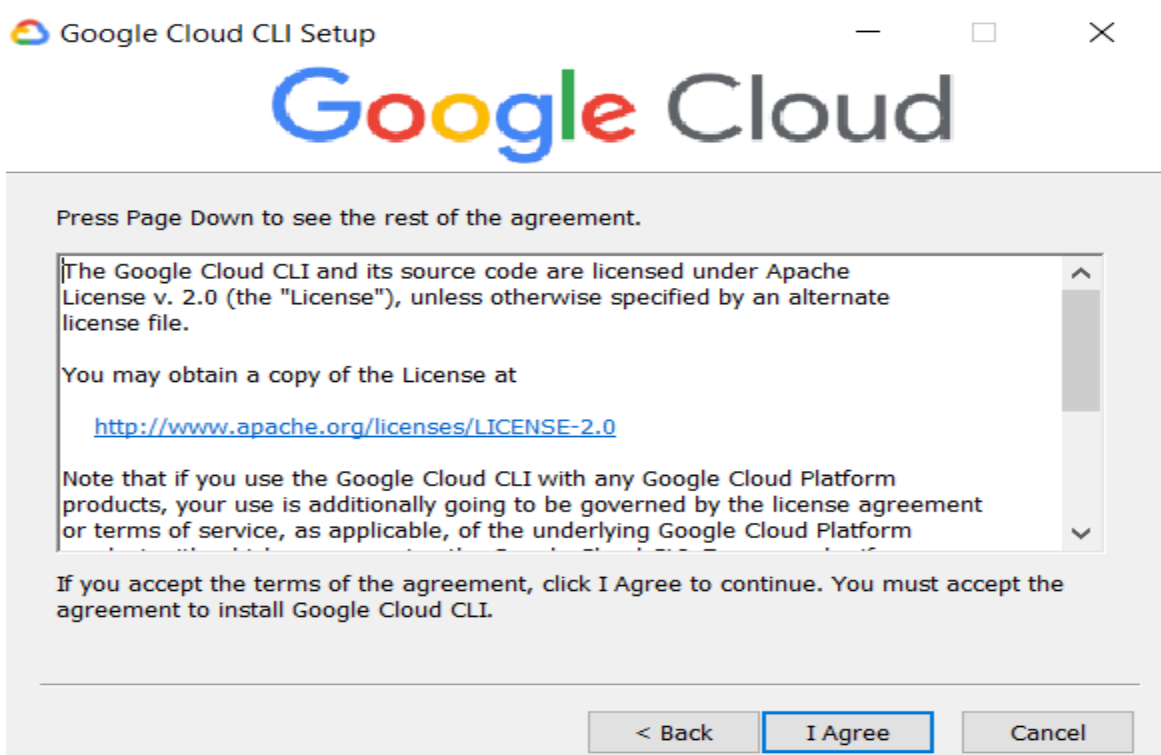
9) Now in Google Browser copy paste below link you will find the software downloaded
<https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe>



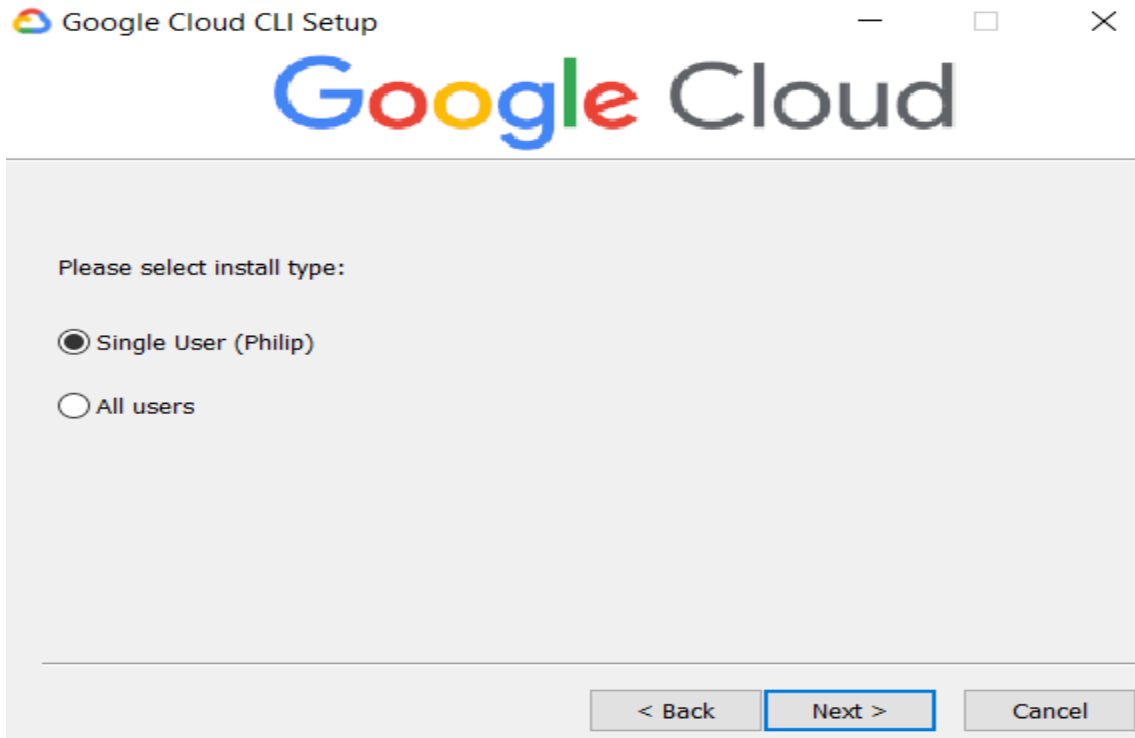
10) Click file in folder after below window appears click Next



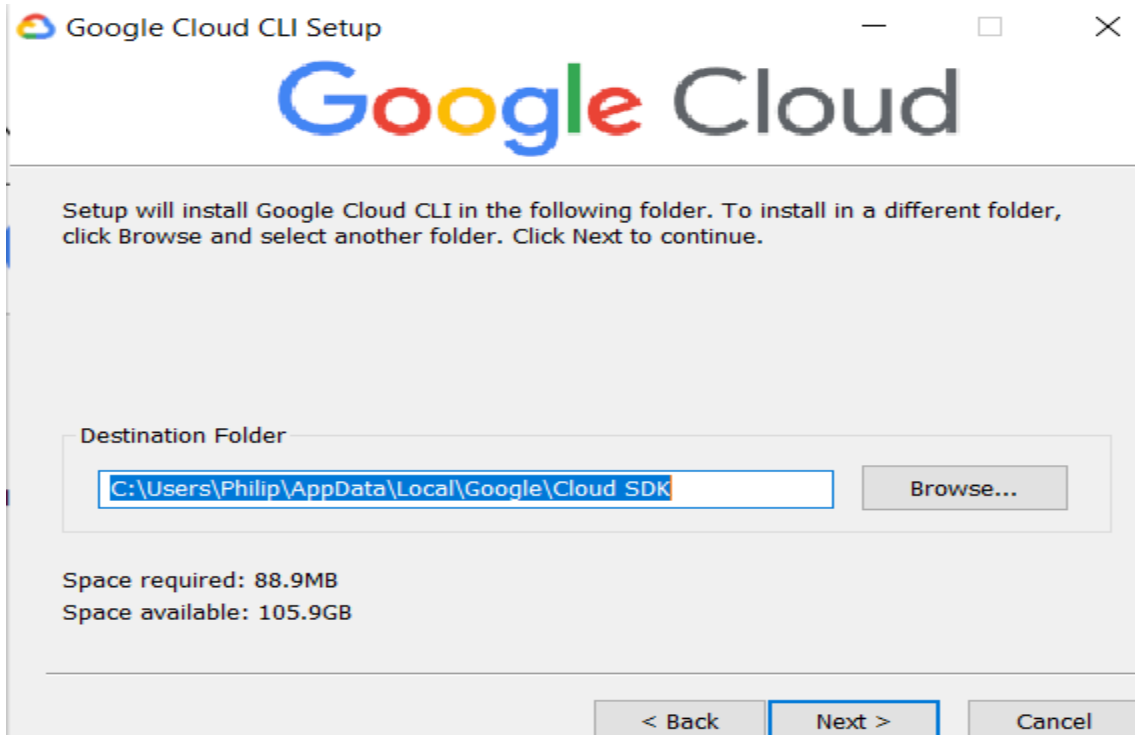
11) Choose I agree



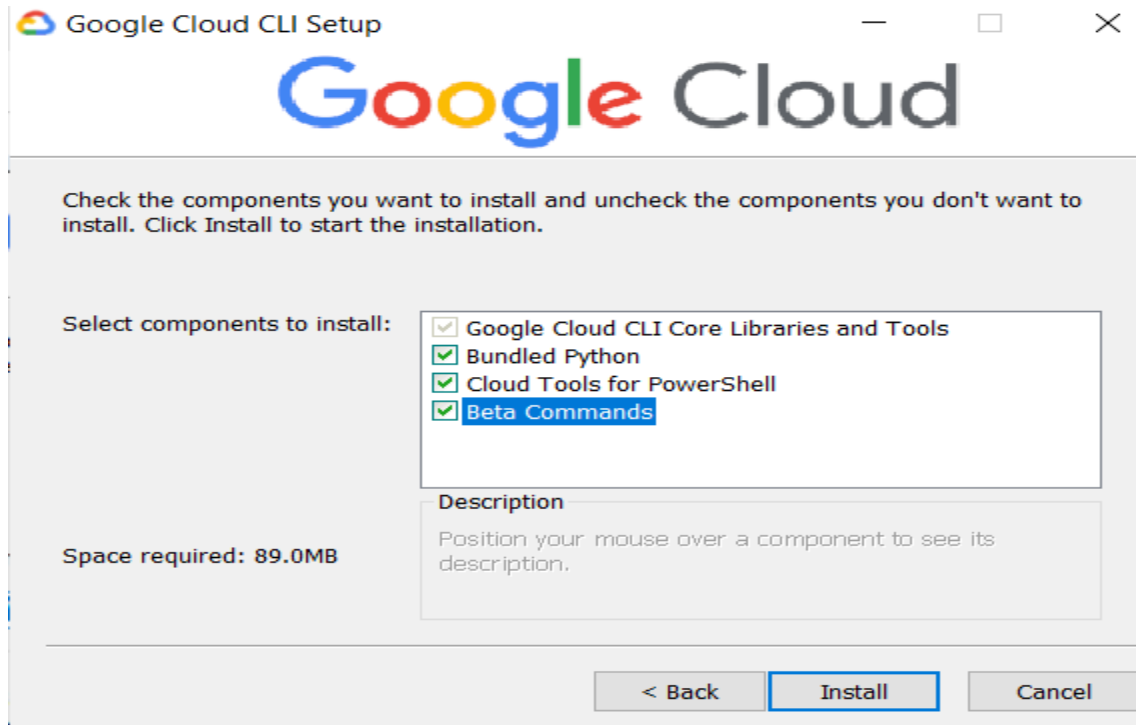
12) Choose Single User & click Next



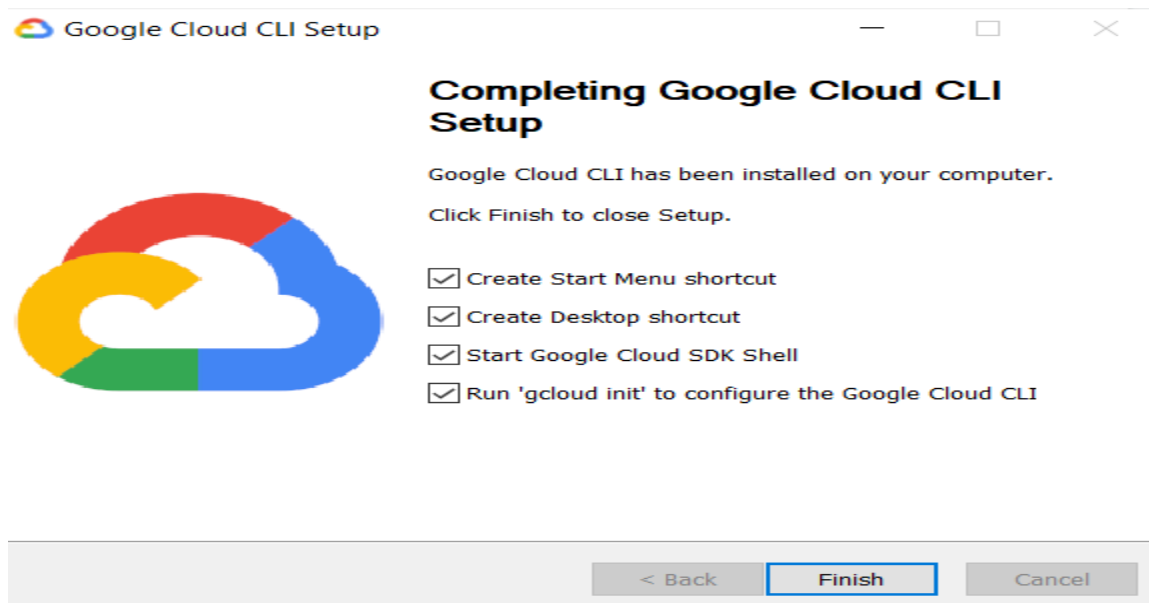
13) By default path is set you may change if you need. Choose Next



14) Choose all the options shown in below image and click on Install



15) After Installation clicking Next will goto below window, then click Finish



16) Below window opens, Type Y

```
C:\ Select C:\WINDOWS\SYSTEM32\cmd.exe - gcloud init
Welcome to the Google Cloud CLI! Run "gcloud -h" to get the list of available commands.
---
Welcome! This command will take you through the configuration of gcloud.

Your current configuration has been set to: [default]

You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

You must log in to continue. Would you like to log in (Y/n)? Y

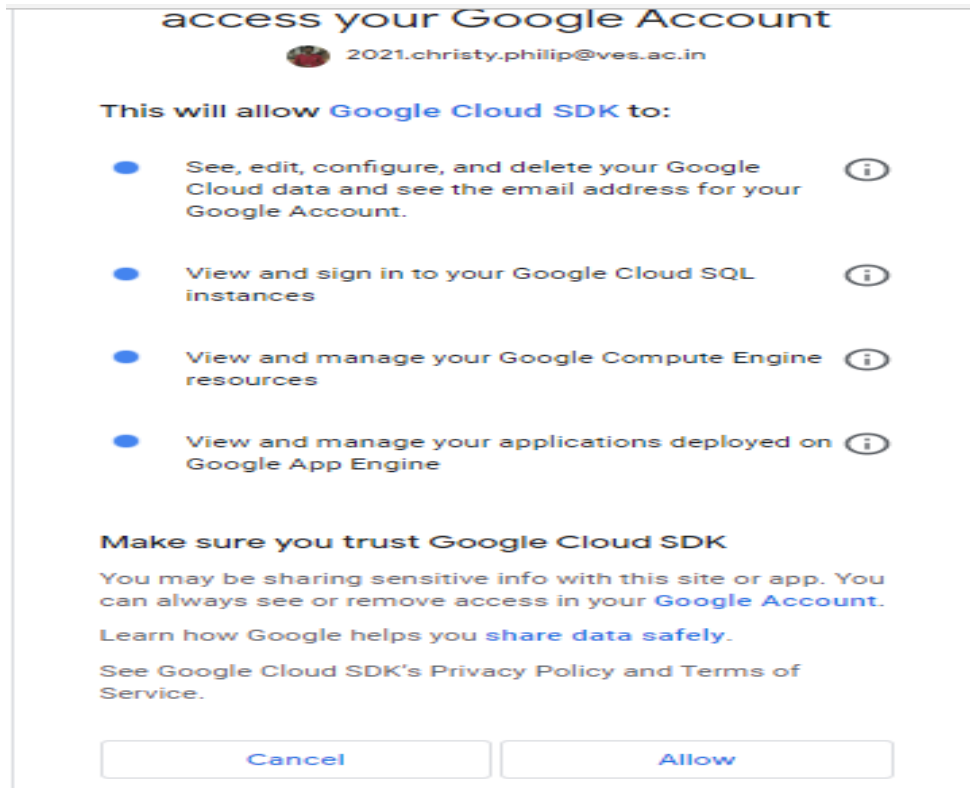
You must log in to continue. Would you like to log in (Y/n)? Y

Your browser has been opened to visit:
https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&scope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fsqlservice.login+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&state=2Png1r90JNXnawF1J9sA9gtTyCAXhc&access_type=offline&code_challenge=XEhlqMI08ppQSMEEyqsy7JdNPA5mwHDxhQj_RuCeGaY&code_challenge_method=S256

You are logged in as: [2021.christy.philip@ves.ac.in].

This account has no projects.
```

17) Google Account options open in Browser press allow



18) Close the sdk and open it again after logging in the google and write the below command

Note: - Run as administrator the shell

```
C:\Users\user\AppData\Local\Google\Cloud SDK>gcloud components install app-engine-java
```

19) New shell appears as prompt which is shown below

```
cmd.exe /c ""C:\Users\user\AppData\Local\Temp\tmps_v65s8r\python\python.exe" "-S" "
```

```
Your current Google Cloud CLI version is: 408.0.0
Installing components from version: 408.0.0
```

These components will be installed.		
Name	Version	Size
Cloud Datastore Emulator	2.3.0	35.1 MiB
gRPC Python library	1.20.0	1.5 MiB
gcloud app Java Extensions	2.0.9	64.5 MiB
gcloud app Python Extensions	1.9.101	8.6 MiB

```
For the latest full release notes, please visit:
https://cloud.google.com/sdk/release_notes

Do you want to continue (Y/n)?
```

Choose Y

Do you want to continue (Y/n)? Y

Below installation starts time will be required a bit

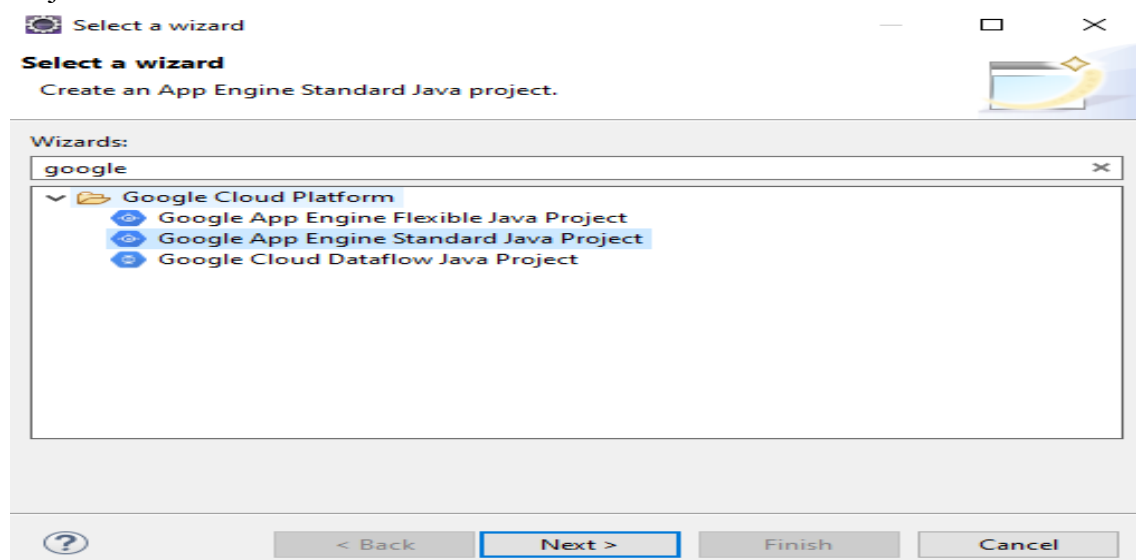
```
Do you want to continue (Y/n)? Y
#=====
#= Creating update staging area                               =#
#=====
#= Installing: Cloud Datastore Emulator                      =#
#=====
#= Installing: gRPC Python library                           =#
#=====
#= Installing: gRPC Python library                           =#
#=====
#= Installing: gcloud app Java Extensions                   =#
#=====
#= Installing: gcloud app Python Extensions                 =#
#=====
#= Creating backup and activating new installation           =#
#=====

Performing post processing steps...done.
Update done!
Press any key to continue . . .
```

20) Now check again by opening the terminal whether the components are installed.

```
C:\Users\user\AppData\Local\Google\Cloud SDK>gcloud components install app-engine-java
All components are up to date.
C:\Users\user\AppData\Local\Google\Cloud SDK>
```

21) Creating the Project in Eclipse Goto Eclipse→Choose Google App Engine Standard Java Project



22) Now name your project and let the other details be as show below

New App Engine Standard Project

App Engine Standard Project
Create a new Eclipse project for App Engine standard environment development.

Project name:

☒ Use default location
Location:

Java version:

Java package:

App Engine service:

☐ Create as Maven project

Maven project coordinates

Group ID:

Artifact ID:

Version:

23) Download jar file from below site

<http://www.java2s.com/Code/Jar/j/Downloadjavaservlet30jar.htm>

CHRISTY - prac9/src/main/java/HelloAppEngine.java - Eclipse IDE

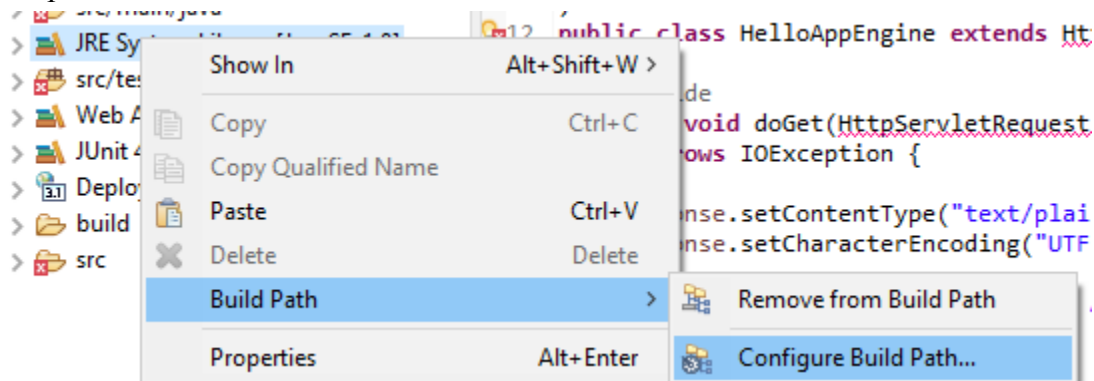
File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer: prac9

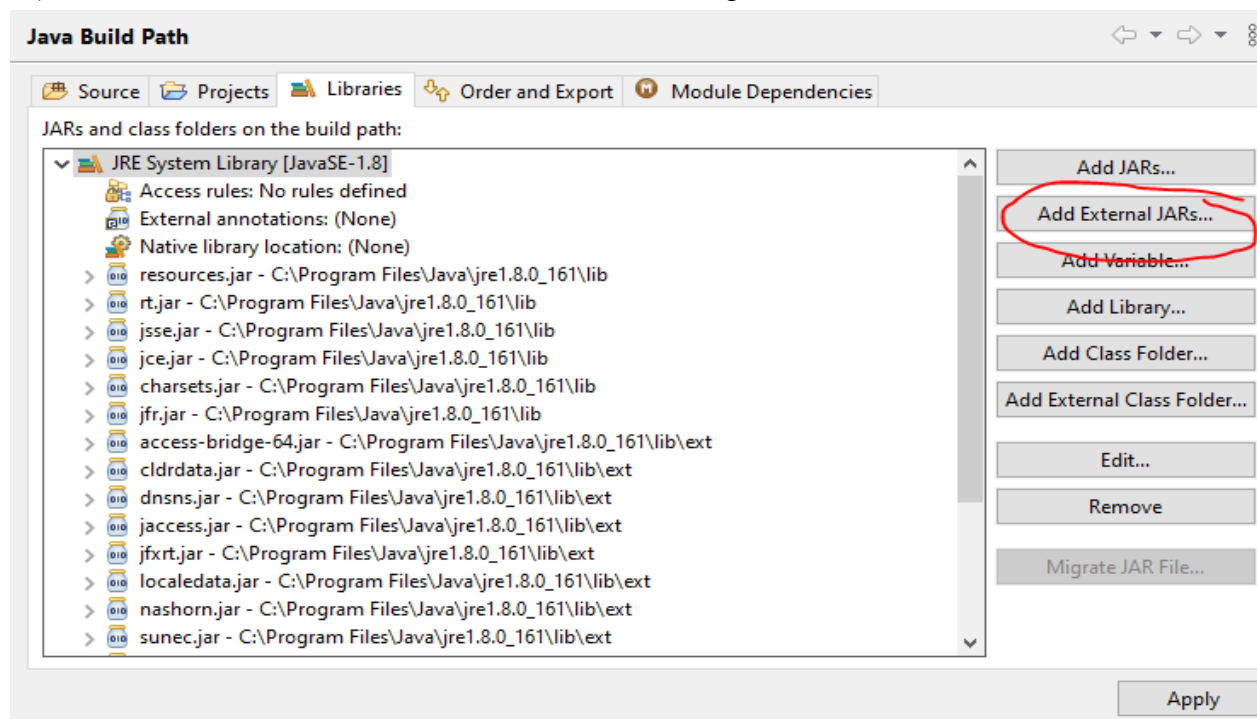
HelloAppEngine.java

```
1 import java.io.IOException;
2
3
4
5
6
7
8 @WebServlet(
9     name = "HelloAppEngine",
10    urlPatterns = {"/hello"}
11 )
12 public class HelloAppEngine extends HttpServlet {
13
14     @Override
15     public void doGet(HttpServletRequest request, HttpServletResponse response)
16         throws IOException {
17
18         response.setContentType("text/plain");
19         response.setCharacterEncoding("UTF-8");
20
21         response.getWriter().print("Hello App Engine!\r\n");
22
23     }
24 }
```

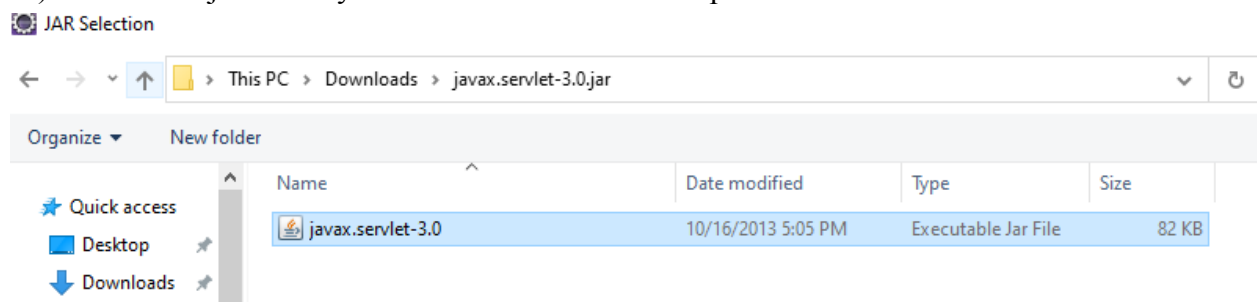
24) Expand your project in the pane and right click JRE System Library→Build Path→Configure build path



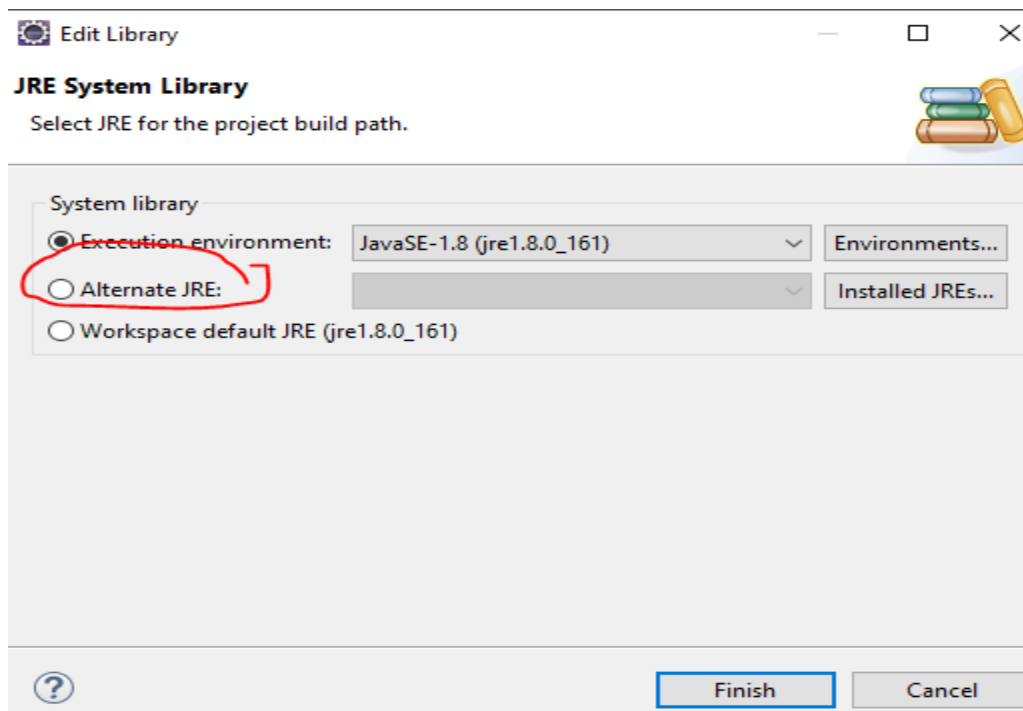
25) Choose Add External JARs which is circled below option



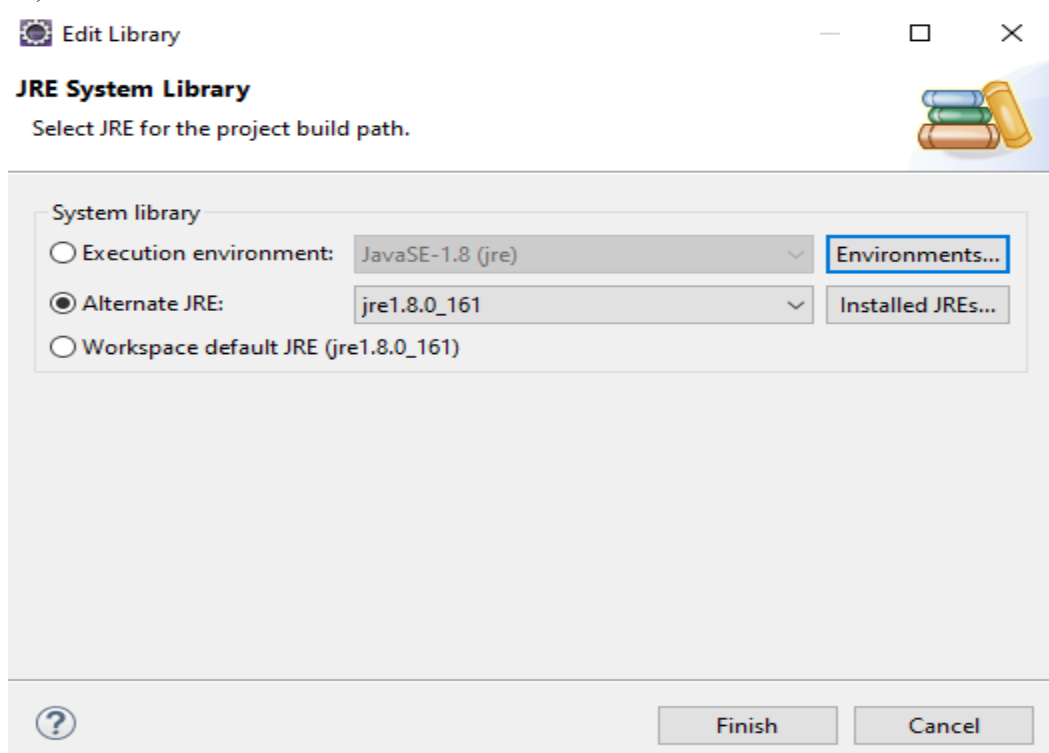
26) Browse the jar where you have downloaded and kept



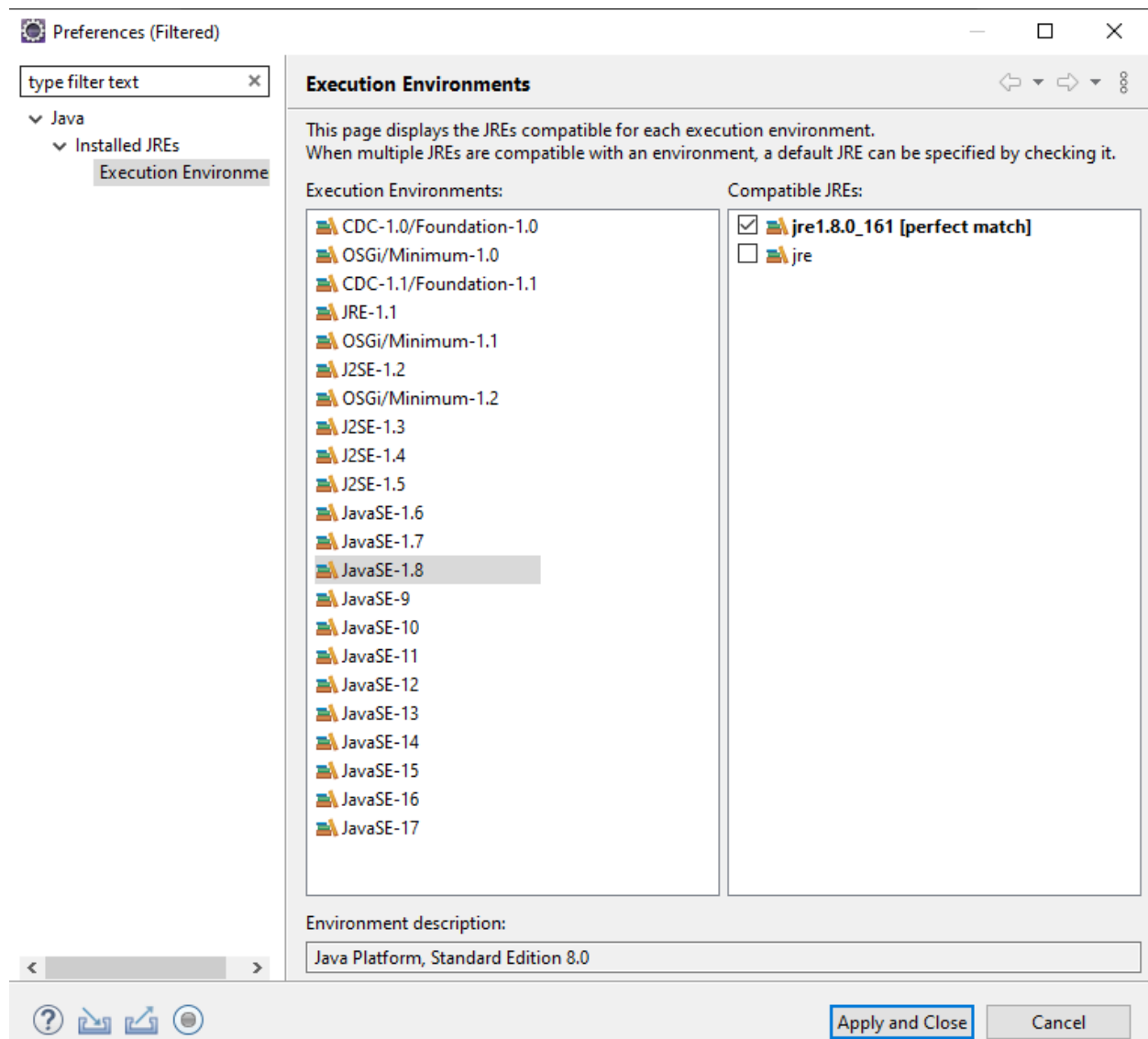
27) Choose Alternate JRE as shown below



28) After alternate JRE is set



29) Choose Environments and select which is selected in image



30) After apply and close

Right click java file and do run as google

CODE: -

index.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<meta http-equiv="content-type"
content="application/xhtml+xml; charset=UTF-8" />
```

```

<title>Hello App Engine</title>
</head>
<body>
<h1>Hello App Engine!</h1>
<table>
<tr>
<td colspan="2" style="font-weight: bold;">Available Servlets:</td>
</tr>
<tr>
<td><a href="/hello">Binary Search</a></td>
</tr>
</table>
<a>Developed by CHRISTY PHILIP</a>
</body>
</html>

```

HelloAppEngine.java

```

import java.io.IOException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(name = "HelloAppEngine", urlPatterns = { "/hello" })
public class HelloAppEngine extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
        IOException {

        response.setContentType("text/plain");
        response.setCharacterEncoding("UTF-8");
        int arr[] = { 10, 20, 30, 40, 50 };
        int key = 30;
        response.getWriter().print("Array : ");
        for (int i = 0; i < 5; i++) {
            response.getWriter().print(" " + arr[i]);
        }
        response.getWriter().println("\n\nSearch for element: " + key + "\n");
        int last = arr.length - 1;
        int first = 0;
        int mid = (first + last) / 2;

```

```

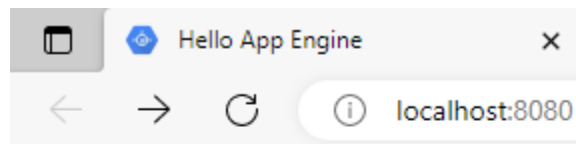
while (first <= last) {
if (arr[mid] < key) {
first = mid + 1;
} else if (arr[mid] == key) {
response.getWriter().print("Element is found at index: " + mid +

"\n");

break;
} else {
last = mid - 1;
}
mid = (first + last) / 2;
}
if (first > last) {
response.getWriter().print("Element is not found!");
}
}
}

```

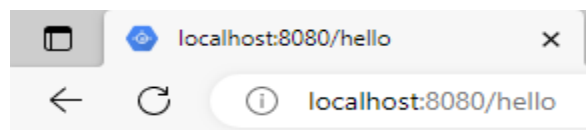
OUTPUT: -



Hello App Engine!

Available Servlets:

[The servlet](#)



Array : 10 20 30 40 50

Search for element: 30

Element is found at index: 2

App Engine Standard at localhost

```
WARNING: Could not open/create prefs root node Software\JavaSoft\Prefs at root 0x8
Nov 02, 2022 2:02:03 PM com.google.appengine.tools.development.SystemPropertiesMan
INFO: Overwriting system property key 'java.util.logging.config.file', value 'C:\U
2022-11-02 14:02:03.228:INFO::main: Logging initialized @486ms to org.eclipse.jett
2022-11-02 14:02:03.280:WARN:oejs.AbstractConnector:main: Ignoring deprecated sock
2022-11-02 14:02:03.464:INFO:oejs.Server:main: jetty-9.4.49.v20220914; built: 2022
2022-11-02 14:02:03.642:INFO:oeja.AnnotationConfiguration:main: Scanning elapsed t
2022-11-02 14:02:03.857:INFO:oejsh.ContextHandler:main: Started c.g.a.t.d.j.DevApp
2022-11-02 14:02:03.857:INFO:oejs.session:main: DefaultSessionIdManager workerName
2022-11-02 14:02:03.857:INFO:oejs.session:main: node0 Scavenging disabled
2022-11-02 14:02:03.869:INFO:oejs.AbstractConnector:main: Started NetworkTrafficSe
2022-11-02 14:02:03.872:INFO:oejs.Server:main: Started @1130ms
Nov 02, 2022 8:32:03 AM com.google.appengine.tools.development.jetty9.JettyContain
INFO: Full scan of the web app in place every 1s.
Nov 02, 2022 8:32:03 AM com.google.appengine.tools.development.AbstractModule star
INFO: Module instance default is running at http://localhost:8080/
Nov 02, 2022 8:32:03 AM com.google.appengine.tools.development.AbstractModule star
INFO: The admin console is running at http://localhost:8080/ ah/admin
Nov 02, 2022 2:02:03 PM com.google.appengine.tools.development.DevAppServerImpl do
INFO: Dev App Server is now running
```

CONCLUSION: -

From this practical I have learned to develop applications using Google App Engine by using Eclipse IDE. Find the position of a target value within a sorted integer array.(Binary Search)

PRACTICAL 10

Identity Access Management

AIM: - Description and Implementation of Identity Management in Amazon Web Services.

THEORY: -

Amazon Web Services (AWS): Amazon Web Services cloud provides a secure virtual platform where users can deploy their applications. Compared to an on-premises environment, AWS security provides a high level of data protection at a lower cost to its users.

What is AWS Security?

Cloud security is the highest priority in AWS. When you host your environment in the cloud, you can be assured that it's hosted in a data center or in a network architecture that's built to meet the requirements of the most security-sensitive organization. Additionally, this high level of security is available on a pay-as-you-go basis, meaning there is really no upfront cost, and the cost for using the service is a lot cheaper compared to an on-premises environment. There are many types of security services available but some of them are widely used by AWS, such as:

- IAM
- Key Management System (KMS)
- Cognito
- Web Access Firewall (WAF)

What is IAM?

- AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS resources.
- It enables you to create and control services for user authentication or limit access to a certain set of people who use your AWS resources.

How Does IAM Work?

The IAM workflow includes the following six elements:

- 1. Principal:** A principal is an entity that can perform actions on an AWS resource. A user, a role or an application can be a principal.

2. Authentication: Authentication is the process of confirming the identity of the principal trying to access an AWS product. The principal must provide its credentials or required keys for authentication.

3. Request: A principal sends a request to AWS specifying the action and which resource should perform it.

4. Authorization: By default, all resources are denied. IAM authorizes a request only if all parts of the request are allowed by a matching policy. After authenticating and authorizing the request, AWS approves the action.

5. Actions: Actions are used to view, create, edit or delete a resource.

6. Resources: A set of actions can be performed on a resource related to your AWS account.

Components of IAM:

Users:

An IAM user is an identity with an associated credential and permissions attached to it. This could be an actual person who is a user, or it could be an application that is a user. With IAM, you can securely manage access to AWS services by creating an IAM user name for each employee in your organization. Each IAM user is associated with only one AWS account. By default, a newly created user is not authorized to perform any action in AWS. The advantage of having one-to-one user specification is that you can individually assign permissions to each user.

Groups:

A collection of IAM users is an IAM group. You can use IAM groups to specify permissions for multiple users so that any permissions applied to the group are applied to the individual users in that group as well. Managing groups is quite easy. You set permissions for the group, and those permissions are automatically applied to all the users in the group. If you add another user to the group, the new user will automatically inherit all the policies and the permissions already assigned to that group. This lessens the administrative burden.

Policies:

An IAM policy sets permission and controls access to AWS resources. Policies are stored in AWS as JSON documents. Permissions specify who has access to the resources and what actions they can perform. For example, a policy could allow an IAM user to access one of the buckets in Amazon S3. The policy would contain the following information:

- Who can access it

- What actions that user can take
- Which AWS resources that user can access
- When they can be accessed

There are two types of policies:

1) Managed policies:

A managed policy is a default policy that you attach to multiple entities (users, groups, and roles) in your AWS account. Managed policies, whether they are AWS-managed or customer-managed, are stand-alone identity-based policies attached to multiple users and/or groups.

2) Inline policies:

Inline policies are policies that you create that are embedded directly into a single entity (user, group or role).

Roles:

An IAM role is a set of permissions that define what actions are allowed and denied by an entity in the AWS console. It is similar to a user in that it can be accessed by any type of entity (an individual or AWS service). Role permissions are temporary credentials.

Features of IAM:

- **Shared access to the AWS account.** The main feature of IAM is that it allows you to create separate usernames and passwords for individual users or resources and delegate access.
- **Granular permissions.** Restrictions can be applied to requests. For example, you can allow the user to download information, but deny the user the ability to update information through the policies.
- **Multifactor authentication (MFA).** IAM supports MFA, in which users provide their username and password plus a one-time password from their phone—a randomly generated number used as an additional authentication factor.
- **Identity Federation.** If the user is already authenticated, such as through a Facebook or Google account, IAM can be made to trust that authentication method and then allow access based on it. This can also be used to allow users to maintain just one password for both onpremises and cloud environment work.

- **Free to use.** There is no additional charge for IAM security. There is no additional charge for creating additional users, groups or policies.
- **PCI DSS compliance.** The Payment Card Industry Data Security Standard is an information security standard for organizations that handle branded credit cards from the major card schemes. IAM complies with this standard.
- **Password policy.** The IAM password policy allows you to reset a password or rotate passwords remotely. You can also set rules, such as how a user should pick a password or how many attempts a user may make to provide a password before being denied access.

Multi-Factor Authentication (MFA):

MFA is an additional level of security process provided by AWS. Here, a user's identity is confirmed for AWS login only after performing two levels of verification. For ex.: For Email login first level of verification would be providing username and password and second level of verification will be the OTP send to Mobile.

CREATING ACCOUNT

1) Click on create account after going to below link: -

<https://portal.aws.amazon.com/billing/signup#/start/email>

Enter Details→Click Verify email address→Enter Verification Code received in email

Sign up for AWS

Root user email address

Used for account recovery and some administrative functions

AWS account name

Choose a name for your account. You can change this name in your account settings after you sign up.

Verify email address

OR

Sign in to an existing AWS account

Sign up for AWS

Confirm you are you

Making sure you are secure -- it's what we do.

We sent an email with a verification code to

2021.christy.philip@ves.ac.in. ([not you?](#))

Enter it below to confirm your email.

Verification code

Verify

Resend code

2) Set the password for the account

Sign up for AWS

Create your password

✔ It's you! Your email address has been successfully verified. ✕

Your password provides you with sign in access to AWS, so it's important we get it right.

Root user password

Confirm root user password

Continue (step 1 of 5)

OR

3) Enter the following details

- a) Contact Information
- b) Billing Information
- c) Confirming your Identity
- d) Entering the OTP received in phone number given

NOTE: -

Creating Virtual Card

For entering the Card Details you can download the ICICI Pockets App from the Google Play Store in your mobile phone.

Enter the details

You will get a virtual card of the VISA add around 10 Rs by any payment options to card

We are doing it to prevent any loss from your original bank account

Contact Information

How do you plan to use AWS?

- ☐ Business - for your work, school, or organization
- ☒ Personal - for your own projects

Who should we contact about this account?

Full Name

Christy Philip

Phone Number

+91 8454879108

Country or Region

India

Address

Mumbai

1,Dombivli,421202

City

Dombivli

State, Province, or Region

Maharashtra

Postal Code

421202

Customers with an Indian contract address are served by Amazon Internet Services Private Ltd. (AISPL). AISPL is the local seller for AWS services in India.

☒ I have read and agree to the terms of the AWS Customer Agreement [link](#).

[Continue \(step 2 of 5\)](#)

Confirm your identity

Before you can use your AWS account, you must verify your phone number. When you continue, the AWS automated system will contact you with a verification code.

How should we send you the verification code?

- ☒ Text message (SMS)
- ☐ Voice call


Country or region code

United States (+1)

Mobile phone number

Billing Information

Credit or Debit card number

 The credit card number is required.



AWS accepts all major credit and debit cards. To learn more about payment options, review our [FAQ](#)

Expiration date

Month Year

Cardholder's name

CVV

Billing address

- ☒ Use my contact address

Mumbai
Dombivli Maharashtra 421202
IN

- ☐ Use a new address

Sign up for AWS

Confirm your identity

Verify code

 SMS PIN required

[Continue \(step 4 of 5\)](#)

Having trouble? Sometimes it takes up to 10 minutes to retrieve a verification code. If it's been longer than that, [return to the previous page](#) and try again.

4) Now choose Basic Support Free then a Congratulations message is shown

Sign up for AWS

Select a support plan

Choose a support plan for your business or personal account. [Compare plans and pricing examples](#). You can change your plan anytime in the AWS Management Console.

☒ **Basic support - Free**

- Recommended for new users just getting started with AWS
- 24x7 self-service access to AWS resources
- For account and billing issues only
- Access to Personal Health Dashboard & Trusted Advisor



☐ **Developer support - From \$29/month**

- Recommended for developers experimenting with AWS
- Email access to AWS Support during business hours
- 12 (business)-hour response times



☐ **Business support - From \$100/month**

- Recommended for running production workloads on AWS
- 24x7 tech support via email, phone, and chat
- 1-hour response times
- Full set of Trusted Advisor best-practice recommendations



Congratulations

Thank you for signing up for AWS.

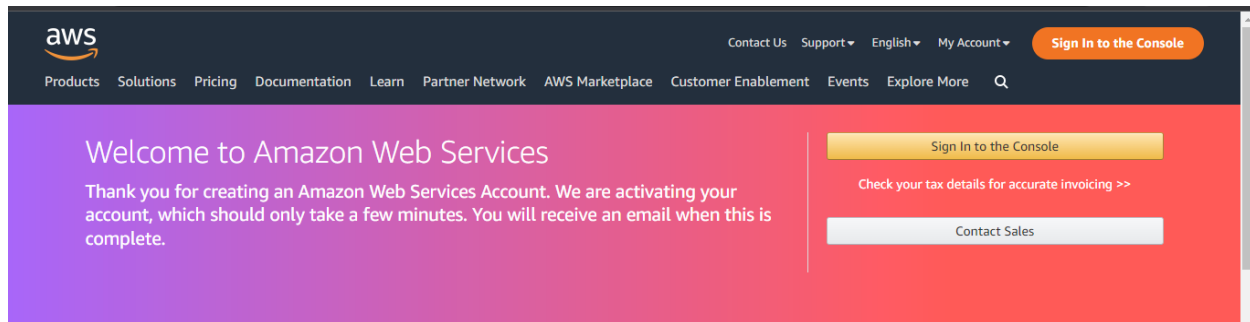
We are activating your account, which should only take a few minutes. You will receive an email when this is complete.

[Go to the AWS Management Console](#)

SIGNING IN

1) <https://signin.aws.amazon.com/signin>

Goto above link and sign into the console



2) Choose Root user→Enter root user email address→click Next→Enter Password→Sign in

Sign in

☒ **Root user**
Account owner that performs tasks requiring unrestricted access. [Learn more](#)

☐ **IAM user**
User within an account that performs daily tasks. [Learn more](#)

Root user email address



Root user sign in ⓘ

Email: 2021.christy.philip@ves.ac.in

Password

[Forgot password?](#)

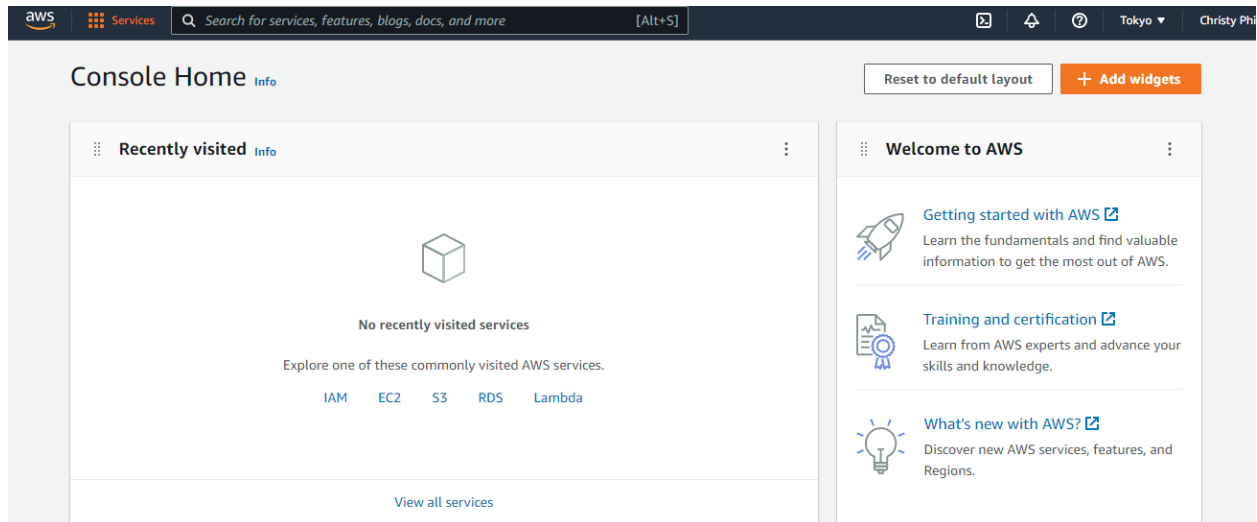
Sign in

[Sign in to a different account](#)

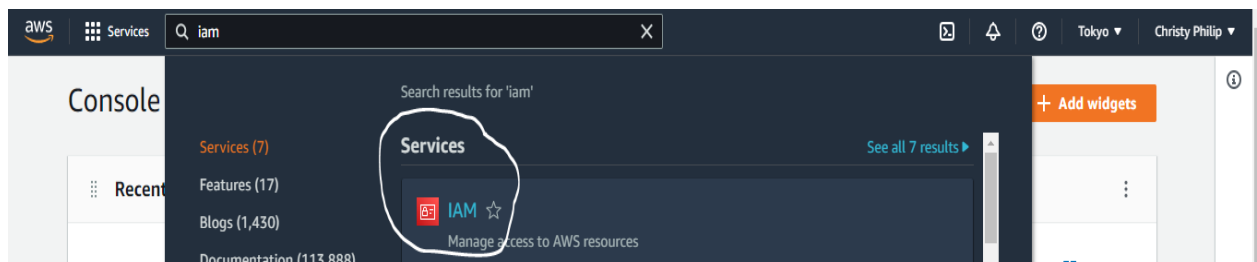
[Create a new AWS account](#)

STEPS: -

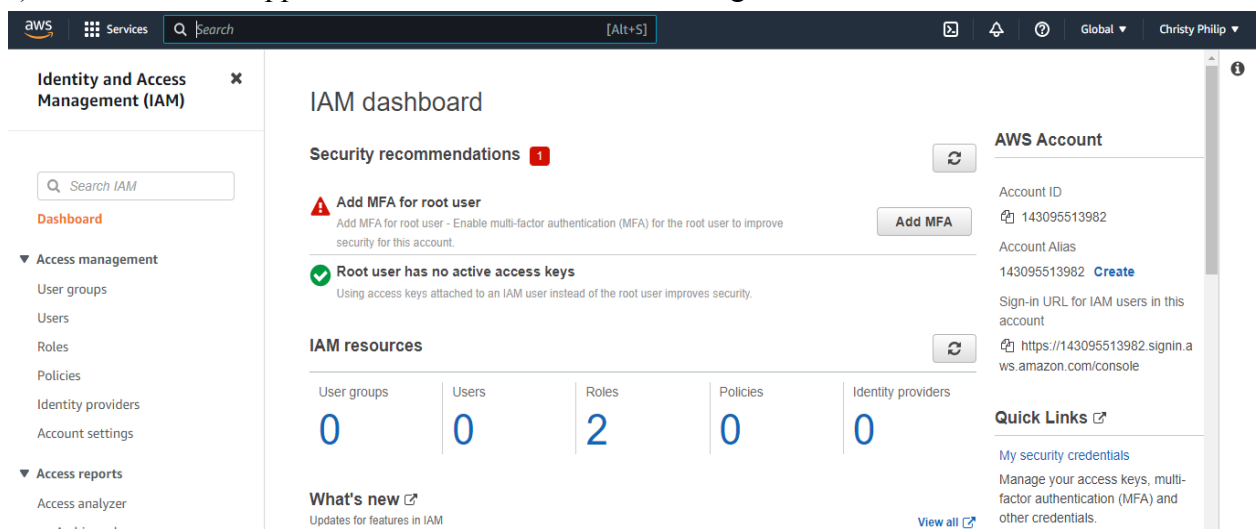
1) After login, you will be redirected to AWS Dashboard:



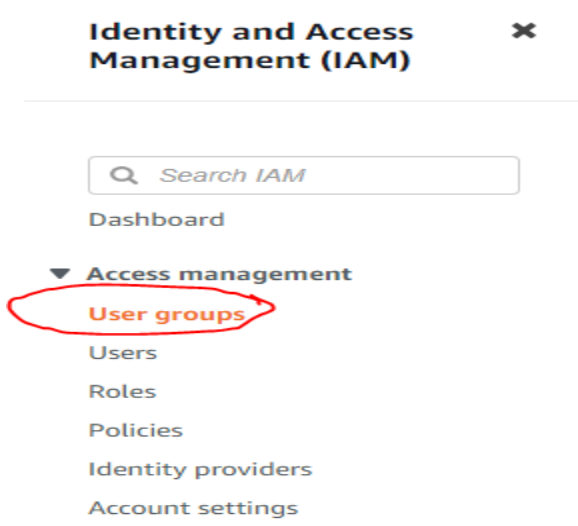
2) In search bar of service section, search “IAM”:
And click on IAM:



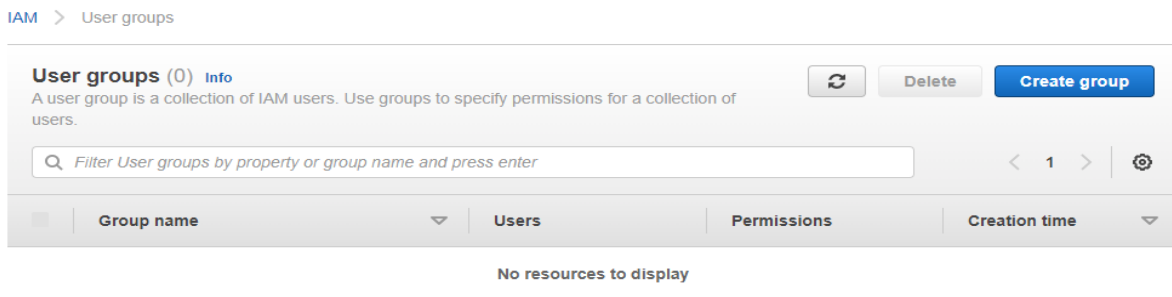
3) I am Dashboard appears which is shown in below image



4) In Left side menu→Access Management→Choose User groups



5) Below window appears Click on Create Group



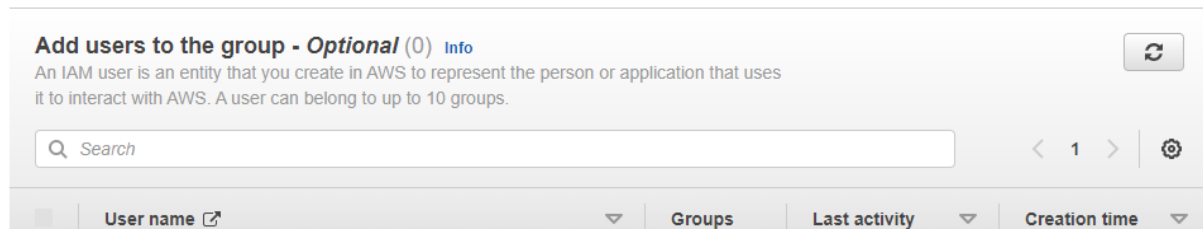
6) Give unique & meaningful name to the group: [user_rights]

Create user group

Name the group

User group name
Enter a meaningful name to identify this group.

Maximum 128 characters. Use alphanumeric and '+-=, @-_' characters.



7) Select attach permissions policies

Search for selected name("AmazonS3") you will find below result select it

Select "AmazonS3FullAccess" [Reason: allowing to do all things such as create, update, delete, read s3 bucket]

Attach permissions policies - *Optional* (Selected 1/777)






Info

You can attach up to 10 policies to this user group. All the users in this group will have permissions that are defined in the selected policies.

Q Filter policies by property or policy name and press enter.

5 matches < 1 > ⚙

"AMAZONS3" X Clear filters

<input type="checkbox"/>	Policy name	Type	Description
<input checked="" type="checkbox"/>	 AmazonS3FullAccess	AWS managed	Provides full access to Amazon S3
<input type="checkbox"/>	 AmazonS3ReadOnlyAccess	AWS managed	Provides read only access to Amazon S3
<input type="checkbox"/>	 AmazonS3OutpostsFullAccess	AWS managed	Provides full access to Amazon S3 Outposts
<input type="checkbox"/>	 AmazonS3ObjectLambdaExecutionRolePolicy	AWS managed	Provides AWS Lambda permissions for Amazon S3 Object Lambda
<input type="checkbox"/>	 AmazonS3OutpostsReadOnlyAccess	AWS managed	Provides read only access to Amazon S3 Outposts

8) The group created message below can be shown below when successfully created.

user_rights user group created.

View group X

IAM > User groups

User groups (1) Info

A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.

Filter User groups by property or group name and press enter

< 1 > ⚙

<input type="checkbox"/>	Group name	Users	Permissions	Creation time
<input type="checkbox"/>	user_rights	Loading	Defined	Now

9) We need to create one more group which have only limited to access
Give unique & meaningful name to the group[user_rights_read_only]

IAM > User groups > Create user group

Create user group

Name the group

User group name

Enter a meaningful name to identify this group.

user_rights_read_only

Maximum 128 characters. Use alphanumeric and '+=, @-_' characters.

Add users to the group - *Optional* (0) [Info](#)

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS. A user can belong to up to 10 groups.

Search

User name [↗](#) Groups Last activity Creation time

10) Select attach permissions policies

Select “AmazonS3ReadOnlyAccess” [Reason: only allowed to read s3 bucket]

Attach permissions policies - *Optional* (Selected 1/777) [Info](#)

You can attach up to 10 policies to this user group. All the users in this group will have permissions that are defined in the selected policies.

Filter policies by property or policy name and press enter.


5 matches

"AMAZONS3" [×](#)

Clear filters

	Policy name ↗	Type	Description
<input type="checkbox"/>	+ AmazonS3FullAccess	AWS managed	Provides full access to all buckets via th...
<input checked="" type="checkbox"/>	+ AmazonS3ReadOnlyAccess	AWS managed	Provides read only access to all buckets...
<input type="checkbox"/>	+ AmazonS3OutpostsFullAccess	AWS managed	Provides full access to Amazon S3 on O...
<input type="checkbox"/>	+ AmazonS3ObjectLambdaExecutionRoleP...	AWS managed	Provides AWS Lambda functions permis...
<input type="checkbox"/>	+ AmazonS3OutpostsReadOnlyAccess	AWS managed	Provides read only access to Amazon S...

11) The group created message below can be shown below when successfully created.

 user_rights_read_only user group created. View group ×

[IAM](#) > User groups

User groups (2) [Info](#)

↻ Delete Create group

☐

Group name

▼

Users

Permissions

Creation time

▼

☐

user_rights

↻ Loading

✔ Defined

4 minutes ago

☐

user_rights_read_only

↻ Loading

✔ Defined

Now

12) Now go to Dashboard and in left pane→Choose Users

Identity and Access Management (IAM) ×

Dashboard

▼ Access management

User groups

Users

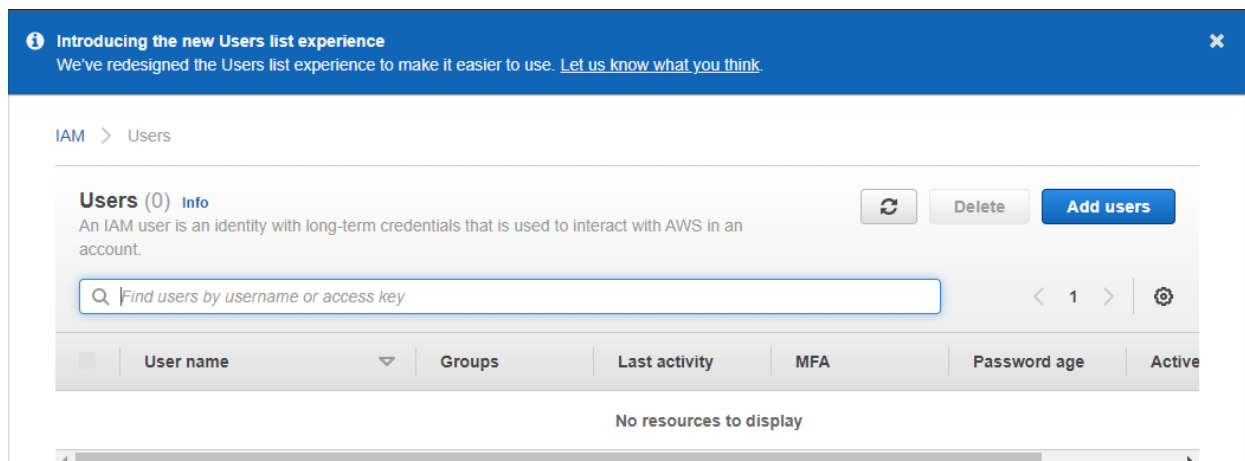
Roles

Policies

Identity providers

Account settings

13) You will be redirected to Users section→Click on Add users



14) Click on “Add User”:

We need 2 users for demonstration of the Identity Access Management(IAM) & S3 bucket:
Create user “admin_christy”:

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[+ Add another user](#)

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

- Select AWS credential type* ☒ **Access key - Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- ☒ **Password - AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

Console password* ☐ Autogenerated password
☒ Custom password

☐ Show password


Require password reset ☐ User must create a new password at next sign-in
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.


15) Select Add user to group: select “user_rights” [reason to choose this: admin will have the access to all s3 bucket services]


Add user

1 2 3 4 5

Set permissions

 Add user to group

 Copy permissions from existing user

 Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Add user to group

Create group

Refresh

Q Search		Showing 2 results
Group	Attached policies	
<input checked="" type="checkbox"/> user_rights	AmazonS3FullAccess	
<input type="checkbox"/> user_rights_read_only	AmazonS3ReadOnlyAccess	

16) Nothing to be entered in below part

Add user

1 2 3 4 5

Add tags (optional)

IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. [Learn more](#)

Key	Value (optional)	Remove
<input type="text" value="Add new key"/>	<input type="text"/>	

You can add 50 more tags.

17) Click on Create User

Add user

1 2 3 4 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	admin_christy
AWS access type	Programmatic access and AWS Management Console access
Console password type	Custom
Require password reset	No
Permissions boundary	Permissions boundary is not set

Permissions summary

The user shown above will be added to the following groups.

Type	Name
Group	user_rights

Tags

No tags were added.


Cancel

Previous

Create user


18) Now you can see in this section the user created success message is shown.





Add user 1 2 3 4 **5**

 **Success**



You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://143095513982.signin.aws.amazon.com/console>

 Download .csv

	User	Access key ID	Secret access key	Email login instructions
	 admin_christy	AKIASCUJIIN7GYH5PVNG 	***** Show	Send email 


19) Click on Add User for adding second user



 The user admin_christy have been created. 



[IAM](#) > [Users](#)

Users (1) [Info](#)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

 [Delete](#) [Add users](#)

 Find users by username or access key < 1 > 

<input type="checkbox"/>	User name	Groups	Last activity	MFA	Password age	Active key age
<input type="checkbox"/>	admin_christy	user_rights	Never	None	 Now	 Now

20) Create user “manager_christy”:

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[Add another user](#)

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Select AWS credential type* ☒ **Access key - Programmatic access**
Enables an **access key** ID and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☒ **Password - AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

Console password* ☐ Autogenerated password
☒ Custom password

☐ Show password

Require password reset ☐ User must create a new password at next sign-in
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

* Required Cancel Next: Permissions

21) Add user to group: select “user_rights_ready_only” [reason to choose this: manager will have the limited access to s3 bucket services(only read)]

Add user 1 2 3 4 5

▼ Set permissions

Add user to group

Copy permissions from existing user

Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Add user to group

Create group Refresh

Showing 2 results

Group ▼	Attached policies
<input type="checkbox"/> user_rights	AmazonS3FullAccess
<input checked="" type="checkbox"/> user_rights_read_only	AmazonS3ReadOnlyAccess

► Set permissions boundary

22) Nothing to be entered below

Add user

1 2 3 4 5

Add tags (optional)

IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. [Learn more](#)

Key	Value (optional)	Remove
<input type="text" value="Add new key"/>	<input type="text"/>	

You can add 50 more tags.

23) Click on Create User

Add user

1 2 3 4 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	manager_christy
AWS access type	Programmatic access and AWS Management Console access
Console password type	Custom
Require password reset	No
Permissions boundary	Permissions boundary is not set

Permissions summary

The user shown above will be added to the following groups.

Type	Name
Group	user_rights_read_only

Tags

No tags were added.

Cancel Previous **Create user** Active

24) Now you can see in this section the user created success message is shown.

Add user

1 2 3 4 5



Success


You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://143095513982.signin.aws.amazon.com/console>

Download .csv

	User	Access key ID	Secret access key	Email login instructions
▶	✓ manager_christy	AKIASCUJIIN7ECIQ5GM7	***** Show	Send email

25) You can see both the users are created



IAM > Users

Users (2) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

< 1 >

☐

User name

Groups

Last activity

MFA

Password age

Active key age

☐

admin_christy

user_rights

Never

None

7 minutes ago

7 minutes ago

☐

manager_christy

user_rights_read_onl

Never

None

Now

Now

26) Check permissions [navigate to users section]:

a) manager_christy

Users > manager_christy

Summary

Delete user ?

User ARN am:aws:iam::143095513982:user/manager_christy

Path /

Creation time 2022-11-02 11:39 UTC+0530

Permissions

Groups (1)

Tags

Security credentials

Access Advisor

Permissions policies (1 policy applied)

Add permissions

Add inline policy

Policy name	Policy type
Attached from group	
AmazonS3ReadOnlyAccess	AWS managed policy from group user_rights_read_only

Permissions boundary (not set)

b) admin_christy

Users > admin_christy

Summary

Delete user ?

User ARN am:aws:iam::143095513982:user/admin_christy

Path /

Creation time 2022-11-02 11:32 UTC+0530

Permissions

Groups (1)

Tags

Security credentials

Access Advisor

Permissions policies (1 policy applied)

Add permissions

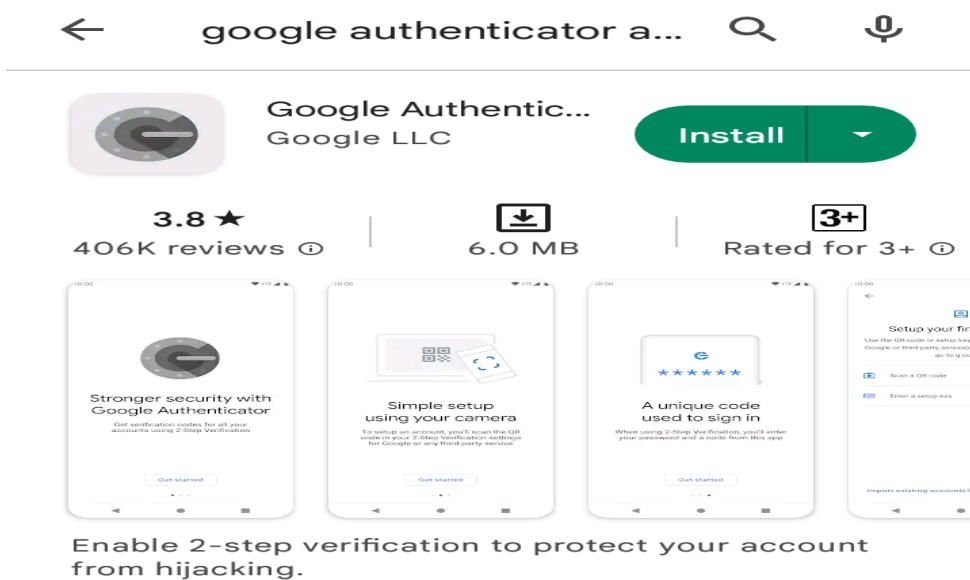
Add inline policy

Policy name	Policy type
Attached from group	
AmazonS3FullAccess	AWS managed policy from group user_rights

Permissions boundary (not set)

MFA - Multi Factor Authentication

27) On your mobile Install Google Authenticator in google play store



28) Attaching MFA to admin_christy

Go to admin_christy Summary [in search bar of service select “IAM”, then select users section]

Click on Security credentials and go to Assigned MFA device and select virtual MFA device

Users > admin_christy

Summary

Details

User ARN: [arn:aws:iam::143095513982:user/admin_christy](#)
Path: [/](#)
Creation time: 2022-11-02 11:32 UTC+0530

Permissions

Groups (1)

Tags

Security credentials

Access Advisor

▼ Permissions policies (1 policy applied)

Add permissions

Add

Policy name ▼	Policy type ▼
Attached from group	
<div><div></div><div>AmazonS3FullAccess</div></div>	AWS managed policy from group user_rights

▶ Permissions boundary (not set)

▼ Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this user, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

Share your [feedback](#) and help us improve the policy generation experience.

Generate policy

Activate Windows

29) Click on Manage

Summary

User ARN

arn:aws:iam::143095513982:user/admin_christy

Path

/

Creation time

2022-11-02 11:32 UTC+0530

Permissions

Groups (1)

Tags

Security credentials

Access Advisor

Sign-in credentials

Summary

• Console sign-in link: <https://143095513982.signin.aws.amazon.com/console>


Console password

Enabled (never signed in) | [Manage](#)

Assigned MFA device

Not assigned | [Manage](#)

Signing certificates

None 

30) Prompt appears select Virtual MFA device

Manage MFA device

Choose the type of MFA device to assign:

☒ **Virtual MFA device**
Authenticator app installed on your mobile device or computer

☐ **Security key**
Authenticate by touching a FIDO security key, such as Yubikey

☐ **Other hardware MFA device**
Hardware TOTP token

For more information about supported MFA devices, see [AWS Multi-Factor Authentication](#)

Cancel


Continue

31) Scan the below code using Google Authenticator app which is installed on your mobile
Code starts coming 2 codes we need to enter

Set up virtual MFA device

1. Install a compatible app on your mobile device or computer
[See a list of compatible applications](#)

2. Use your virtual MFA app and your device's camera to scan the QR code



Alternatively, you can type the secret key. [Show secret key](#)

3. Type two consecutive MFA codes below

MFA code 1

MFA code 2

Cancel

Previous

Assign MFA

32) Success message is shown there of creating the MFA.

Set up virtual MFA device

 You have successfully assigned virtual MFA
This virtual MFA will be required during sign-in.

Close

33) Security credentials section check MFA assigned or not.

Summary

User ARNarn:aws:iam::143095513982:user/admin_christy

Path/

Creation time2022-11-02 11:32 UTC+0530

Permissions

Groups (1)

Tags

Security credentials

Access Advisor

Sign-in credentials

Summary

- Console sign-in link: <https://143095513982.signin.aws.amazon.com/console>
- MFA is required when signing in. [Learn more](#)

Console password

Enabled (never signed in) | [Manage](#)

Assigned MFA device

arn:aws:iam::143095513982:mfa/admin_christy (Virtual) | [Manage](#)

Signing certificates

None

Access keys

34) Goto Users Section and verify whether MFA is assigned or not to admin_christy

IAM > Users

Users (2) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Find users by username or access key

< 1 >

admin_christy

user_rights

Never

Virtual

28 minutes ago

28 minutes ago

manager_christy

user_rights_read_only

Never

None

21 minutes ago

21 minutes ago

35) In search bar of service section, search “S3”:

And click on S3:

S3

Search results for 'S3'

Services (7)

Features (14)

Blogs (1,144)

Documentation (103,559)

Knowledge Articles (30)

Tutorials (9)

Services

See all 7 results

S3

Scalable Storage in the Cloud

S3 Glacier

Archive Storage in the Cloud

36) S3 Dashboard is shown below

The screenshot shows the Amazon S3 console dashboard. At the top, a blue banner contains a message: "We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose [Provide feedback](#)." Below this, the main header area has the word "Storage" in small text, followed by "Amazon S3" in large bold font, and the tagline "Store and retrieve any amount of data from anywhere". A sub-header states: "Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance." To the right, there is a "Create a bucket" section with a description: "Every object in S3 is stored in a bucket. To upload and folders to S3, you'll need to create a bucket the objects will be stored." Below this is an orange "Create bucket" button. Further down is a "Pricing" section with text: "With S3, there are no minimum fees. You only pay for what you use. Prices are based on the location of the bucket." and a link to "Estimate your monthly bill using the [AWS Simple Monthly Calculator](#)". On the left, under "How it works", there is a video player titled "Introduction to Amazon S3" with a "Copy link" button.

37) Click on the create a bucket:

Give a meaningful name to the bucket [s3-christy]

Select any AWS Region [preferred is mumbai]

The screenshot shows the "Create bucket" page in the Amazon S3 console. The breadcrumb navigation at the top reads "Amazon S3 > Buckets > Create bucket". The main heading is "Create bucket" with an "Info" link. Below the heading, it says "Buckets are containers for data stored in S3. [Learn more](#)". The "General configuration" section contains a "Bucket name" field with the value "s3-christy". Below the field, a note states: "Bucket name must be globally unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)". The "AWS Region" is set to "Asia Pacific (Mumbai) ap-south-1" in a dropdown menu. At the bottom, there is a section for "Copy settings from existing bucket - optional" with the text "Only the bucket settings in the following configuration are copied." and a "Choose bucket" button.

38) Keep scrolling below, let all settings be as shown in below images

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☒ **ACLs disabled (recommended)**

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ **ACLs enabled**

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner enforced

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☒ **Block all public access**

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☒ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☒ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**

S3 will ignore all ACLs that grant public access to buckets and objects.

Bucket Versioning

☒ **Disable**

☐ **Enable**

Tags (0) - optional

Track storage cost or other criteria by tagging your bucket. [Learn more](#)

No tags associated with this bucket.

Add tag

Default encryption

Automatically encrypt new objects stored in this bucket. [Learn more](#)

Server-side encryption

☒ **Disable**

☐ **Enable**

39) Click on create bucket:

Amazon S3 > Buckets

Account snapshot [View Storage Lens dashboard](#)

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

Buckets (1) [Info](#) [Refresh](#) [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

Buckets are containers for data stored in S3. [Learn more](#)

	Name ▲	AWS Region ▼	Access ▼	Creation date ▼
<input type="radio"/>	s3-christy	Asia Pacific (Mumbai) ap-south-1	Bucket and objects not public	November 2, 2022, 12:09:12 (UTC+05:30)

40) Click on the s3-christy:

Click on Create folder: [christyp10]

Objects (0)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Refresh](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions ▼](#)

[Create folder](#) [Upload](#)

	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
--	--------	--------	-----------------	--------	-----------------

41) Enter details as shown below and click on Create folder

Folder

Folder name

 /

Folder names can't contain "/". See rules for naming [🔗](#)

Server-side encryption

The following settings apply only to the new folder object and not to the objects contained within it.

Server-side encryption

☒ Disable

☐ Enable

Active

42) Click on Upload when you are in christyp10 folder

christyp10/

Objects | Properties

Objects (0)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
--------------------------	------	------	---------------	------	---------------

43) Click on Add files

Amazon S3 > Buckets > s3-christy > christyp10/ > Upload

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

Files and folders (0)

All files and folders in this table will be uploaded.

< 1 >

<input type="checkbox"/>	Name	Folder	Type	Size
--------------------------	------	--------	------	------

No files or folders
You have not chosen any files or folders to upload.

Destination

Destination

[s3://s3-christy/christyp10/](#)

► **Destination details**

Bucket settings that impact new objects stored in the specified destination.

► **Permissions**

Grant public access and access to other AWS accounts.

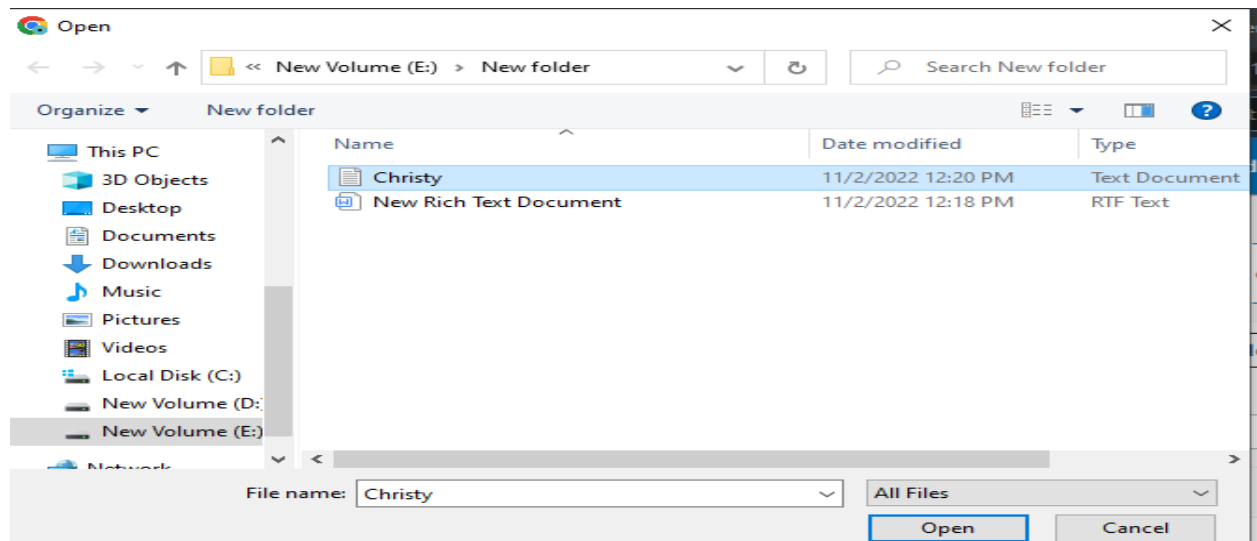
► **Properties**

Specify storage class, encryption settings, tags, and more.

Cancel

Upload

44) Browse the file and choose it



45) After file appears then click Upload

Files and folders (1 Total, 40.0 B)

Remove

Add files

Add folder

All files and folders in this table will be uploaded.

Q

Find by name

< 1 >

<input type="checkbox"/>	Name ▲	Folder ▼	Type ▼	Size ▼
<input type="checkbox"/>	Christy.txt	-	text/plain	40.0 B

Destination

Destination

s3://s3-christy/christyp10/

► Destination details

Bucket settings that impact new objects stored in the specified destination.

► Permissions

Grant public access and access to other AWS accounts.

► Properties

Destination

Destination

s3://s3-christy/christyp10/

► Destination details

Bucket settings that impact new objects stored in the specified destination.

► Permissions

Grant public access and access to other AWS accounts.

► Properties

Specify storage class, encryption settings, tags, and more.

Cancel

Upload

46) Check the status whether the file is uploaded or not:

Upload succeeded
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

Summary

Destination s3://s3-christy/christyp10/	Succeeded 1 file, 40.0 B (100.00%)
--	---------------------------------------

Files and folders

Configuration

Files and folders (1 Total, 40.0 B)

Find by name

Name	Folder	Type	Size	Status
Christy.txt	-	text/plain	40.0 B	Succeeded

47) Now, check identity management on S3 bucket by logging using manager_christy & admin_christy:

Note: -

Account ID will be getting by hovering mouse on dashboard profile section

A) manager_christy login: -

Sign in as IAM user

Account ID (12 digits) or account alias

143095513982

IAM user name

manager_christy

Password

.....|

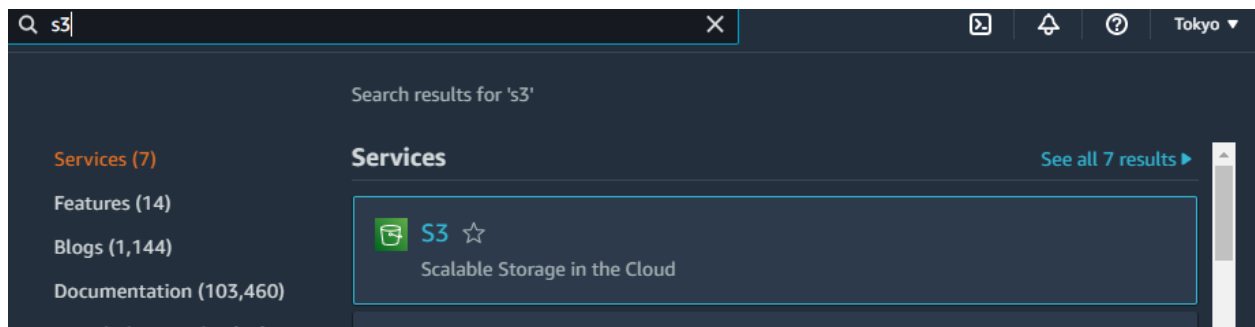
☐ Remember this account

Sign in

[Sign in using root user email](#)

[Forgot password?](#)

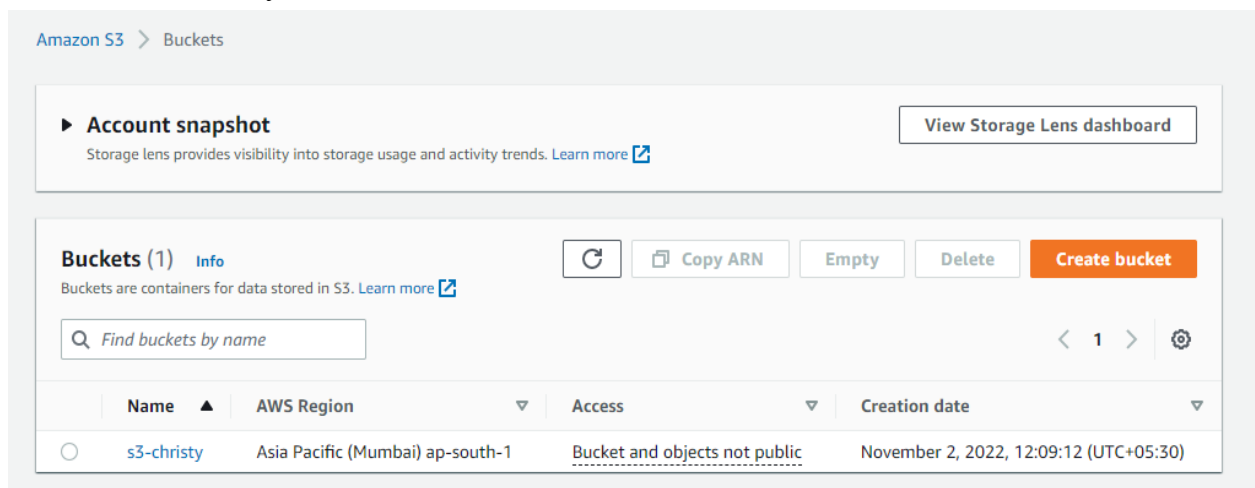
48) Search for S3



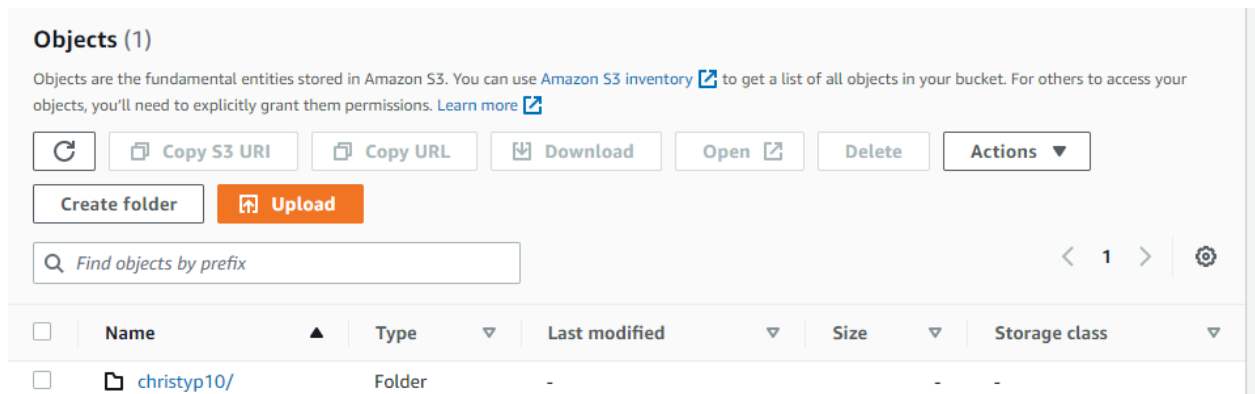
49) As we specified read only permission for manager_christy:

Now we try to delete s3 bucket/objects/files:

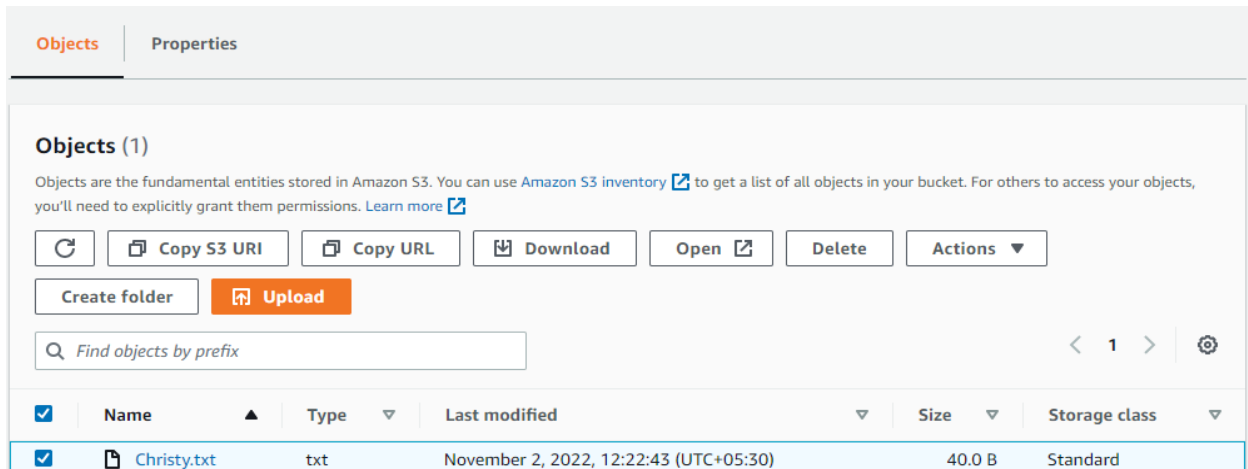
So click on s3-christy



50) Click on christyp10



51) Click on Christy.txt and choose delete



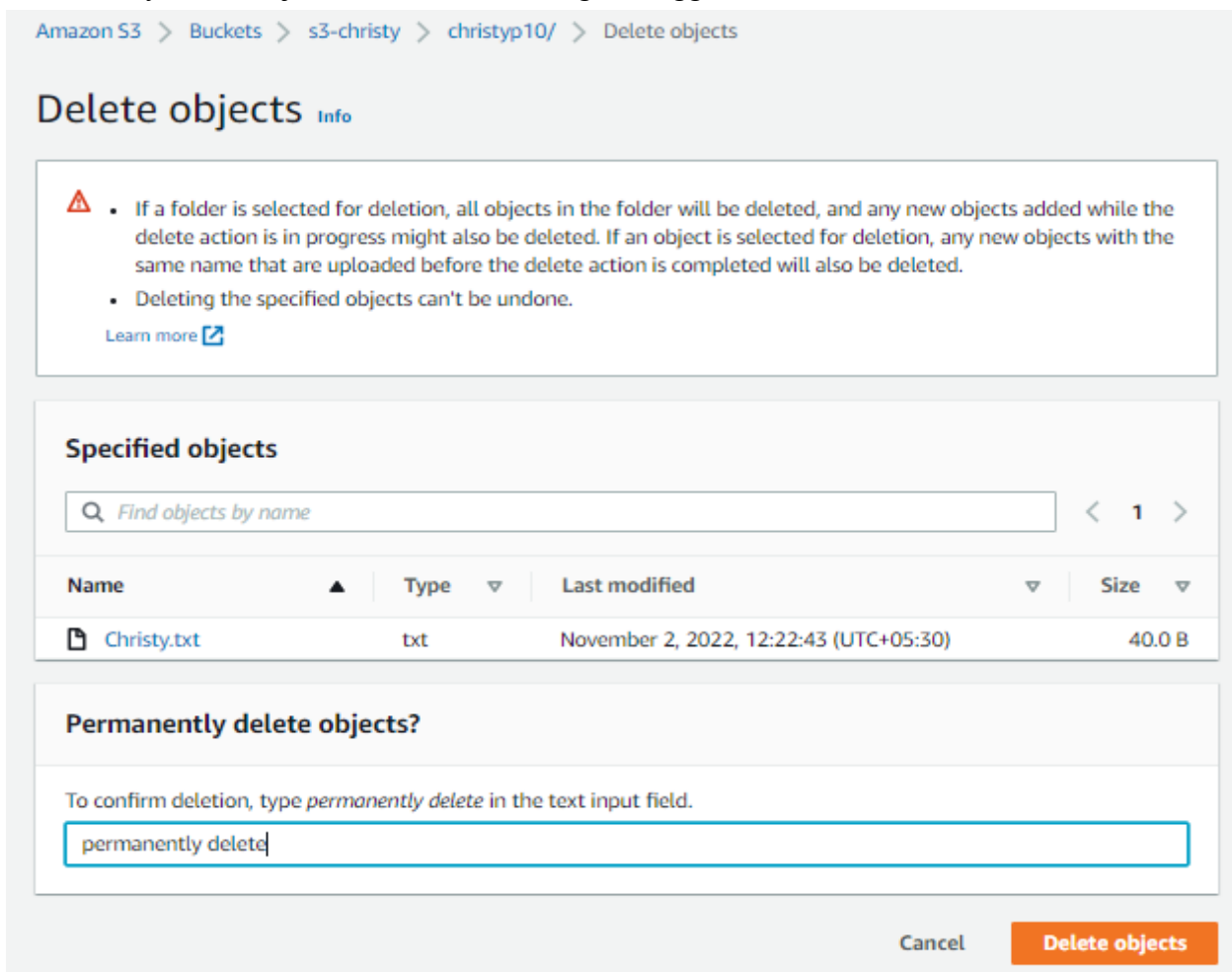
The screenshot shows the Amazon S3 console interface. At the top, there are tabs for 'Objects' and 'Properties'. Below the tabs, there's a section titled 'Objects (1)' with a description and links for 'Amazon S3 inventory' and 'Learn more'. A toolbar contains buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', and 'Actions'. Below the toolbar is a search bar labeled 'Find objects by prefix' and a pagination control showing '1'. A table lists the objects with columns: Name, Type, Last modified, Size, and Storage class. The table contains one row for 'Christy.txt' with type 'txt', last modified 'November 2, 2022, 12:22:43 (UTC+05:30)', size '40.0 B', and storage class 'Standard'.

<input checked="" type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input checked="" type="checkbox"/>	Christy.txt	txt	November 2, 2022, 12:22:43 (UTC+05:30)	40.0 B	Standard

52) Below window appears, enter the details

Note: -

Permanently delete objects check what message is suggested there it should be entered



The screenshot shows the 'Delete objects' dialog box in the Amazon S3 console. The breadcrumb trail at the top reads 'Amazon S3 > Buckets > s3-christy > christyp10/ > Delete objects'. The main heading is 'Delete objects' with an 'Info' link. A warning box contains a red triangle icon and text: 'If a folder is selected for deletion, all objects in the folder will be deleted, and any new objects added while the delete action is in progress might also be deleted. If an object is selected for deletion, any new objects with the same name that are uploaded before the delete action is completed will also be deleted. Deleting the specified objects can't be undone. Learn more'. Below the warning is a section titled 'Specified objects' with a search bar 'Find objects by name' and a pagination control '1'. A table lists the objects to be deleted with columns: Name, Type, Last modified, and Size. The table contains one row for 'Christy.txt' with type 'txt', last modified 'November 2, 2022, 12:22:43 (UTC+05:30)', and size '40.0 B'. Below the table is a section titled 'Permanently delete objects?' with a text input field containing 'permanently delete'. At the bottom right are 'Cancel' and 'Delete objects' buttons.

Delete objects Info

Specified objects

Find objects by name

Name	Type	Last modified	Size
Christy.txt	txt	November 2, 2022, 12:22:43 (UTC+05:30)	40.0 B

Permanently delete objects?

To confirm deletion, type *permanently delete* in the text input field.

permanently delete

Cancel Delete objects

53) The object can't be deleted because manager_christy does not have the access/permission

Summary

Source
s3://s3-christy/christyp10/

Successfully deleted
0 objects

Failed to delete
❌ 1 object, 40.0 B

Failed to delete

Configuration

❌ **Failed to delete** (1 object, 40.0 B)

Name	Folder	Type	Last modified	Size	Error
Christy.txt	christyp10/	txt	November 2, 2022, 12:22:43 (UTC+05:30)	40.0 B	❌ Access denied

54) Now we will try to upload files in manager_christy:
Click on Upload

christyp10/ Copy S3 URI

Objects

Properties

Objects (1)
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open Delete **Actions** Create folder Upload

< 1 >

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	Christy.txt	txt	November 2, 2022, 12:22:43 (UTC+05:30)	40.0 B	Standard

55) Click on Add files→Browse the file→Click on upload

Amazon S3 > Buckets > s3-christy > christyp10/ > Upload

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

Files and folders (1 Total, 834.0 KB) [Remove](#) [Add files](#) [Add folder](#)
All files and folders in this table will be uploaded.

< 1 >

<input type="checkbox"/>	Name ▲	Folder ▼	Type ▼	Size ▼
<input type="checkbox"/>	PRACTICAL 9 DSCC.pdf	-	application/pdf	834.0 KB

Destination
Destination
[s3://s3-christy/christyp10/](#)
► **Destination details**
Bucket settings that impact new objects stored in the specified destination.

► **Permissions**
Grant public access and access to other AWS accounts.

56) As the manager_christy does not have the permission to upload, the file upload failed.

Upload failed
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

Summary

Destination s3://s3-christy/christyp10/	Succeeded ✔ 0 files, 0 B (0%)	Failed ✘ 1 file, 553.2 KB (100.00%)
--	----------------------------------	--

B) admin_christy login: -

Sign in as IAM user

Account ID (12 digits) or account alias

143095513982

IAM user name

admin_christy

Password

.....|

☐ Remember this account

Sign in

57) Get the below asked code from the Google Authenticator app

Multi-factor Authentication

Enter an MFA code to complete sign-in.

MFA Code:

529546

Submit

[Cancel](#)

58) Switch to S3 section [search in service section of search bar]

Click on s3-christy

Amazon S3 > Buckets

► Account snapshot View Storage Lens dashboard
Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

Buckets (1) [Info](#) Refresh Copy ARN Empty Delete Create bucket
Buckets are containers for data stored in S3. [Learn more](#)

	Name ▲	AWS Region ▼	Access ▼	Creation date ▼
<input type="radio"/>	s3-christy	Asia Pacific (Mumbai) ap-south-1	Bucket and objects not public	November 2, 2022, 12:09:12 (UTC+05:30)

59) Now, try to upload more files:

Click on the upload button:

Add files:

christyp10/

Copy S3 URI

Objects

Properties

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Refresh

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

< 1 >

Settings

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	Christy.txt	txt	November 2, 2022, 12:22:43 (UTC+05:30)	40.0 B	Standard

60) Browse the file and click on upload

Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

Files and folders (1 Total, 499.8 KB)

Remove

Add files

Add folder

All files and folders in this table will be uploaded.

Find by name

< 1 >

<input type="checkbox"/>	Name	Folder	Type	Size
<input type="checkbox"/>	PRACTICAL 9 DSCC.pdf	-	application/pdf	499.8 KB

Destination

Destination

s3://s3-christy/christyp10/

Destination details

Bucket settings that impact new objects stored in the specified destination.

Permissions

Grant public access and access to other AWS accounts.


Properties

Specify storage class, encryption settings, tags, and more.


Cancel

Upload


61) Successful upload message is been shown

 **Upload succeeded**
View details below.

Upload: status

 The information below will no longer be available after you navigate away from this page.


Summary

Destination s3://s3-christy/christyp10/	Succeeded  1 file, 499.8 KB (100.00%)
--	---

Files and folders | Configuration

Upload was successful, because the admin_christy has the all the access to s3 bucket services of AWS [defined in user group role]










62) Now , we will try to delete file:
Select the file which you want to delete:




christyp10/  Copy S3 URI




Objects | Properties

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

  Copy S3 URI  Copy URL  Download  Open  Delete  Actions  Create folder  Upload

 1  


	Name	Type	Last modified	Size	Storage class
<input checked="" type="checkbox"/>	 Christy.txt	txt	November 2, 2022, 12:22:43 (UTC+05:30)	40.0 B	Standard
<input type="checkbox"/>	 PRACTICAL 9 DSCC.pdf	pdf	November 2, 2022, 13:00:16 (UTC+05:30)	499.8 KB	Standard

63) Click on the delete button:

Type permanently delete in textbox:

Click on the delete objects:

Delete objects Info




- If a folder is selected for deletion, all objects in the folder will be deleted, and any new objects added while the delete action is in progress might also be deleted. If an object is selected for deletion, any new objects with the same name that are uploaded before the delete action is completed will also be deleted.
- Deleting the specified objects can't be undone.

[Learn more](#)

Specified objects

< 1 >

Name	Type	Last modified	Size
 Christy.txt	txt	November 2, 2022, 12:22:43 (UTC+05:30)	40.0 B


Permanently delete objects?

To confirm deletion, type *permanently delete* in the text input field.


Cancel

Delete objects


64) File is successfully deleted message is shown

 Successfully deleted objects
View details below.

Delete objects: status


 The information below will no longer be available after you navigate away from this page.

Summary

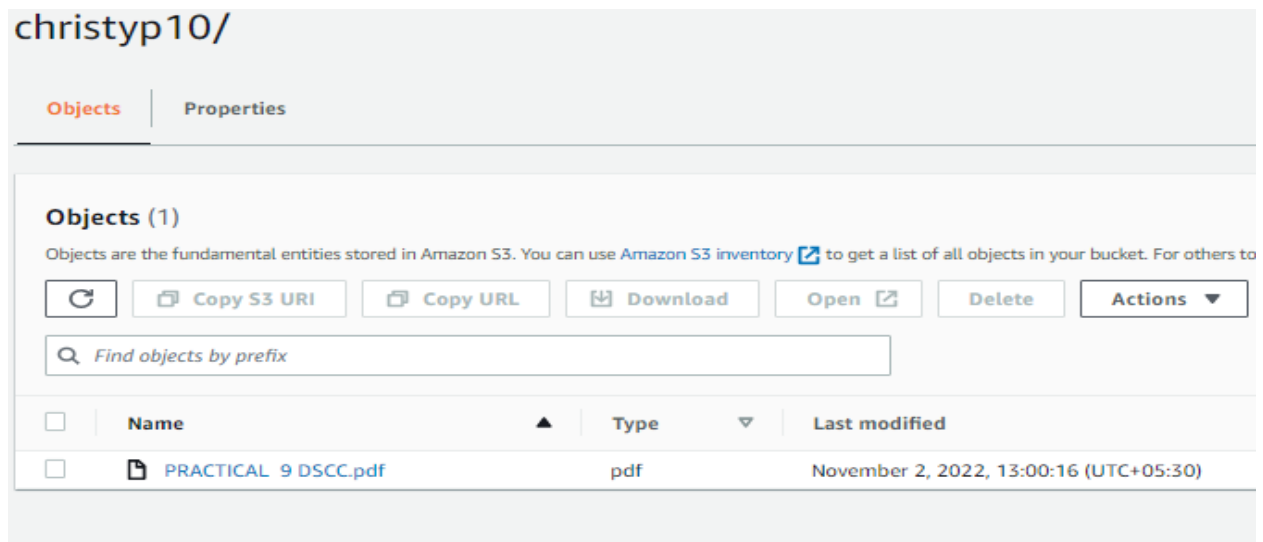
Source s3://s3-christy/christyp10/	Successfully deleted  1 object, 40.0 B	Failed to delete 0 objects
---------------------------------------	--	-------------------------------

Failed to delete

Configuration

 Failed to delete (0)

So the file is not visible now



File delete was successful, because the admin_christy has the all the access to s3 bucket services of AWS [defined in user group role]

CONCLUSION: -

From this practical I have learned Description and Implementation of Identity Management in Amazon Web Services.