Index

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.No** | **Practical** | **Date** | **Sign** | **Remark if**  **any** |
| **1** | **Program on Remote Process**  **Communication** |  |  |  |
| **2** | **Programs on Remote Procedure Call** |  |  |  |
| **3** | **Programs on Remote Method**  **Invocation** |  |  |  |
| **4** | **Practical on Remote Object**  **Communication** |  |  |  |
| **5** | **Implementation of Mutual Exclusion**  **using Token Ring Algorithm** |  |  |  |
| **6** | **Implementation of Storage as a Service**  **using Google Docs** |  |  |  |
| **7** | **Implementation of Identity**  **Management.** |  |  |  |
| **8** | **App development using cloud**  **computing** |  |  |  |

**Practical No: 1**

**Aim: Program on Remote Process Communication Description:**

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.

**TCP in Remote Processing:**

If we are creating a connection between client and server using TCP then it has a few functionalities like, TCP is suited for applications that require high reliability, and transmission time is relatively less critical. It is used by other protocols like HTTP, HTTPs, FTP, SMTP, and Telnet.

TCP rearranges data packets in the order specified. There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent. TCP does Flow Control and requires three packets to set up a socket connection before any user data can be sent.

TCP handles reliability and congestion control. It also does error checking and error recovery. Erroneous packets are retransmitted from the source to the destination

**UDP in Remote Processing:**

In UDP, the client does not form a connection with the server like in TCP and instead just sends a datagram. Similarly, the server need not accept a connection and just waits for datagrams to arrive. Datagrams upon arrival contain the address of the sender which the server uses to send data to the correct client.

**Practical** - **1.1 Aim: Write a program to get host address**

**Code:**

import java.io.\*; import java.net.\*;

public class IP

{

public static void main(String args[])

{

try{

}

InetAddress ip = InetAddress.getByNam[e("w](http://www.facebook.com/)ww[.faceboo](http://www.facebook.com/)k[.com](http://www.facebook.com/)"); System.out.println("Host Names: "+ip.getHostName()); System.out.println("Host Address: "+ip.getAddress());

catch(UnknownHostException u)

{

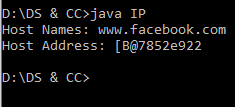
System.out.println(u);

}

}

}

**Output:**



**Practical – 1.2**

**Aim: Write a socket program to connect client and server to TCP**

**Code: Client.java** import java.io.\*; import java.net.\*;

public class Client

{

private Socket socket=null;

private DataInputStream input=null; private DataOutputStream out=null;

public Client(String address,int port)

{

try

{

}

socket=new Socket(address,port); System.out.println("Connected");

input =new DataInputStream(System.in);

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

catch(UnknownHostException uh)

{

System.out.println(uh);

}

catch(IOException i)

{

System.out.println(i);

}

String line=""; while(!line.equals("Over"))

{

try

{

}

line=input.readLine(); out.writeUTF(line);

catch(IOException i)

{

System.out.println(i);

}

}

try

{

}

socket.close(); input.close(); out.close();

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Client client=new Client("127.0.0.1",5000);

}

}

**Server.java** import java.net.\*; import java.io.\*; public class server

{

private Socket socket = null;

private ServerSocket sersocket = null; private DataInputStream in = null; public server(int port)

{

try{

sersocket = new ServerSocket(port); System.out.println("Server Started"); System.out.println("Waiting for a client..."); socket = sersocket.accept(); System.out.println("Client Accepted");

//takes input from client socket in = new DataInputStream(new

BufferedInputStream(socket.getInputStream()));

String line = "";

//reads message from client until "Over" is input while(!line.equals("Over"))

{

try{

}

line = in.readUTF(); System.out.println(line);

catch(IOException i)

{

System.out.println(i);

}

}

//close connection socket.close(); in.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

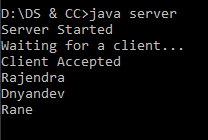
{

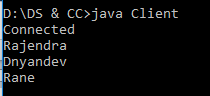
server s = new server(5000);

}

}

Output:





### Practical - 1.3

#### Aim: Write a Socket Program to connect Client and Server through UDP Code:

**UDPClient.java**

import java.net.\*; public class UDPClient

{

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

{

DatagramSocket ds = new DatagramSocket(12345);

InetAddress address = InetAddress.getByName("localhost"); String message = "Hey";

DatagramPacket sendPacket = new DatagramPacket(message.getBytes(),message.length(), address, 1234);

ds.send(sendPacket);

// to recieve message from server byte[] buffer = new byte[1024];

DatagramPacket receivePacket = new DatagramPacket(buffer,buffer.length); ds.receive(receivePacket);

//ds.close();

String msg = new String(receivePacket.getData(),0,receivePacket.getLength());

System.out.println("Message from server: "+msg);

}

}

**UDPServer.java** import java.net.\*; public class UDPServer

{

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

{

DatagramSocket ds = new DatagramSocket(1234); byte[] buffer = new byte[1024];

DatagramPacket dp = new DatagramPacket(buffer, buffer.length); ds.receive(dp);

//ds.close();

String msg = new String(dp.getData(),0,dp.getLength());

System.out.println("Message from client"+msg);

// Retrieving the Address of client

InetAddress clientAddress = dp.getAddress(); int clientPort = dp.getPort();

String response = "Hello"; DatagramPacket outMessage = new

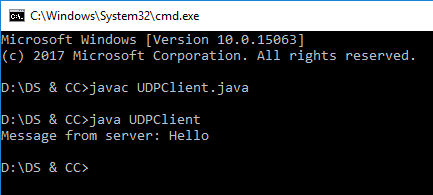
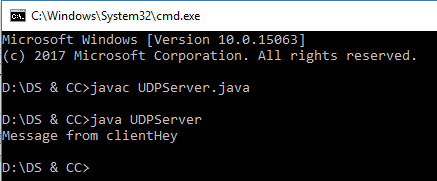
DatagramPacket(response.getBytes(),response.length(), clientAddress, clientPort);

ds.send(outMessage);

}

}

**Output**



**Practical 1.4**

**Aim:** Develop a program for Client server chat using java socket.

**Code: chatClient.java** import java.net.\*; import java.util.\*; import java.io.\*; public class chatClient

{

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

{

String clientMsg,serverMsg;

Socket s = new Socket("localhost",5000);

DataInputStream in = new DataInputStream(s.getInputStream()); DataOutputStream out = new DataOutputStream(s.getOutputStream()); Scanner input = new Scanner(System.in);

while(true)

{

}

s.close();

}

}

System.out.println("Client:"); clientMsg=input.nextLine(); out.writeUTF(clientMsg); serverMsg=in.readUTF(); System.out.println("Server:"+serverMsg); if(serverMsg.equals("exit"))

break;

**//chatServer.java** import java.net.\*; import java.util.\*; import java.io.\*; public class chatServer

{

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

{

ServerSocket ss = new ServerSocket(5000); Socket s = ss.accept();

String str;

DataInputStream in = new DataInputStream(s.getInputStream()); DataOutputStream out = new DataOutputStream(s.getOutputStream()); Scanner input = new Scanner(System.in);

while(true)

{

str=in.readUTF(); if(str.equals("exit"))

{

out.writeUTF("exit"); break;

}

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

//server message to client System.out.println("Server:"); str=input.nextLine(); out.writeUTF(str);

}

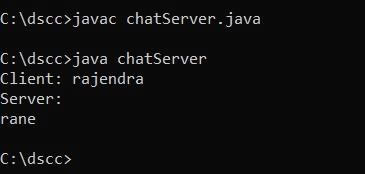
ss.close();

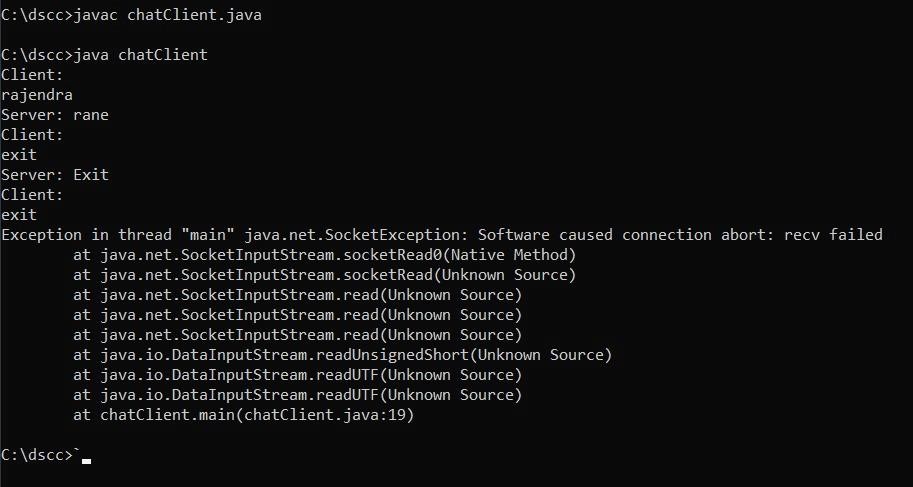
s.close();

}

}

**Output:**





**Practical 1.5**

**Aim:** Develop a program for client-server GUI chat

# Code:

**Client.java** import java.net.\*; import java.io.\*;

import javax.swing.\*; import java.awt.\*; import java.awt.event.\*;

public class Client extends JFrame

{

Socket socket; BufferedReader br; PrintWriter out;

//Declare Components

private JLabel heading = new JLabel("Client"); private JTextArea messageArea = new JTextArea(); private JTextField messageInput = new JTextField();

private Font font = new Font("Times New Roman",Font.PLAIN,25);

//constructor public Client()

{

try

{

}

System.out.println("Sending request to server"); socket = new Socket("127.0.0.1",5000); System.out.println("Connection done..");

br = new BufferedReader(new InputStreamReader(socket.getInputStream())); out = new PrintWriter(socket.getOutputStream());

createGUI(); handleEvents(); startReading();

// startWriting();

catch(Exception e)

{

e.printStackTrace();

}

}

` public void createGUI()

{

this.setTitle("Client Messanger[END]"); this.setSize(600,600); this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); this.setVisible(true);

//coding for component heading.setFont(font);

heading.setHorizontalAlignment(SwingConstants.CENTER); heading.setBorder(BorderFactory.createEmptyBorder(20,20,20,20)); messageArea.setEditable(false);

messageArea.setFont(font); messageInput.setFont(font);

//set layout of frame this.setLayout(new BorderLayout());

//adding components to frame this.add(heading,BorderLayout.NORTH); this.add(messageArea,BorderLayout.CENTER); this.add(messageInput,BorderLayout.SOUTH);

}

public void handleEvents()

{

messageInput.addKeyListener(new KeyListener(){ @Override

public void keyTyped(KeyEvent e)

{

}

@Override

public void keyPressed(KeyEvent e)

{

}

@Override

public void keyReleased(KeyEvent e)

{

//System.out.println("Key released "+ e.getKeyCode()); if(e.getKeyCode() == 10)

{

}

});

}

//System.out.println("You have pressed enter button."); String contentToSend = messageInput.getText(); messageArea.append("Me :"+contentToSend+"\n"); out.println(contentToSend);

out.flush(); messageInput.setText(""); messageInput.requestFocus();

}

//Method

public void startReading()

{

//thread- read data Runnable r1 = () ->{

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

try

{

while(true)

{

String msg = br.readLine(); if(msg.equals("exit"))

{

System.out.println("Server terminated the chat!"); JOptionPane.showMessageDialog(this,"Server terminated the chat!"); messageInput.setEnabled(false);

socket.close(); out.close(); break;

}

//System.out.println("Server: " +msg); messageArea.append("Server: " +msg+ "\n");

}

}

catch(Exception e)

{

System.out.println("Connection is closed!");

}

};

new Thread(r1).start();

}

//Method

public void startWriting()

{

Runnable r2 = ()->{ System.out.println("Writer Started.."); try

{

while(!socket.isClosed())

{

BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));

String content = br1.readLine(); out.println(content); out.flush(); if(content.equals("exit"))

{

socket.close(); break;

}

}

}

catch(Exception e)

{

System.out.println("Connection is closed!");

}

};

new Thread(r2).start();

}

public static void main(String args[])

{

System.out.println("This is Client"); new Client();

}

}

**Server.java** import java.net.\*; import java.io.\*;

import javax.swing.\*; import java.awt.\*; import java.awt.event.\*;

class Server extends JFrame

{

ServerSocket server;

Socket socket;

BufferedReader br;

PrintWriter out;

//Declare Components

private JLabel heading = new JLabel("Server"); private JTextArea messageArea = new JTextArea(); private JTextField messageInput = new JTextField();

private Font font = new Font("Times New Roman",Font.PLAIN,25);

//constructor public Server()

{

try

{

}

server = new ServerSocket(5000); System.out.println("Server is ready to accept connection"); System.out.println("Waiting. ");

socket = server.accept();

br = new BufferedReader(new InputStreamReader(socket.getInputStream())); out = new PrintWriter(socket.getOutputStream());

createGUI(); handleEvents(); startReading();

// startWriting();

catch(Exception e)

{

e.printStackTrace();

}

}

public void createGUI()

{

this.setTitle("Server Messanger[END]"); this.setSize(600,600); this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); this.setVisible(true);

//coding for component heading.setFont(font);

heading.setHorizontalAlignment(SwingConstants.CENTER); heading.setBorder(BorderFactory.createEmptyBorder(20,20,20,20)); messageArea.setEditable(false);

messageArea.setFont(font); messageInput.setFont(font);

//set layout of frame this.setLayout(new BorderLayout());

//adding components to frame this.add(heading,BorderLayout.NORTH); this.add(messageArea,BorderLayout.CENTER); this.add(messageInput,BorderLayout.SOUTH);

}

public void handleEvents()

{

messageInput.addKeyListener(new KeyListener(){ @Override

public void keyTyped(KeyEvent e)

{

}

@Override

public void keyPressed(KeyEvent e)

{

}

@Override

public void keyReleased(KeyEvent e)

{

//System.out.println("Key released "+ e.getKeyCode()); if(e.getKeyCode() == 10)

{

//System.out.println("You have pressed enter button."); String contentToSend = messageInput.getText(); messageArea.append("Me :"+contentToSend+"\n"); out.println(contentToSend);

out.flush(); messageInput.setText(""); messageInput.requestFocus();

}

}

});

}

public void startReading()

{

//thread- read data Runnable r1 = () ->

{

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

{

while(true)

{

String msg = br.readLine(); if(msg.equals("exit"))

{

chat!");

System.out.println("Client terminated the chat!"); JOptionPane.showMessageDialog(this,"Client terminated the

messageInput.setEnabled(false); socket.close();

break;

}

//System.out.println("Client: " +msg); messageArea.append("Client: " +msg+ "\n");

}

}

catch(Exception e)

{

System.out.println("Connection is closed!");

}

};

new Thread(r1).start();

}

public void startWriting()

{

//thread- take data from user and send that data to client Runnable r2 = ()->

{

System.out.println("Writer Started.."); try

{

while(!socket.isClosed())

{

BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));

String content = br1.readLine();

out.println(content); out.flush(); if(content.equals("exit"))

{

socket.close(); break;

}

}

}

catch(Exception e)

{

System.out.println("Connection is closed!");

}

};

new Thread(r2).start();

}

public static void main(String args[])

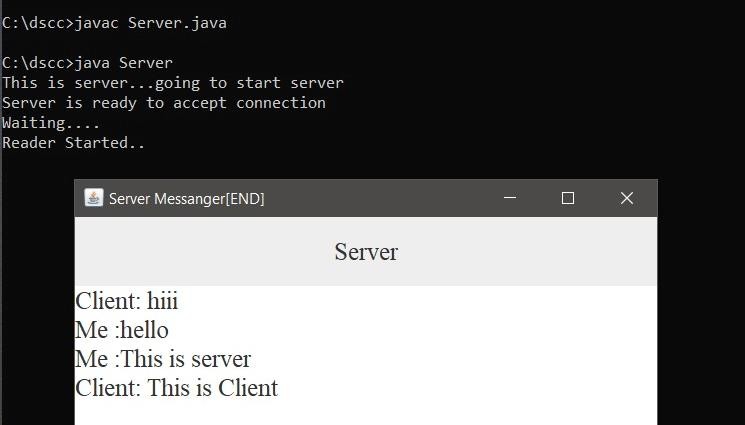
{

System.out.println("This is server...going to start server"); new Server();

}

}

# Output:





**Practical 1.6**

**Aim:** Write a program to implement a sever which calculates sum of two number using java socket

**Code: AddClient.java** import java.io.\*; import java.net.\*; import java.util.\*; public class AddClient

{

private static String host;

private static final int PORT = 5000;

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

{

try

{

}

host = InetAddress.getLocalHost().getHostName();

catch(UnknownHostException u)

{

System.out.println("Host Not Found"); System.exit(1);

}

Socket s=new Socket(host, PORT); int a, b;

DataInputStream in=new DataInputStream(s.getInputStream()); DataOutputStream out=new DataOutputStream(s.getOutputStream()); Scanner sc=new Scanner(System.in);

System.out.println("Enter first number: "); a=sc.nextInt();

System.out.println("Enter second number: "); b=sc.nextInt();

String A=Integer.toString(a); String B=Integer.toString(b); out.writeUTF(A); out.writeUTF(B);

String r=in.readUTF();

int R=Integer.parseInt(r);

System.out.println("The addition of a and b is: "+R); s.close();

}

}

### AddServer.java

import java.io.\*;

import java.net.\*; import java.util.\*;

public class AddServer

{

private static final int PORT = 5000;

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

{

ServerSocket ss=new ServerSocket(PORT); while(true)

{

Socket s=ss.accept();

DataInputStream in=new DataInputStream(s.getInputStream()); DataOutputStream out=new DataOutputStream(s.getOutputStream());

String A=in.readUTF(); int a=Integer.parseInt(A); String B=in.readUTF();

int b=Integer.parseInt(B); int r=a+b;

String R=Integer.toString(r); out.writeUTF(R); out.flush();

ss.close();

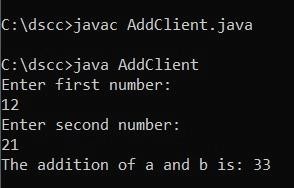
}

}

}

# Output:





**Practical 2**

### Aim: Programs on Remote Procedure Call

Remote Procedure Call or RPC is a powerful technique for constructing distributed, client- server-based applications. It is also known as a function call or a subroutine call. A remote procedure call is when a computer program causes a procedure to execute in a different address space, coded as a local procedure call, without the programmer explicitly stating the details for the remote interaction.

The programmer writes essentially the same code whether the subroutine is local to the executing program or remote. This is a form of client-server interaction implemented via a request-response message-passing system.

### 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.

### Practical 2.1

**Aim**: To implement date time server using RPC concept (Make use of datagram)

#### Code:

* + **DateTimeS.java**

import java.net.\*; import java.io.\*; import java.util.\*; import java.text.\*; public class DateTimeS

{

DateTimeS() throws Exception

{

ServerSocket ss=new ServerSocket(1234); Socket s=ss.accept();

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

out.writeUTF(time()); out.flush();

}

public String date()

{

return new SimpleDateFormat("dd/mm/yyyy").format(new Date()).toString();

}

public String time()

{

return new SimpleDateFormat("dd/mm/yyyy").format(new Date()).toString();

}

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

{

DateTimeS d=new DateTimeS();

}

}

#### DateTimeC.java

import java.net.\*; import java.io.\*; public class DateTimeC

{

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

{

Socket s=new Socket("localhost",1234);

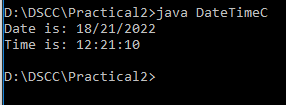
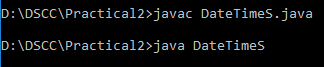
DataInputStream in=new DataInputStream(s.getInputStream()); String dt=in.readUTF();

String tm=in.readUTF(); System.out.println("Date is: "+dt); System.out.println("Time is: "+tm);

}

}

#### Output:



**Practical 2.2 Aim:** Write a Calculator using RPC(Make use of Datagram) Code:

#### RPCCalculatorClient.java

import java.net.\*; import java.io.\*;

class RPCCalculatorClient

{

RPCCalculatorClient()

{

add 12");

try

{

InetAddress ia=InetAddress.getLocalHost(); DatagramSocket ds=new DatagramSocket(); DatagramSocket ds1=new DatagramSocket(1235); System.out.println("\n RPC Client");

System.out.println("Enter Method name With parameter like

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,1234);

ds.send(dp);

dp=new DatagramPacket(b,b.length); ds1.receive(dp);

String s=new String(dp.getData(),0,dp.getLength()); System.out.println("\n Result : "+s+"\n"); System.out.println("\n Enter Method name with

parameter like add 1 2");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

public static void main(String args[])

{

new RPCCalculatorClient();

}

}

#### RPCCalculatorServer.java

import java.net.\*; import java.util.\*;

class RPCCalculatorServer

{

DatagramSocket ds; DatagramPacket dp;

String str,methodName,result; int val1,val2; RPCCalculatorServer()

{

try

{

ds=new DatagramSocket(1234); byte b[]=new byte[4096]; while(true)

{

dp=new DatagramPacket(b,b.length); ds.receive(dp);

str=new String(dp.getData(),0,b.length); if(str.equalsIgnoreCase("quit"))

{

}

else

{

System.exit(1);

StringTokenizer st=new StringTokenizer(str," "); while(st.hasMoreTokens())

{

String token=st.nextToken(); methodName=token; val1=Integer.parseInt(st.nextToken()); val2=Integer.parseInt(st.nextToken());

}

}

System.out.println("/n Client Selected/" +str+ "/n"); System.out.println("\n First Value: "+val1); System.out.println("\n Second Value: "+val2); InetAddress ia=InetAddress.getLocalHost();

if(methodName.equalsIgnoreCase("add")) result=""+add(val1, val2);

else if(methodName.equalsIgnoreCase("sub")) result=""+sub(val1, val2);

else if(methodName.equalsIgnoreCase("mul")) result=""+mul(val1, val2);

else if(methodName.equalsIgnoreCase("div")) result=""+div(val1, val2);

byte b1[]=result.getBytes();

DatagramSocket ds1=new DatagramSocket(); DatagramPacket dp1=new

DatagramPacket(b1,b1.length,InetAddress.getLocalHost(),1235);

System.out.println("Result: " +result+ "\n"); ds1.send(dp1);

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

public int add(int val1, int val2)

{

return val1+val2;

}

public int sub(int val1, int val2)

{

return val1-val2;

}

public int mul(int val1, int val2)

{

return val1\*val2;

}

public int div(int val1, int val2)

{

return val1/val2;

}

public static void main(String args[])

{

new RPCCalculatorServer();

}

}

# Practical 3 Aim: Programs on remote Method Invocation

**Description:**

Remote Method Invocation (RMI) is an API that allows an object to invoke a method on an object that exists in another address space, which could be on the same machine or on a remote machine. Through RMI, an object running in a JVM present on a computer (Client-side) can invoke methods on an object present in another JVM (Server-side). RMI creates a public remote server object that enables client and server-side communications through simple method calls on the server object.

**Stub Object:** The stub object on the client machine builds an information block and sends this information to the server.

The block consists of

* An identifier of the remote object to be used
* Method name which is to be invoked
* Parameters to the remote JVM

**Practical 3.1**

**Aim:** Retrieve time and date function from server to client. This program should display server date and time by implementing RMI **.**

### Code:

* + **DateTimeInterface.java**

import java.rmi.\*;

public interface DateTimeInterface extends Remote

{

public String getDate() throws RemoteException; public String getTime() throws RemoteException;

}

### DTI.java

import java.rmi.\*; import java.rmi.server.\*; import java.util.\*; import java.text.\*;

public class DTI extends UnicastRemoteObject implements DateTimeInterface

{

DTI() throws RemoteException

{

super();

}

public String getDate() throws RemoteException

{

return new SimpleDateFormat("dd/mm/yyyy").format(new Date()).toString();

}

public String getTime() throws RemoteException

{

return new SimpleDateFormat("hh:mm:ss").format(new Date()).toString();

}

}

### DateTimeServer.java

import java.rmi.\*;

import java.rmi.registry.\*; public class DateTimeServer

{

public static void main(String args[])

{

try

{

}

Registry r=LocateRegistry.createRegistry(1234); DTI d=new DTI();

Naming.rebind("DTI",d);

catch(Exception e)

{

e.printStackTrace();

}

}

}

### DateTimeClient.java

import java.rmi.\*;

public class DateTimeClient

{

public static void main(String args[])

{

try

{

}

DateTimeInterface dti=(DateTimeInterface)Naming.lookup("DTI"); System.out.println("Date is:"+dti.getDate()); System.out.println("Time is:"+dti.getTime());

catch(Exception e)

{

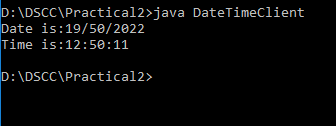
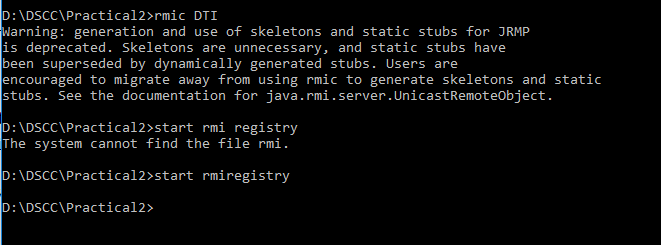
System.out.println(e);

}

}

}

### Output:



**Practical 3.2**

**Aim:** The client will enter two numbers a and b the server will solve the equation c=(a+b )^2 and c=(a+b) ^ 3 and will give back the value of c Implement using RMI.

### Code:

* **EQInterface.java**

import java.rmi.\*;

public interface EQInterface extends Remote

{

public double square(int a,int b) throws RemoteException; public double cube(int a,int b) throws RemoteException;

}

### EQI.java

import java.rmi.\*; import java.rmi.server.\*; import java.util.\*;

public class EQI extends UnicastRemoteObject implements EQInterface

{

public EQI() throws RemoteException

{

super();

}

public double square(int a,int b) throws RemoteException

{

double ans=(a\*a)+(b\*b)+(2\*a\*b); return ans;

}

public double cube(int a,int b) throws RemoteException

{

double ans=(a\*a\*a)+(b\*b\*b)+3\*a\*b\*(a\*b); return ans;

}

}

### EQServer.java

import java.rmi.\*;

import java.rmi.registry.\*; public class EQServer

{

public static void main(String args[])

{

try

{

}

catch

{

}

}

}

Registry reg = LocateRegistry.createRegistry(4444); EQI e = new EQI();

Naming.rebind("EQI",e);

System.out.println("Error");

### EQClient.java

import java.rmi.\*; import java.util.\*; public class EQClient

{

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

{

}

}

### Output:

int a,b;

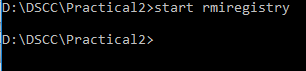
double ans1,ans2;

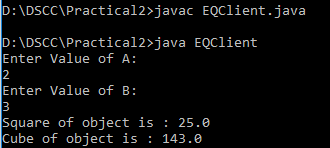
Scanner input = new Scanner(System.in); EQInterface e = (EQInterface)Naming.lookup("EQI"); System.out.println("Enter Value of A: "); a=input.nextInt();

System.out.println("Enter Value of B: "); b=input.nextInt();

ans1=e.square(a,b); ans2=e.cube(a,b);

System.out.println("Square of object is : "+ans1); System.out.println("Cube of object is : "+ans2);





### Practical 3.3

**Aim:** Design a graphical user interface to find greatest of two number implements using RMI

### Code:

* **GreatestInterface.java**

import java.rmi.\*;

public interface GreatestInterface extends Remote

{

public int find(int a,int b) throws RemoteException;

}

### GI.java

import java.rmi.\*; import java.rmi.server.\*;

public class GI extends UnicastRemoteObject implements GreatestInterface

{

public GI() throws RemoteException

{

super();

}

public int find(int a,int b) throws RemoteException

{

}

}

#### GServer.java

if(a>b) else

return a; return b;

import java.rmi.\*;

import java.rmi.registry.\*; public class GServer

{

public static void main(String args[])

{

try

{

}

Registry reg = LocateRegistry.createRegistry(3333); GI g = new GI();

Naming.rebind("GI",g);

catch(Exception e)

{

}

}}

### GClient.java

System.out.println(e);

import java.rmi.\*; import java.awt.\*; import java.awt.event.\*; import javax.swing.\*;

public class GClient extends JFrame implements ActionListener

{

JTextField tf1,tf2; JLabel lb1,lb2,lb3; JButton btn; GClient()

{

tf1=new JTextField(); tf2=new JTextField();

lb1=new JLabel("Enter First Number: "); lb2=new JLabel("Enter Second Number: "); lb3=new JLabel("");

btn=new JButton("Find Greatest");

add(lb1);

add(tf1);

add(lb2);

add(tf2);

add(lb3);

add(btn);

btn.addActionListener(this);

}

public void actionPerformed(ActionEvent ae)

{

try

{

GreatestInterface g=(GreatestInterface)Naming.lookup("GI"); lb3.setText("Here");

int a=Integer.parseInt(tf1.getText()); int b=Integer.parseInt(tf2.getText()); int c=g.find(a,b);

lb3.setText(c+" is greatest number");

}

catch(Exception e)

{

lb3.setText("Error");

}

}

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

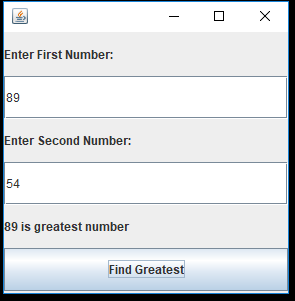
{

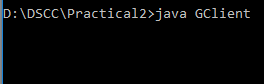
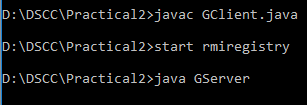
GClient gc=new GClient(); gc.setLayout(new GridLayout(6,1)); gc.setVisible(true); gc.setSize(300,300);

}

}

### Output:





**Practical 3.4**

**Aim:** The client should provide the values of a, b & c. The server will solve the equation (aX2 + bx + c = 0) and will give back the value of x.

### Code:

* **EqnInterface.java:**

import java.rmi.\*;

public interface EqnInterface extends Remote

{

public double equn(int a,int b,int c) throws RemoteException;

}

### EQNI.java:

import java.rmi.\*; import java.rmi.server.\*; import java.util.\*;

public class EQNI extends UnicastRemoteObject implements EqnInterface

{

public EQNI() throws RemoteException

{

super();

}

public double equn(int a,int b,int c) throws RemoteException

{

double det = b\*b-4\*a\*c;

double x1 = (-b+Math.pow(det,0.5))/(2.0\*a); return x1;

}

}

### EQNServer.java:

import java.rmi.\*;

import java.rmi.registry.\*; public class EQNServer

{

public static void main(String args[])

{

try

{

Registry reg = LocateRegistry.createRegistry(4444); EQNI e = new EQNI();

Naming.rebind("EQNI",e);

}

catch(Exception e)

{

System.out.println("Error");

}

}

}

### EQNClient.java:

import java.rmi.\*; import java.util.\*; public class EQNClient

{

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

{

int a,b,c; double ans;

Scanner input = new Scanner(System.in);

EqnInterface e = (EqnInterface)Naming.lookup("EQNI"); System.out.println("Enter Value of A: "); a=input.nextInt();

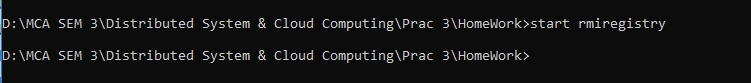
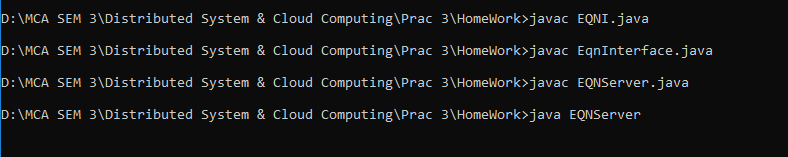
System.out.println("Enter Value of B: "); b=input.nextInt(); System.out.println("Enter value of C: "); c=input.nextInt();

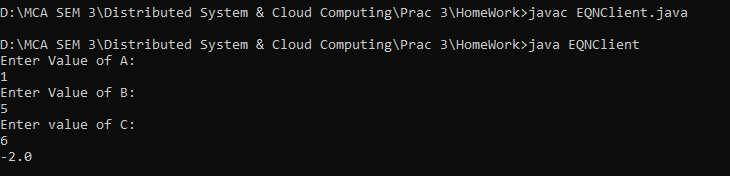
### Output:

ans = e.equn(a,b,c); System.out.println(ans);

}

}





### Practical 4 AIM: Practical on Remote Public Communication Description:

RMI uses stub and skeleton object for communication with the remote object.

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.

## skeleton

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it.

1. Retrive the student score from the database (use concept of JDBC and RMI)

# Code:

#### //StudentScoreInterface.java

package practical4;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface StudentScoreInterface extends Remote{

public double findScore(String Name) throws RemoteException;

}

#### //SSI.java

package practical4;

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;

public class SSI extends UnicastRemoteObject implements StudentScoreInterface { private PreparedStatement pst;

public SSI() throws RemoteException

{

super(); initializeDB();

}

protected void initializeDB()

{

try

{

Class.forName("com.mysql.jdbc.Driver"); Connection conn =

DriverManager.getConnection("jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=convertT oNull [root on Default schema]");

pst=conn.prepareStatement("Select \* from student where Name=?");

}

catch(Exception e)

{

System.out.println(e);

}

}

@Override

public double findScore(String Name) throws RemoteException {

double score = -1; try

{

pst.setString(1, Name);

ResultSet rs = pst.executeQuery(); if(rs.next())

{

score = rs.getDouble("Score:");

}

}

catch(Exception e)

{

System.out.println(e);

}

System.out.println(score); return score;

}

public static void main(String args[])

{

}

}

#### //StudentServer.java

package practical4;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry;

public class StudentServer {

public static void main(String args[])

{

try

{

SSI s = new SSI();

Registry r = LocateRegistry.getRegistry(); Naming.rebind("SSI", r);

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**StudentClient.java** package practical4; import java.rmi.\*; import javax.swing.\*; import java.awt.\*; import java.awt.event.\*;

public class StudentClient extends JApplet{ private StudentScoreInterface student;

private JButton getScore=new JButton("Get Score"); private JTextField jfname= new JTextField();

private JTextField jfscore= new JTextField(); private JLabel lblname = new JLabel("Name"); private JLabel lblscore = new JLabel("Score");

public void init()

{

try

{

student = (StudentScoreInterface)Naming.lookup("SSI");

}

catch(Exception e)

{

}

JPanel panel = new JPanel();

panel.setLayout(null); lblname.setBounds(20,20,80,34); panel.add(lblname); jfname.setBounds(100,20,160,34); panel.add(jfname); lblscore.setBounds(20,70,80,34); panel.add(lblscore); jfscore.setBounds(100,70,160,34); panel.add(jfscore); getScore.setBounds(40, 100, 80, 35);

panel.add(getScore,BorderLayout.CENTER); add(panel,BorderLayout.CENTER);

getScore.addActionListener(new ActionListener() { @Override

public void actionPerformed(ActionEvent ae) { getScore();

}

});

}

public void getScore()

{

try

{

double score = student.findScore(jfname.getText()); if(score < 0)

{

jfscore.setText("Not Found");

}

else

{

jfscore.setText(""+score);

}

}

catch(Exception e)

{

}

}

public static void main(String args[])

{

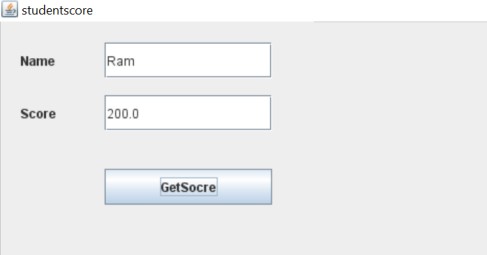
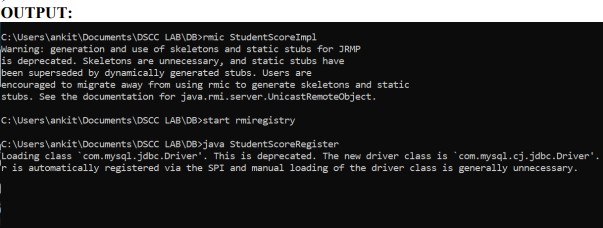
StudentClient sc = new StudentClient(); JFrame frame = new JFrame();

frame.setTitle("Student Score"); frame.add(sc,BorderLayout.CENTER); frame.setSize(300,240);

sc.init(); frame.setVisible(true);

}

}



**Practical 5 Aim: Practical based on TokenRingClient-Server Description:**

What is Mutual exclusion:

**Mutual exclusion** is a concurrency control property which is introduced to prevent race conditions. It is the requirement that a process can not enter its critical section while another concurrent process is currently present or executing in its critical section i.e only one process is allowed to execute the critical section at any given instance of time.

### Code:

**TokenRingClient1.java** import java.net.\*; import java.io.\*;

public class TokenRingClient1

{

public static DatagramSocket ds; public static DatagramPacket dp; public static BufferedReader br;

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

{

boolean hasToken = true; ds=new DatagramSocket(2001); while(true){

if(hasToken==true)

{

System.out.println("do you want to enterCS(yes/no)"); br=new BufferedReader(new InputStreamReader(System.in)); String choice=br.readLine(); if(choice.equalsIgnoreCase("yes"))

{

System.out.println("Client is ready to write:"); System.out.println("Enter the message");

br=new BufferedReader(new InputStreamReader(System.in)); String msg="1"+br.readLine();

dp=new DatagramPacket(msg.getBytes(),msg.length(),InetAddress.getLocalHost(),2000); ds.send(dp);

System.out.println("Message Sent");

}

else if(choice.equalsIgnoreCase("no"))

{

System.out.println("I am not ready to enter the CS"); String msg1="Token";

dp=new DatagramPacket(msg1.getBytes(),msg1.length(),InetAddress.getLocalHost(),2002); ds.send(dp);

hasToken=false;

}

}

else

{

System.out.println("Waiting for token"); byte[] buffer=new byte[2048];

dp = new DatagramPacket(buffer,buffer.length); ds.receive(dp);

String PrevProcessMsg=new String(dp.getData(),0,dp.getLength()); System.out.println("Previous Process Message is"+PrevProcessMsg); if(PrevProcessMsg.equals("Token"))

{

hasToken=true;

System.out.println("I have token now");

}

}

}

}

}

**TokenRingClient2.java** import java.net.\*; import java.io.\*;

public class TokenRingClient2

{

public static DatagramSocket ds; public static DatagramPacket dp; public static BufferedReader br;

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

{

boolean hasToken = true; ds=new DatagramSocket(2002); while(true){

if(hasToken==true)

{

System.out.println("do you want to enterCS(yes/no)"); br=new BufferedReader(new InputStreamReader(System.in)); String choice=br.readLine(); if(choice.equalsIgnoreCase("yes"))

{

System.out.println("Client is ready to write:"); System.out.println("Enter the message");

br=new BufferedReader(new InputStreamReader(System.in)); String msg="2|"+br.readLine();

dp=new DatagramPacket(msg.getBytes(),msg.length(),InetAddress.getLocalHost(),2000); ds.send(dp);

System.out.println("Message Sent");

}

else if(choice.equalsIgnoreCase("no"))

{

System.out.println("I am not ready to enter the CS"); String msg1="Token";

dp=new DatagramPacket(msg1.getBytes(),msg1.length(),InetAddress.getLocalHost(),2003); ds.send(dp);

hasToken=false;

}

}

else

{

System.out.println("Waiting for token"); byte[] buffer=new byte[2048];

dp = new DatagramPacket(buffer,buffer.length); ds.receive(dp);

String PrevProcessMsg=new String(dp.getData(),0,dp.getLength()); System.out.println("Previous Process Message is"+PrevProcessMsg); if(PrevProcessMsg.equals("Token"))

{

hasToken=true;

System.out.println("I have token now");

}

}

}

}

}

**TokenRingClient3.java** import java.net.\*; import java.io.\*;

public class TokenRingClient3

{

public static DatagramSocket ds; public static DatagramPacket dp; public static BufferedReader br;

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

{

boolean hasToken = true; ds=new DatagramSocket(2003); while(true){

if(hasToken==true)

{

System.out.println("do you want to enterCS(yes/no)"); br=new BufferedReader(new InputStreamReader(System.in)); String choice=br.readLine(); if(choice.equalsIgnoreCase("yes"))

{

System.out.println("Client is ready to write:"); System.out.println("Enter the message");

br=new BufferedReader(new InputStreamReader(System.in)); String msg="3|"+br.readLine();

dp=new DatagramPacket(msg.getBytes(),msg.length(),InetAddress.getLocalHost(),2000); ds.send(dp);

System.out.println("Message Sent");

}

else if(choice.equalsIgnoreCase("no"))

{

System.out.println("I am not ready to enter the CS"); String msg1="Token";

dp=new DatagramPacket(msg1.getBytes(),msg1.length(),InetAddress.getLocalHost(),2001); ds.send(dp);

hasToken=false;

}

}

else

{

System.out.println("Waiting for token"); byte[] buffer=new byte[2048];

dp = new DatagramPacket(buffer,buffer.length); ds.receive(dp);

String PrevProcessMsg=new String(dp.getData(),0,dp.getLength()); System.out.println("Previous Process Message is"+PrevProcessMsg); if(PrevProcessMsg.equals("Token"))

{

hasToken=true;

System.out.println("I have token now");

}

}

}

}

}

### TokenRingServer.java

import java.net.\*;

public class TokenRingServer

{

public static DatagramSocket ds; public static DatagramPacket dp; public static void main(String args[])

{

try

{

ds = new DatagramSocket(2000); while(true)

{

byte[] buffer=new byte[1024];

dp=new DatagramPacket(buffer,buffer.length); ds.receive(dp);

String msg=new String(dp.getData(),0,dp.getLength()); System.out.println("Message from client"+msg);

}

}

catch(Exception e)

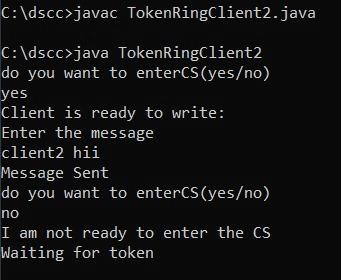
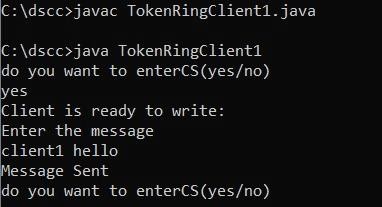
{

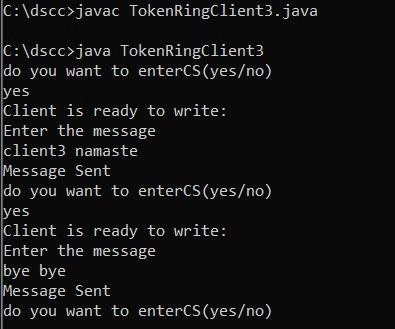
}

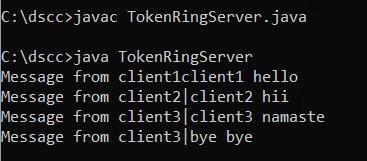
}

}

Output:







# PRACTICAL 6

#### AIM: Implementation of Cloud Computing Services Implementation of Storage as a Service using Google Docs.

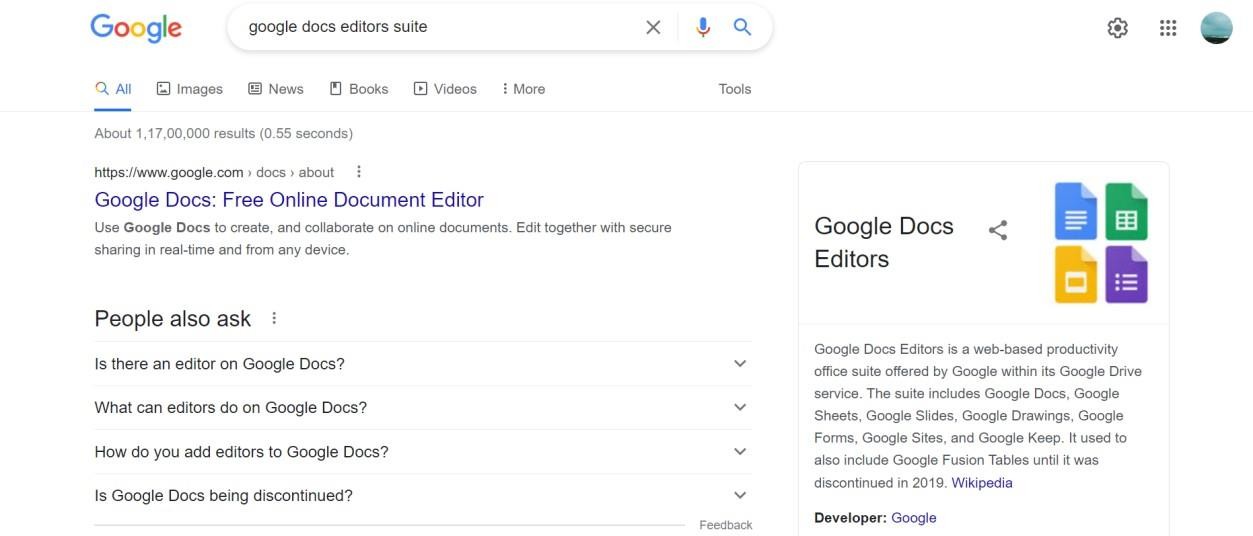
**Description:**

What is cloud storage:

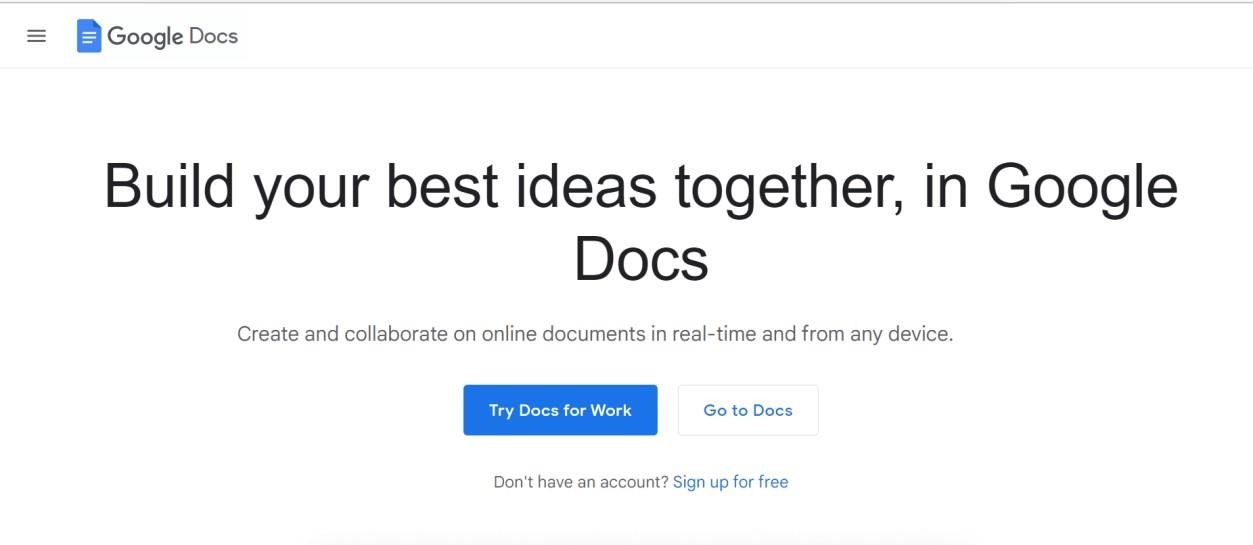
Cloud Storage is a service for storing your [*objects*](https://cloud.google.com/storage/docs/objects) in Google Cloud. An object is an immutable piece of data consisting of a file of any format. You store objects in containers called [*buckets*](https://cloud.google.com/storage/docs/buckets). All buckets are associated with a [*project*](https://cloud.google.com/storage/docs/projects), and you can group your projects under an [*organization*](https://cloud.google.com/resource-manager/docs/cloud-platform-resource-hierarchy#organizations). Each project, bucket, and object in Google Cloud is a resource in Google Cloud, as are things such as [Compute](https://cloud.google.com/compute/docs/instances) [Engine instances](https://cloud.google.com/compute/docs/instances).

#### Creating Google Docs

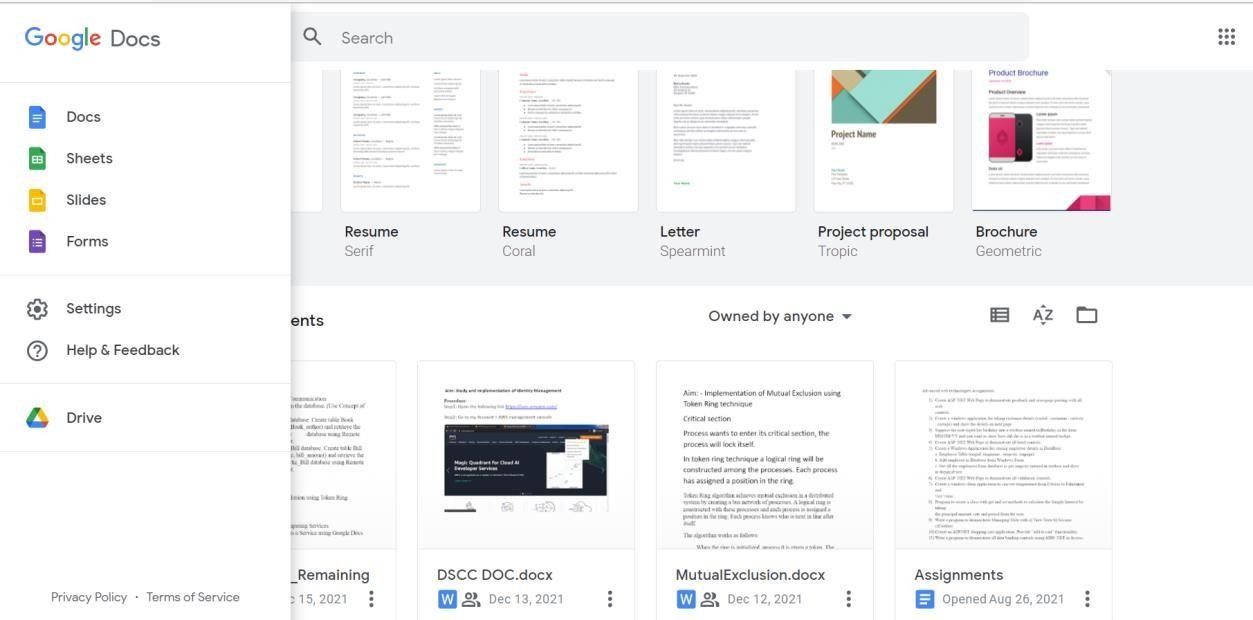
Step 1: In Web Browser Search Bar, write ‘Google Docs Editors Suite’ and press Enter. Following results will be shown, click on below highlighted link for Google Docs.



Step 2: Below page of ‘Google Docs’ will be opened. Click on the ‘Go to Docs’ button.

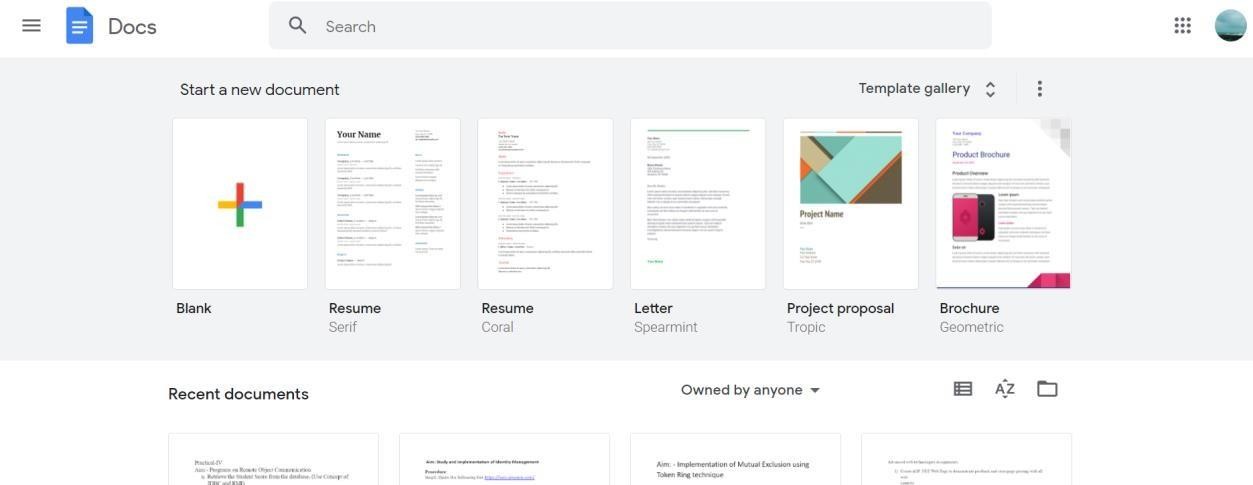


Step 3: After Signing In, following Start Page in Google Docs will be opened. You can Switch between Docs/Sheets/Slides/Forms by clicking on their options given in Side Bar.



**A)** Creating Google Docs

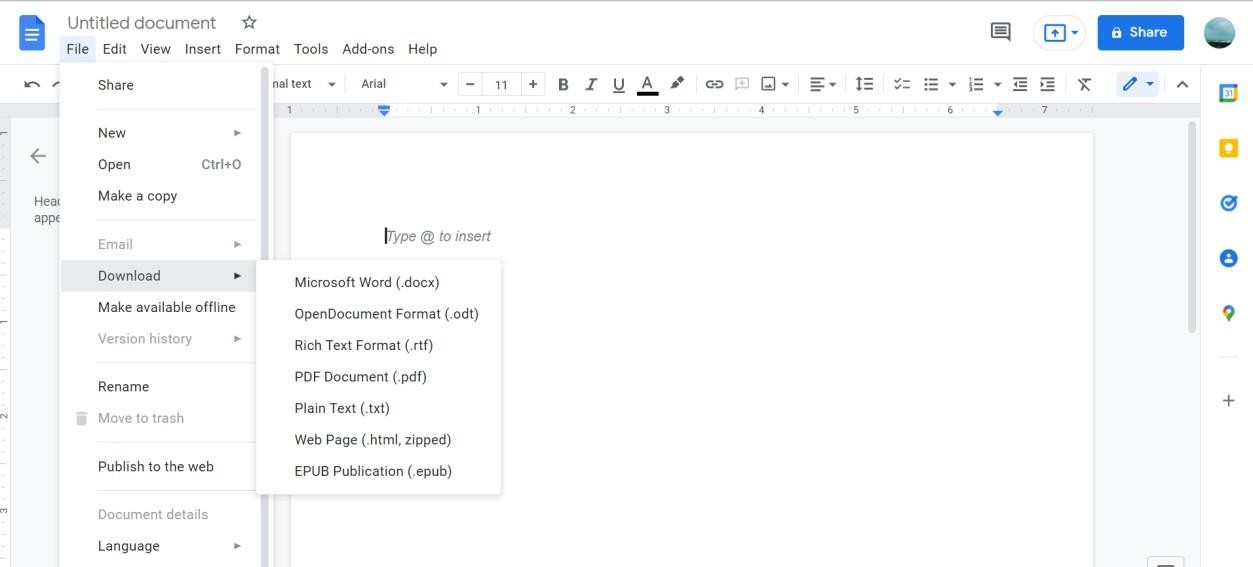
Step 1: From the ‘Start a new document’ area click on ‘+’ sign named ‘Blank’ to Create a New Google Doc File. Also, ready Templates are available for various kinds of documents here which can be used by modifying as per needed.



Step 2: Google Docs File will get opened. We can Insert and Edit the Document file as desired using

various functionality provided by Google Docs and they are AutoSaved in the Cloud (Google Drive) of

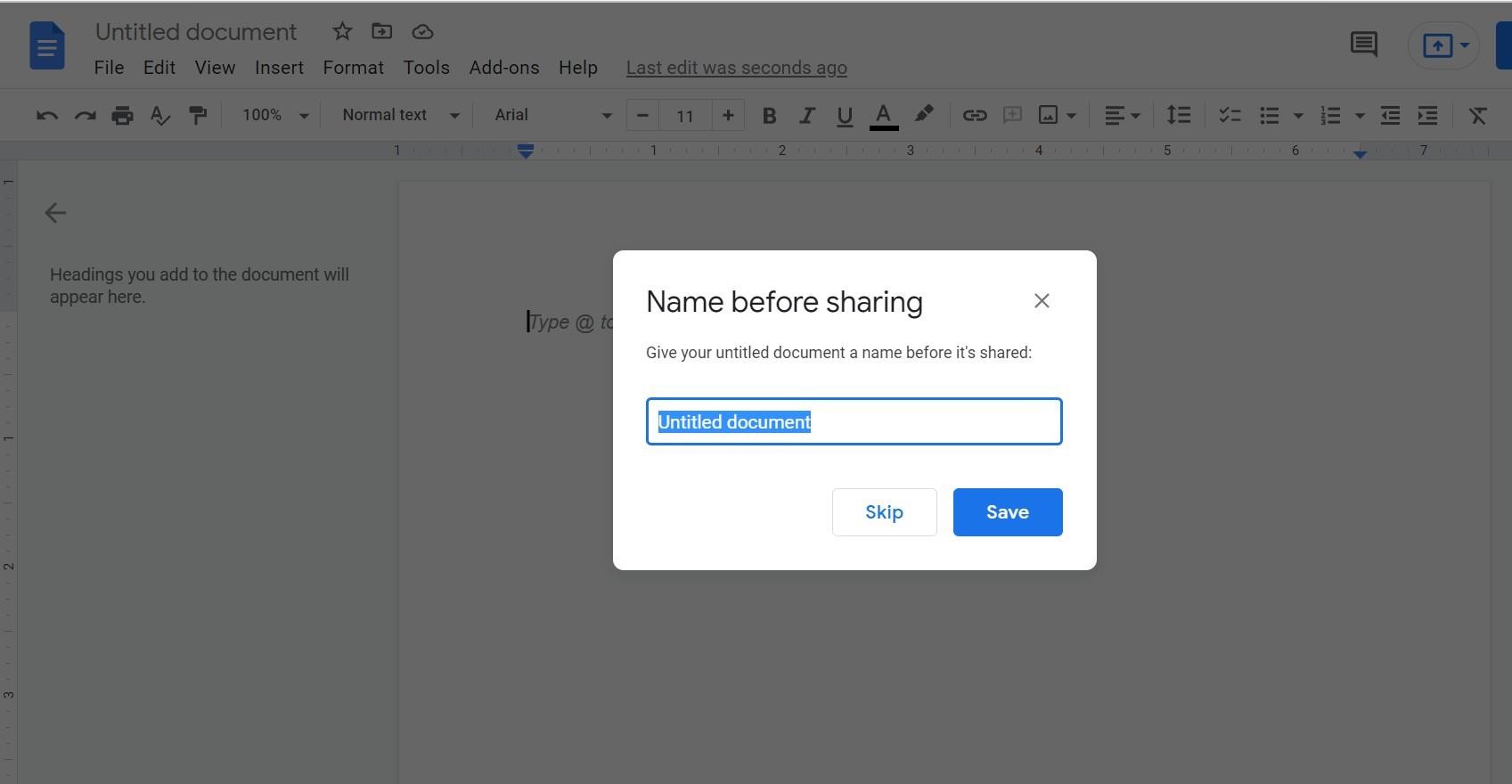
User Account. After which we can Export the file created or Share the Document online withhelp of the below shown option.

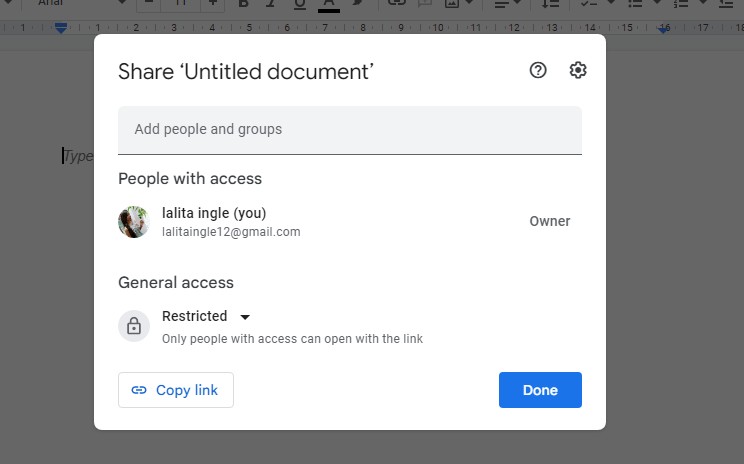


.Step 3: Google Docs File can be shared using various ways by Importing File Offline or bySharing with other Users Online. When clicked on Share button, first we must Name the File.

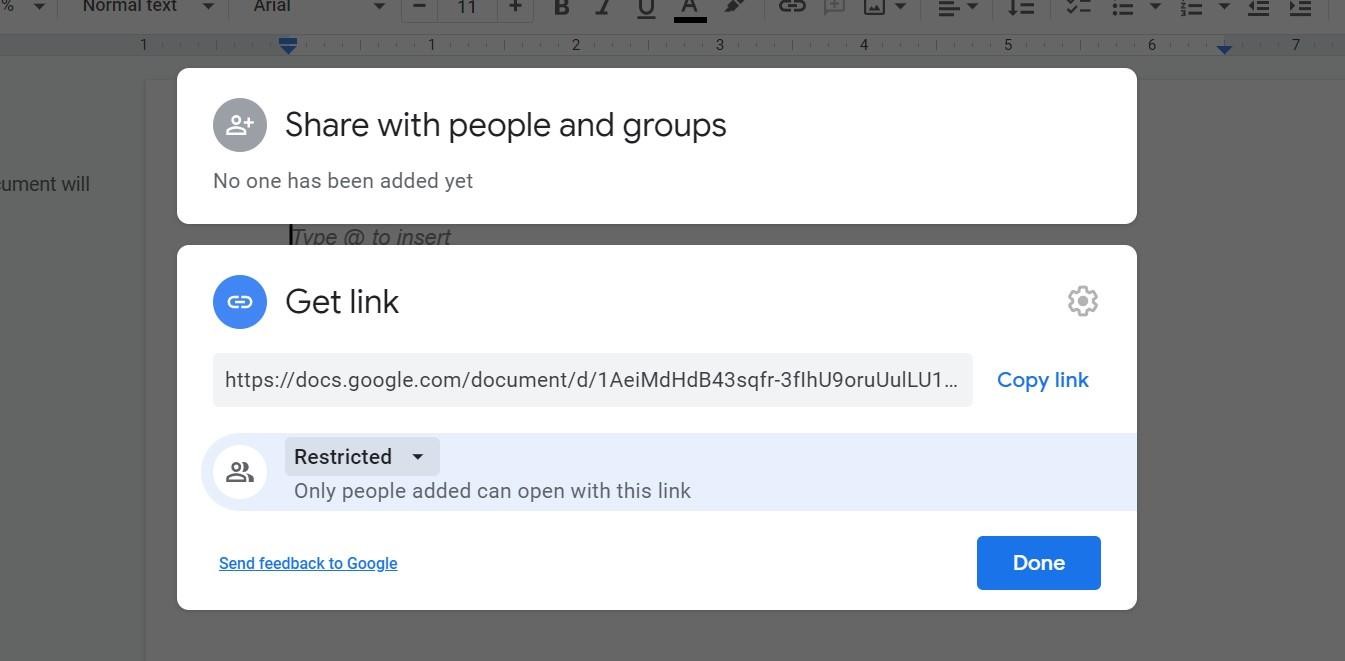
By adding the E-mail addresses of Users, as shown below, we can Share the Google Docs File online

through Google Drive (Cloud) with those Users.



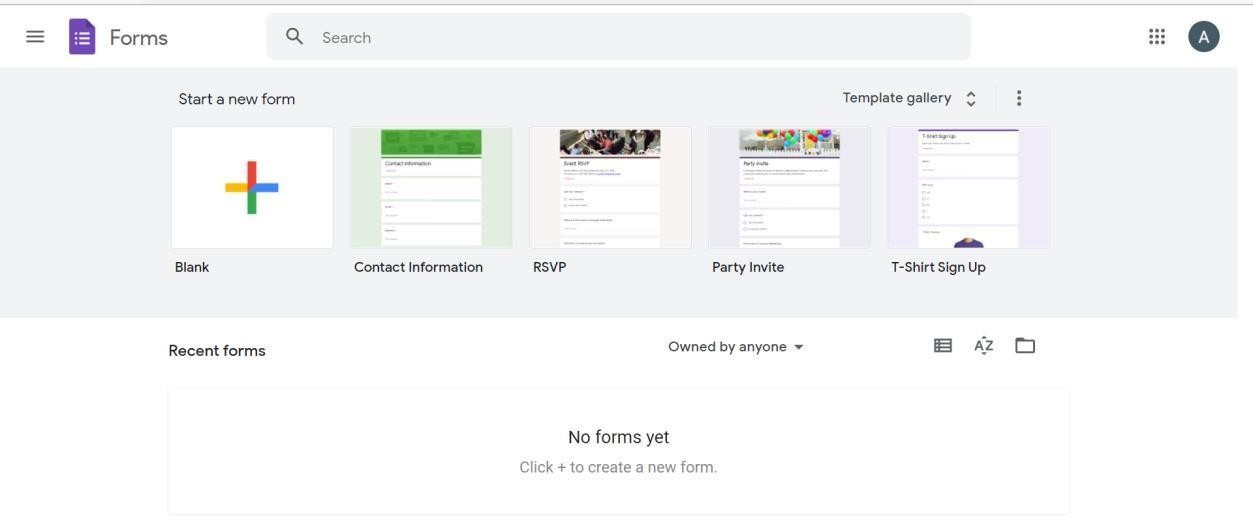


Also, Google Docs can be shared through Link by giving Access Rights as per Author as to whether User with Link should only View/Comment or also Edit.



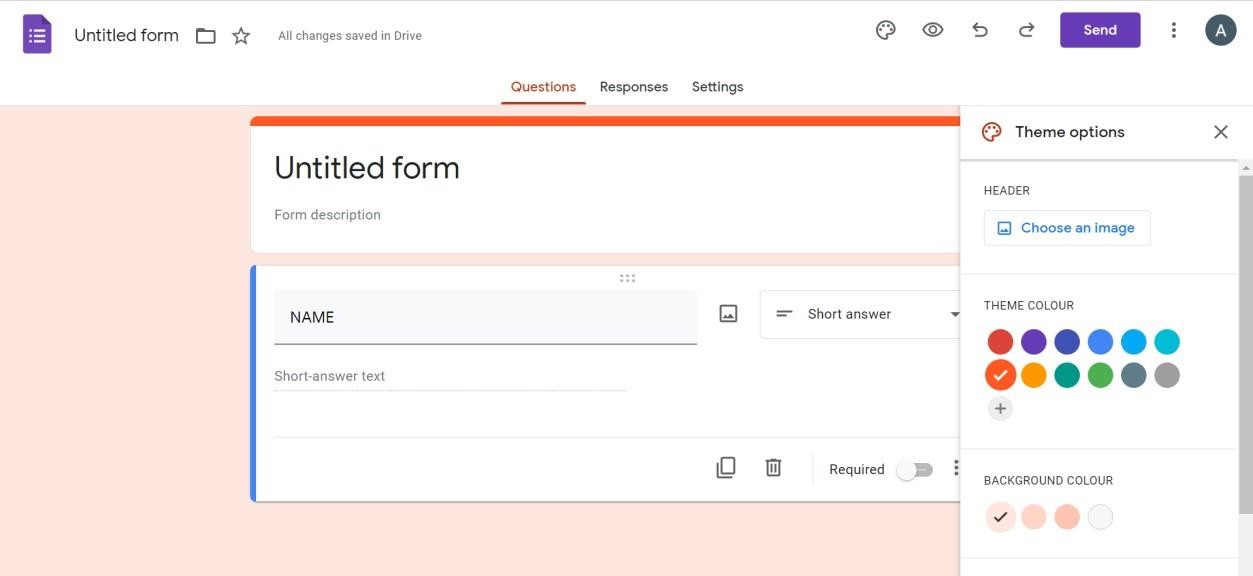
#### Creating Google Forms

Step 1: For Creating Google Forms, select Forms option then from the ‘Start a new form’ area click on ‘+’ sign named ‘Blank’ to Create a New Google Form. Also, ready Templates areavailable for various types of forms here which can be used on the go.



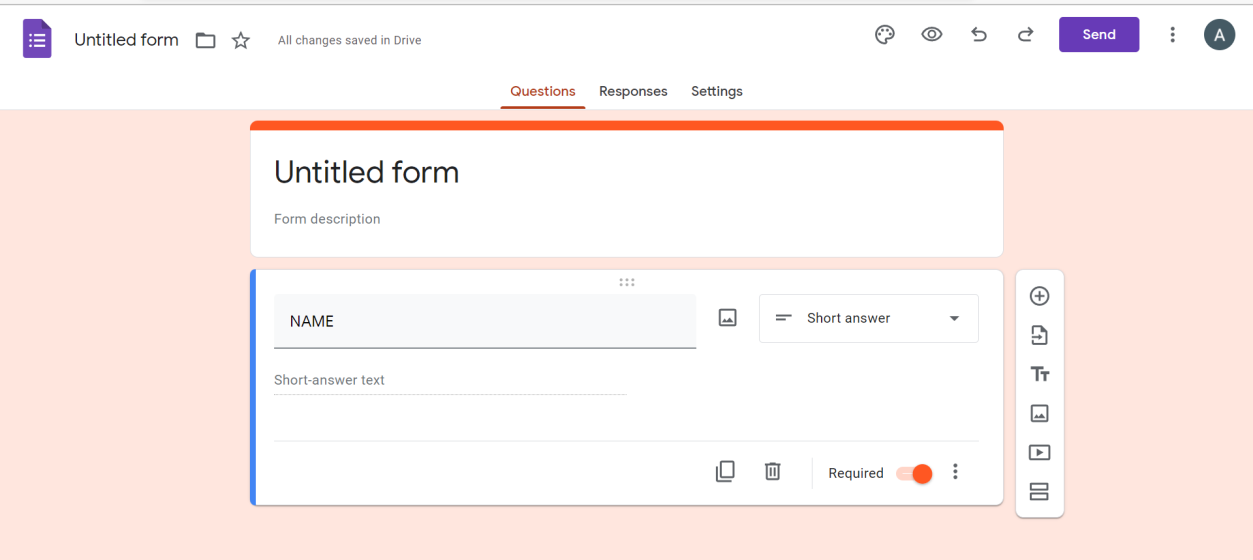
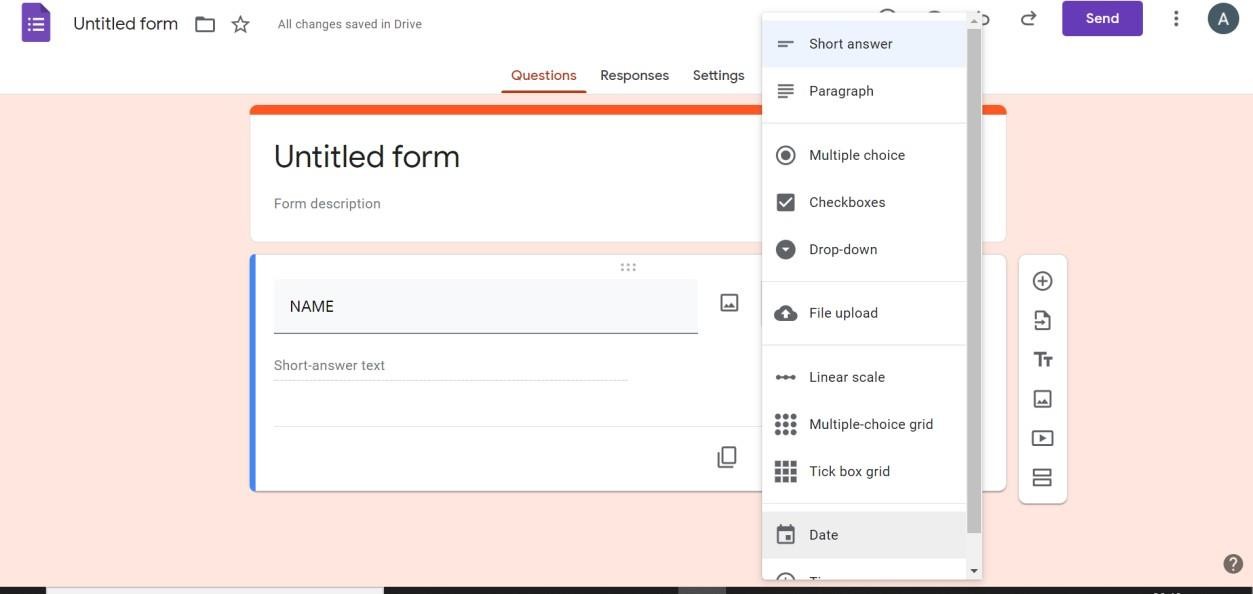
Step 2: Google Form will be opened. We can Insert and Edit the Form as desired using various components provided by Google Forms and they are AutoSaved in the Cloud (Google Drive)of User Account. Below Theme Options are shown using which Theme Colour, Background Colour,Header

Image, Font Style can be set for particular Form.

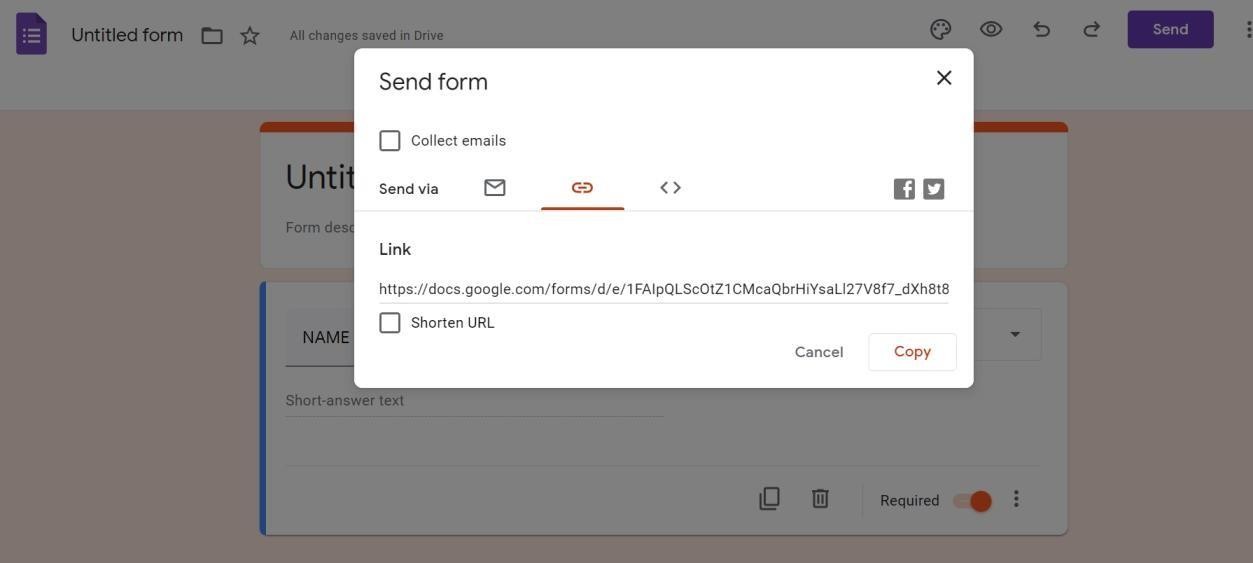


Step 3: Google Form provides with Components for Forms as shown below using which user-friendly forms can be made very easily. Also, we can add different Sections.

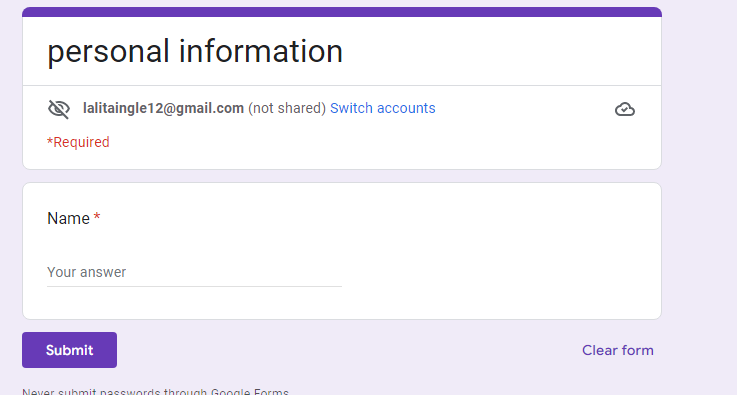
Video/Image can also be added to Google Forms. Certain Field can also made mandatory tobe filled by marking it as Required with option shown below.



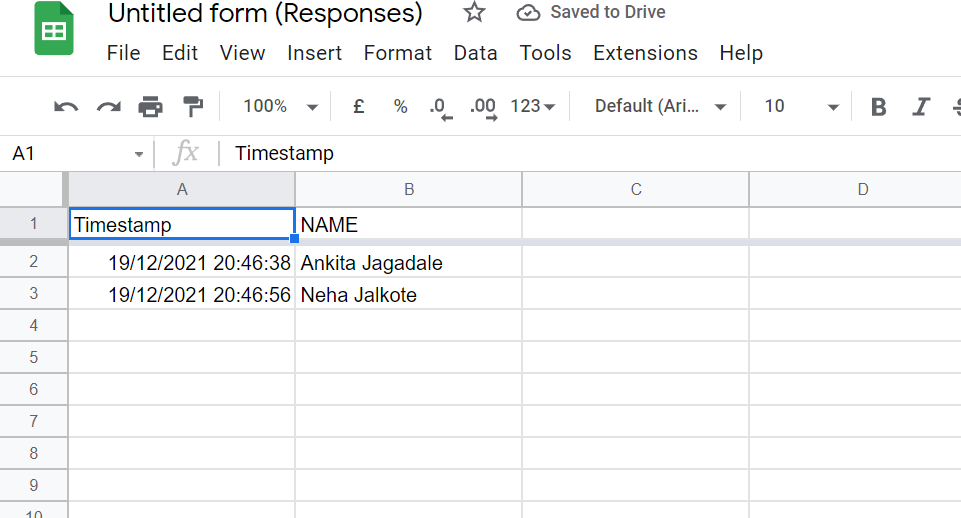
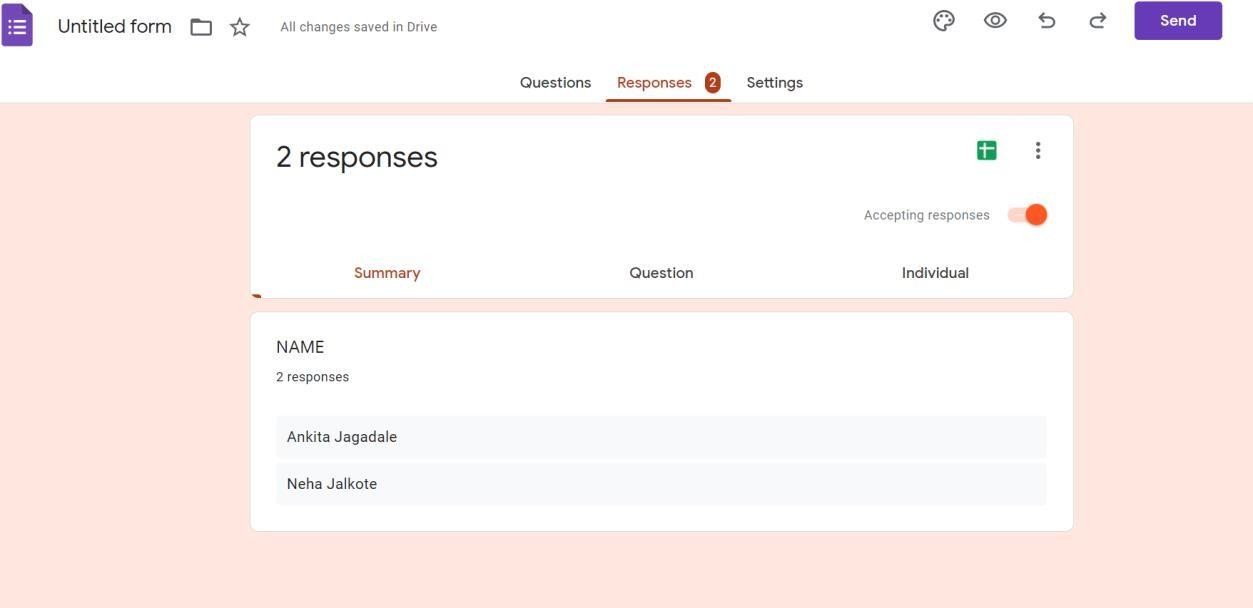
Step 4: When clicked on Share button, Google Form can be shared using various ways by mailing through E-mail addresses of Users or sharing through Link to the Form.



Step 5: Form User will attempt the from and Submit the Form Responses.

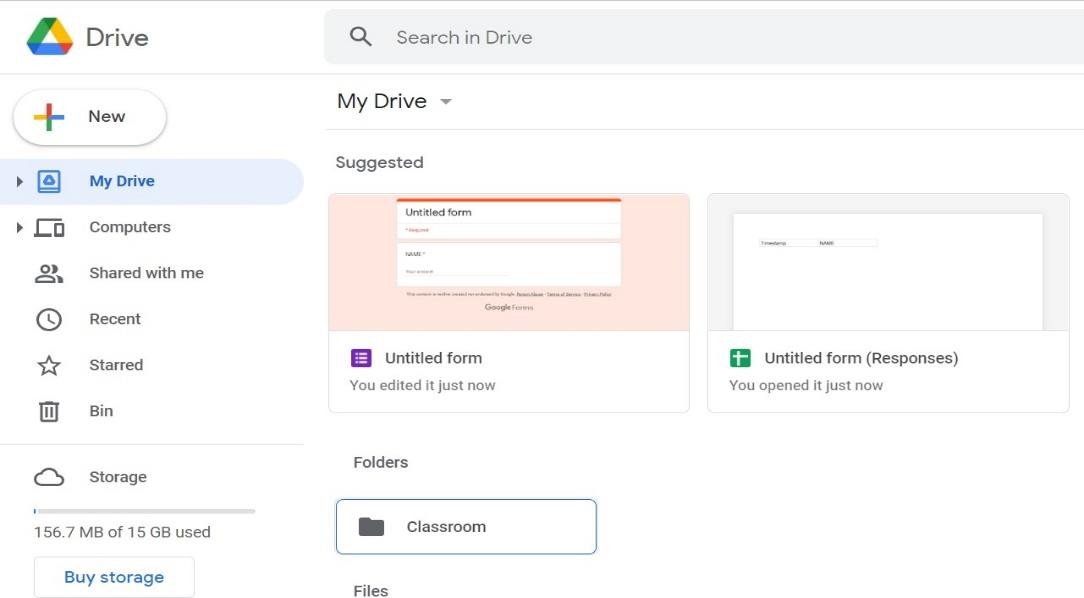


Step 6: Form creator can check the Form Replies received in the Responses Section as shownbelow. The responses can also be exported in .csv format offline.



Step 7: All Google Docs Files can be accessed when required from Google Drive (Cloud) in User

Account.



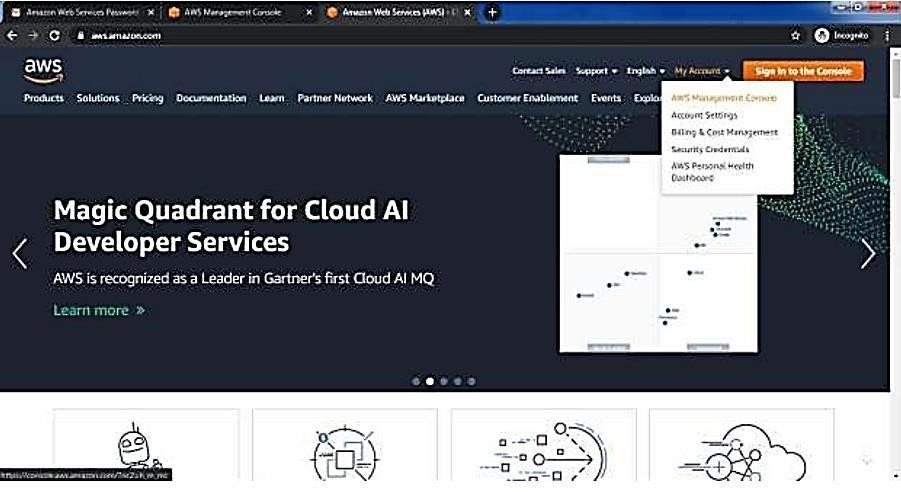
# PRACTICAL 7

**AIM: : Implementation of Identity Management using Cloud Computing concept Description:**

Identity management in cloud computing is the subsequent step of identity and access management (IAM) solutions. However, it is a lot more than merely a straightforward web app single sign-on (SSO) solution. This next generation of IAM solution is a holistic move of the identity provider right to the cloud.

What is AWS: AWS (Amazon Web Services) is a comprehensive, evolving [cloud](https://www.techtarget.com/searchcloudcomputing/definition/cloud-computing) [computing](https://www.techtarget.com/searchcloudcomputing/definition/cloud-computing) platform provided by Amazon that includes a mixture of infrastructure-as-a- service ([IaaS](https://www.techtarget.com/searchcloudcomputing/definition/Infrastructure-as-a-Service-IaaS)), platform-as-a-service ([PaaS](https://www.techtarget.com/searchcloudcomputing/definition/Platform-as-a-Service-PaaS)) and packaged-software-as-a-service ([SaaS](https://www.techtarget.com/searchcloudcomputing/definition/Software-as-a-Service)) offerings. AWS services can offer an organization tools such as compute power, database storage and content delivery services.

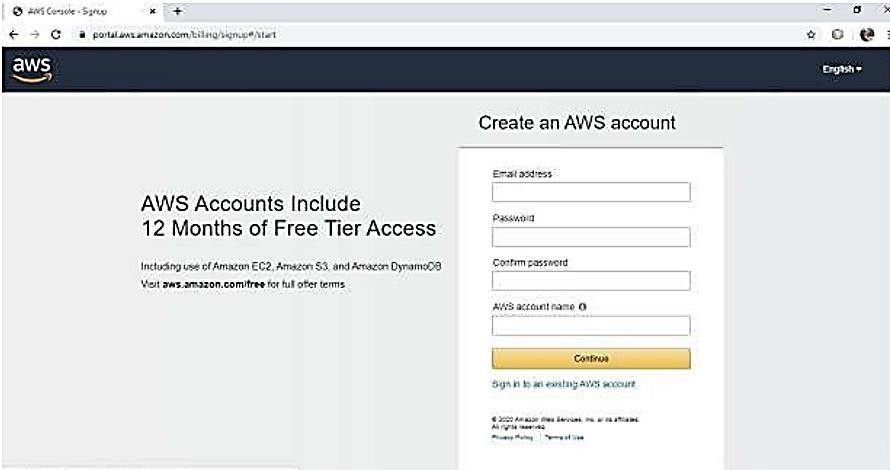
#### Implementation of Identity Management. Procedure :

Step1: Open the following link <https://aws.amazon.com/> Step2: Go to my Account-> AWS management console

Step3: Click on Create new user AWS account



Step4: Fill all the details and click on Continue

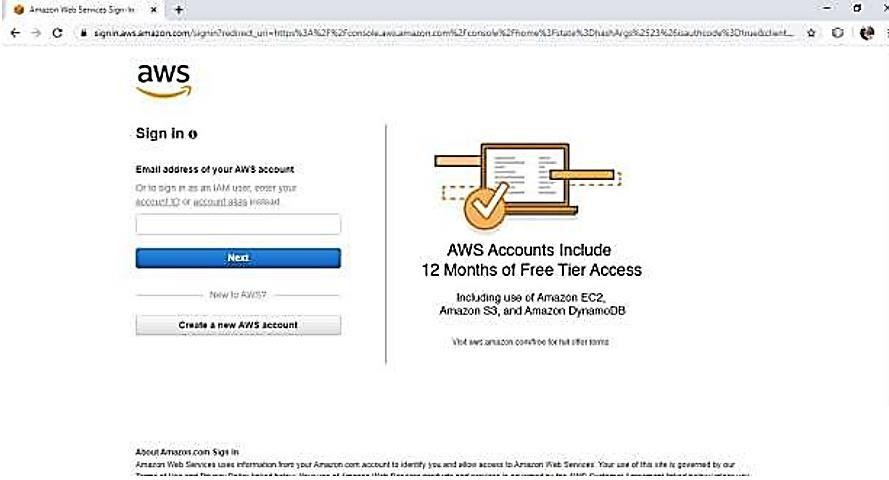


Step5: Fill your contact number and home address and click on create account and continue Step6: Now most curtail step AWS will ask for credit card and debit card details. You have

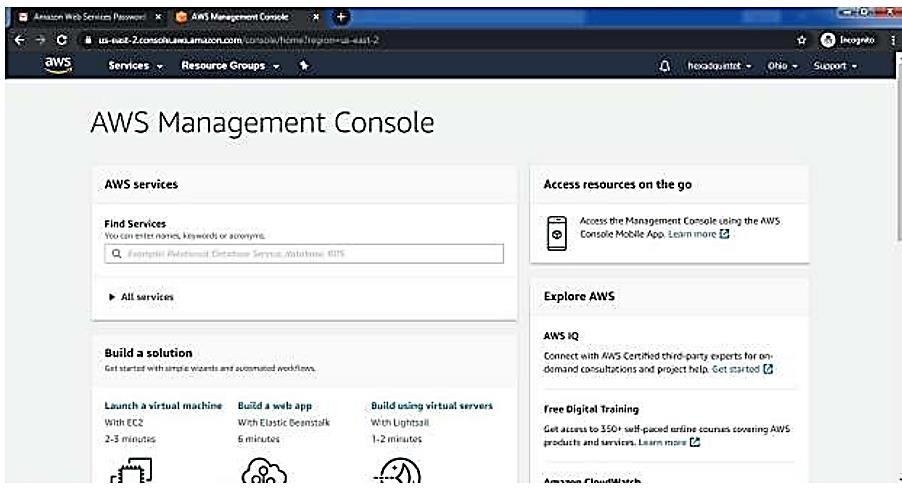
to close the browser.

Step7: now again open the link <https://aws.amazon.com/>

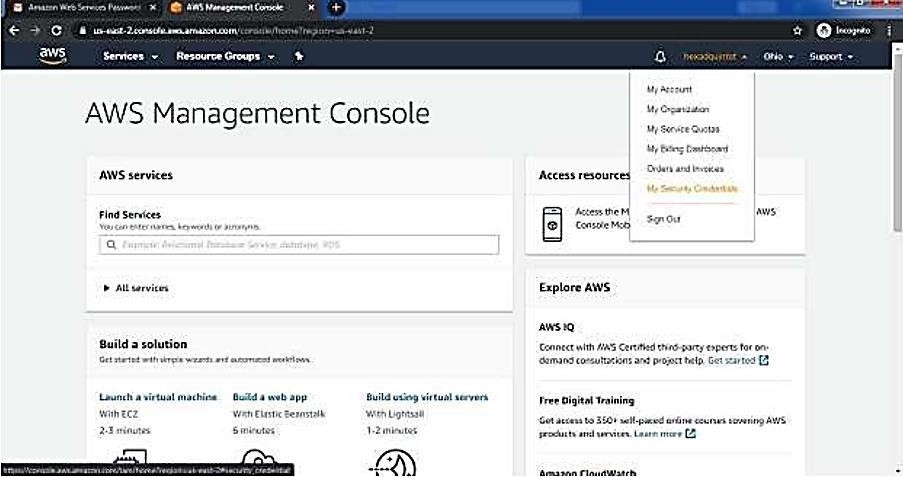
Step8: Go to my Account->AWS Management console



Enter your ID and click on next, After that enter password and click on sign in Step 9: you will get the following screen



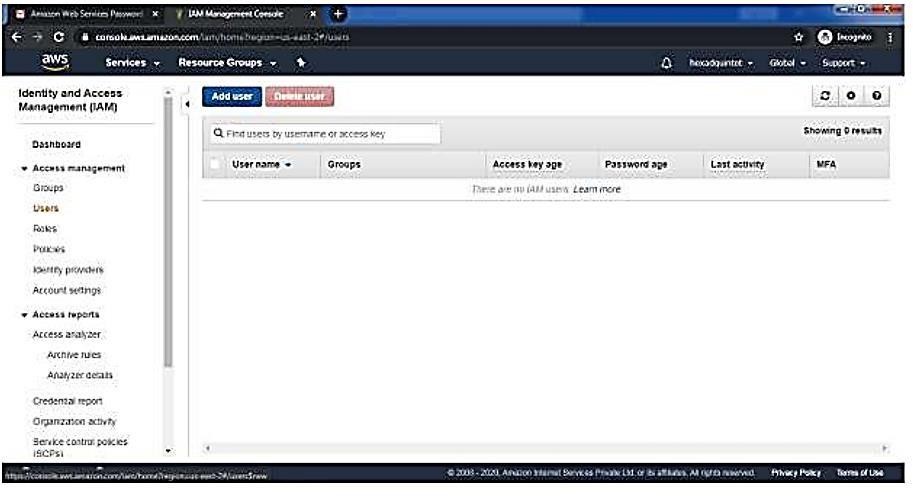
Step 10: Go to My Security credential



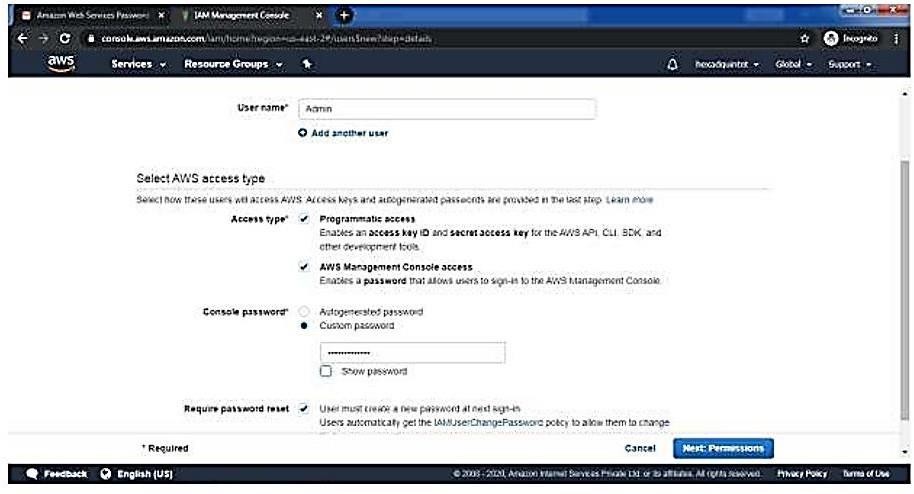
Step 11: now click on user



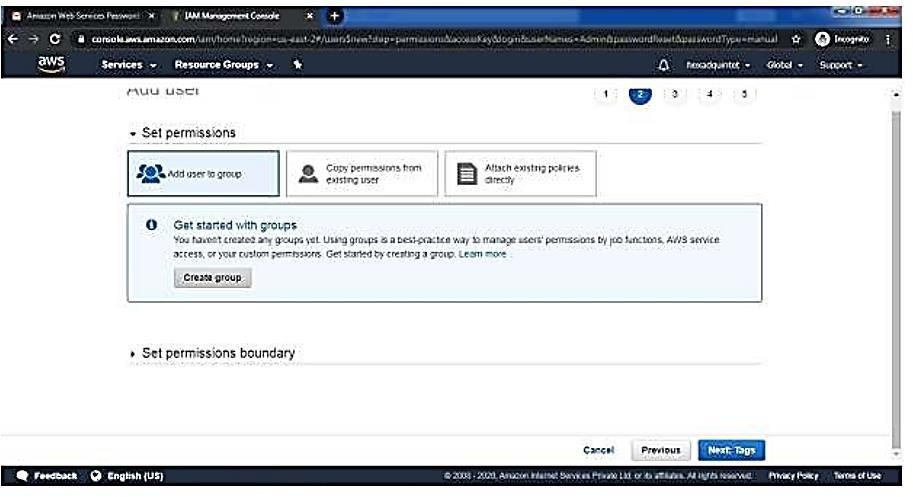
Step12: Click on add user

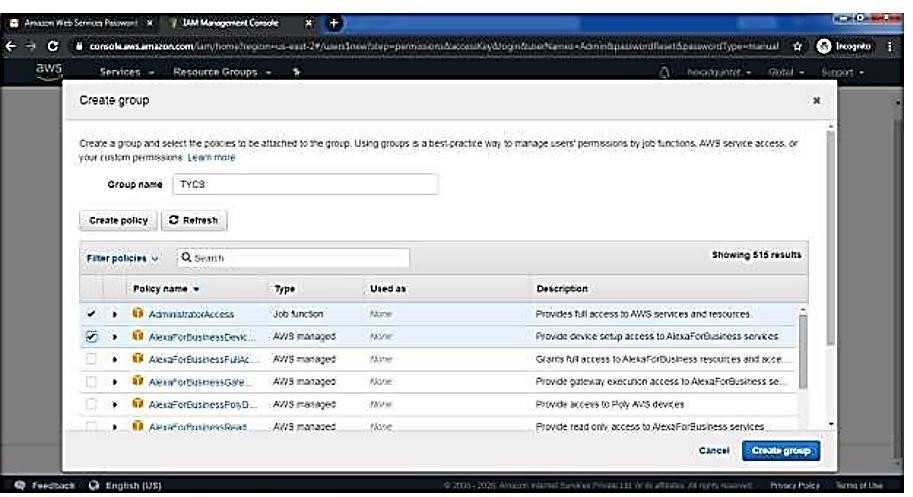


Step 13: Provide the user name and check the check box in front of programmatic access and AWS Management console Access and enter the password for new user Click on custompassword and click on next permission

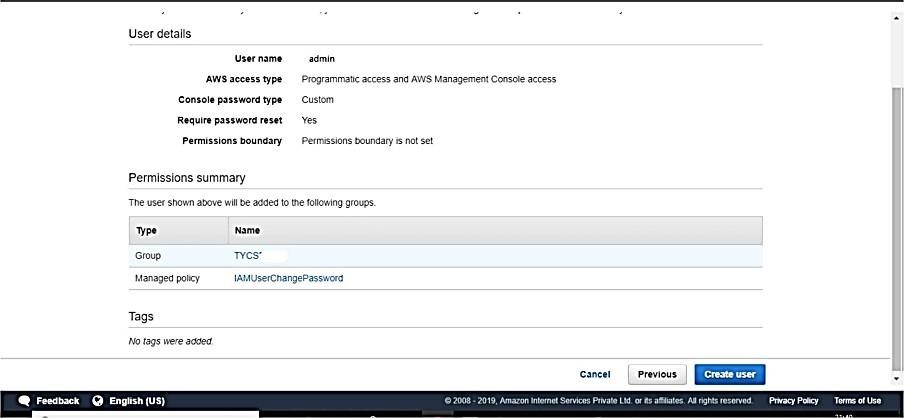


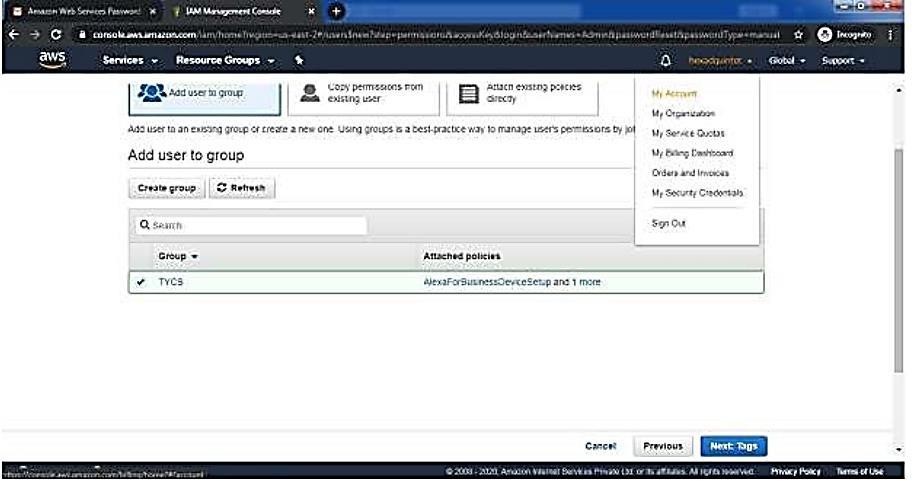
Step 14: click on create Group



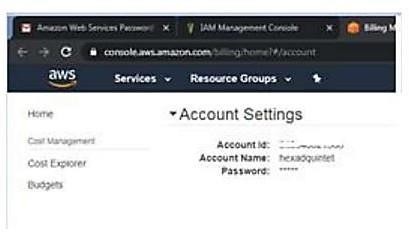
Step15: fill the information and click on Create Group

Step16: Click on next tag leave blank, again click on next review leave as it is and click oncreate user



Step 17: Click on close

And COPY Account ID

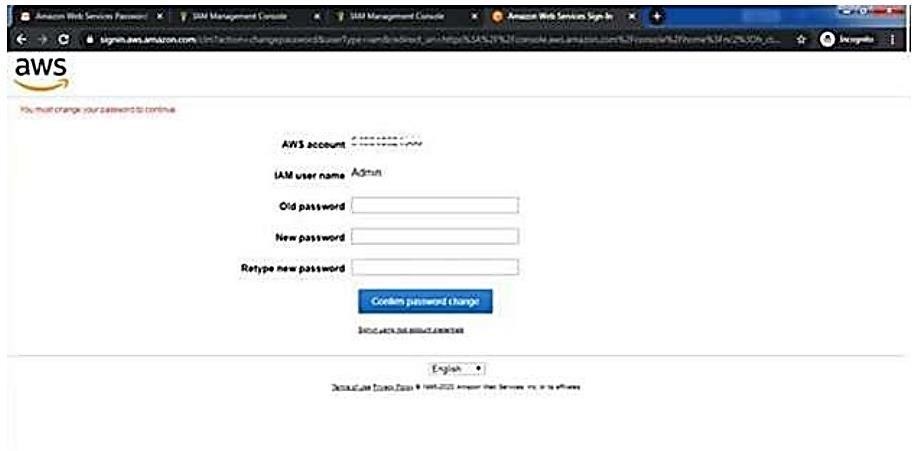


Now logout the admin account and try to login as user (newly created).

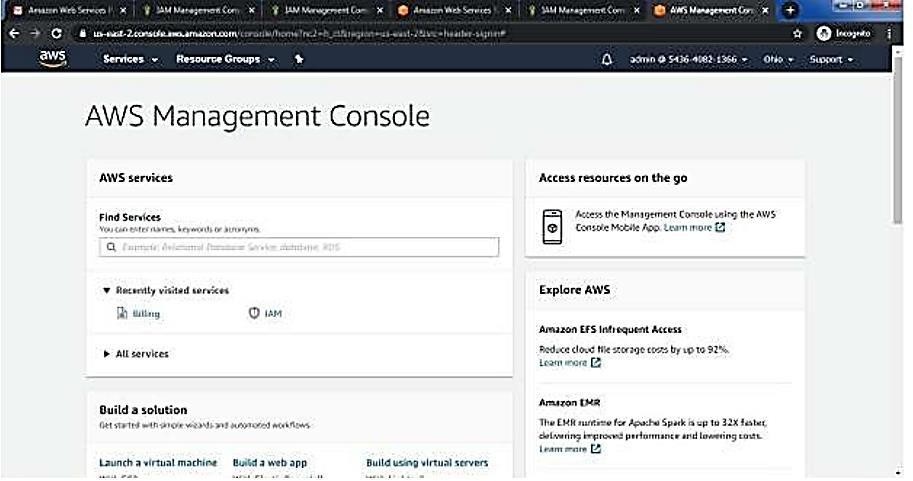
Step18: Again, Go to my Account->AWS Management console



Click on next Provide the Account ID username and password and click on sign in It will askyou to change the password which is been set by administrator



Yow will redirect to home screen



Conclusion: Hence we have studied the concept and implementation of identity managementusing amazon AWS.

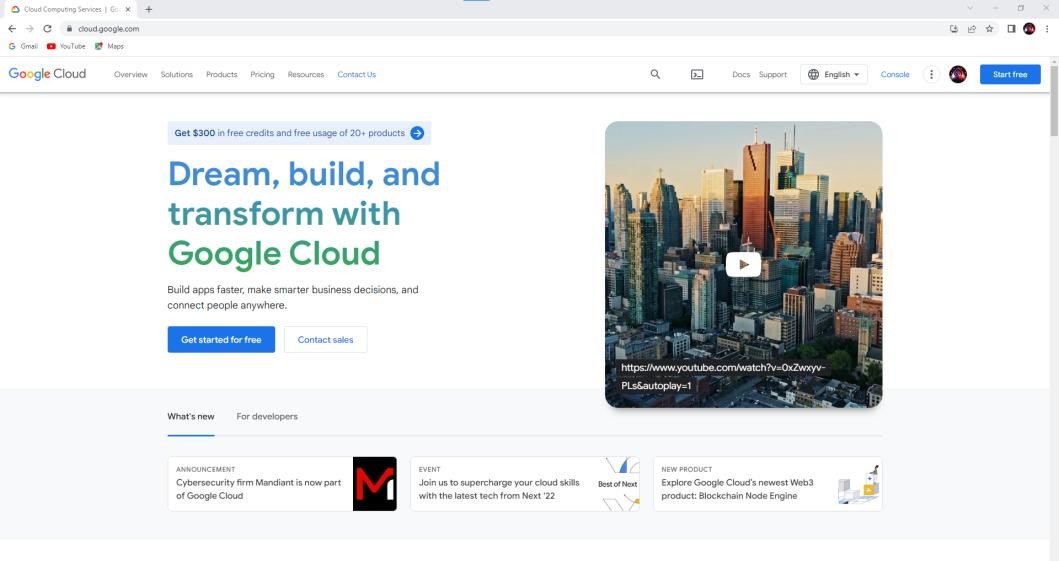
**Practical 8**

**Aim: App development using cloud computing Description:**

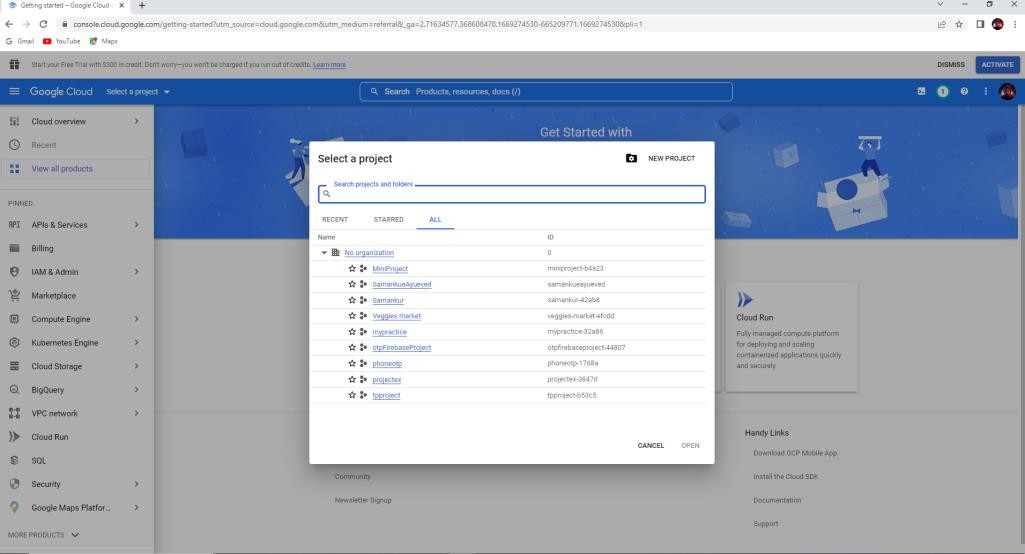
Cloud application development is the process through which a Cloud-based app is built. It involves different stages of software development, each of which prepares your app to go live and hit the market. The best Cloud app development teams use [DevOps practices](http://www.koombea.com/services/devops) and tools like [Kubernetes](http://www.koombea.com/blog/devops-tools-kubernetes/).

However, an [experienced app development company](http://www.koombea.com/) should ideally be technology agnostic, which means being able to build your Cloud app using any technology you prefer. Most apps built using the Cloud are highly dependent on the Cloud to operate.

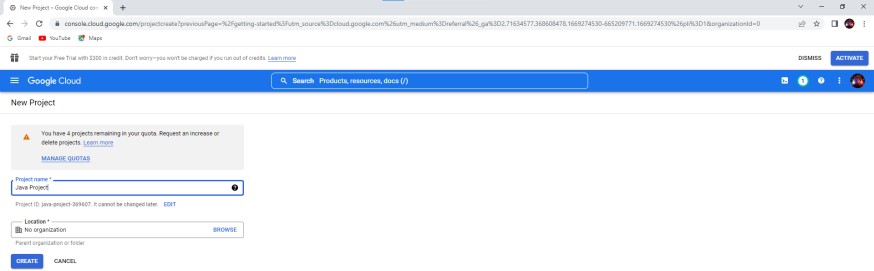
1. open <https://cloud.google.com/>



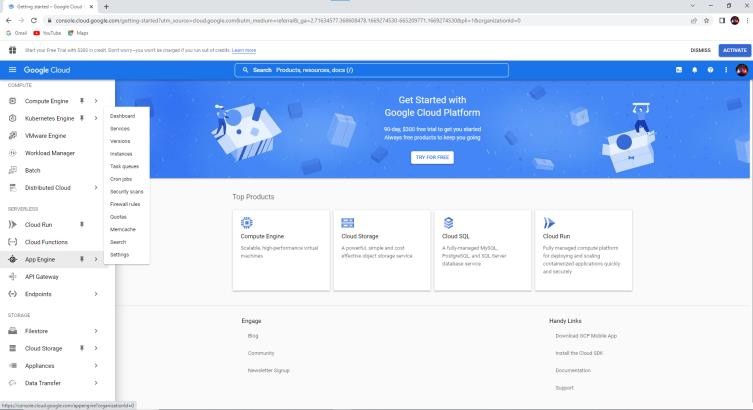
1. go to console – select for new project



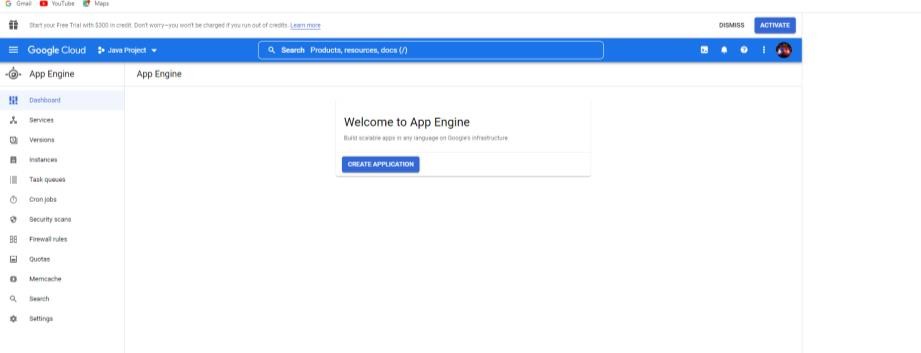
1. select new project-give name java project



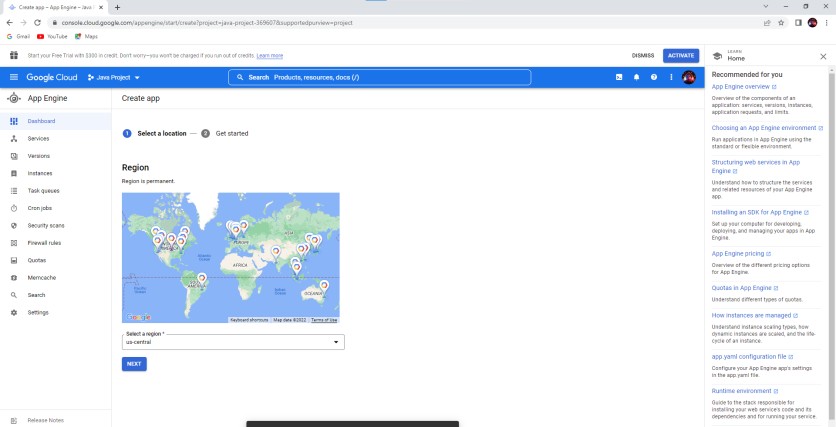
1. go to severLess-App Engine



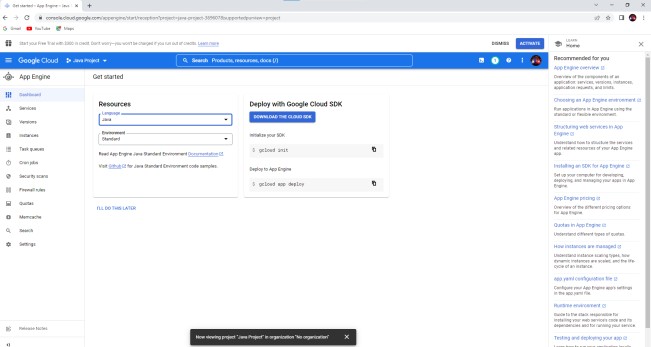
1. go app engine – select app – select create application .



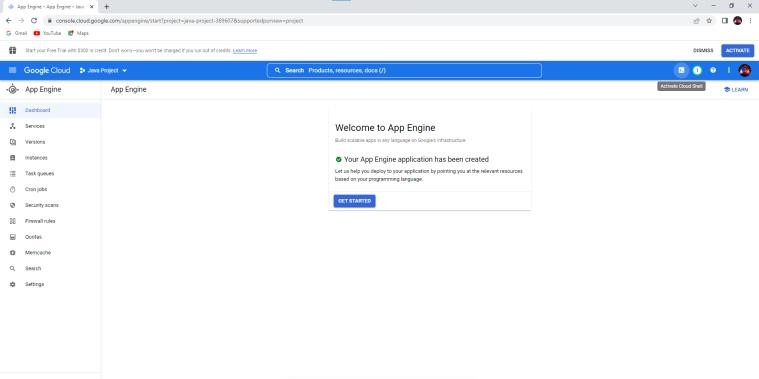
1. By default region select – click Next

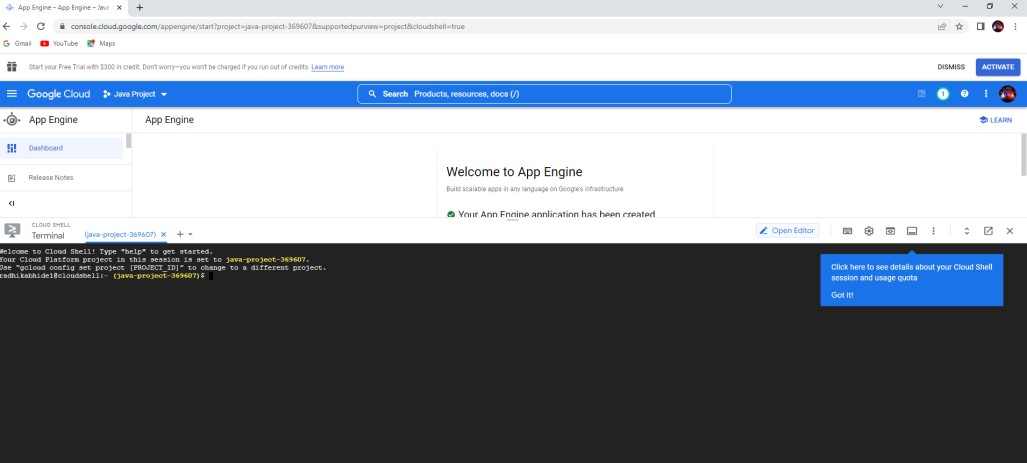


1. Select Language (java) - click on I’ll do later

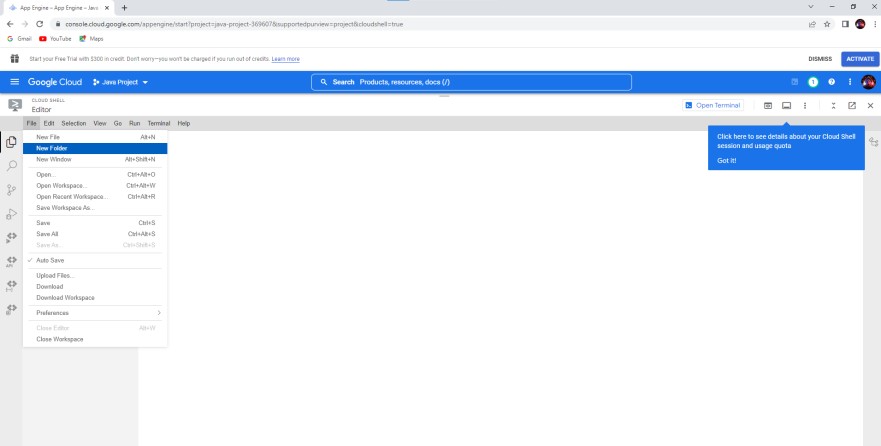


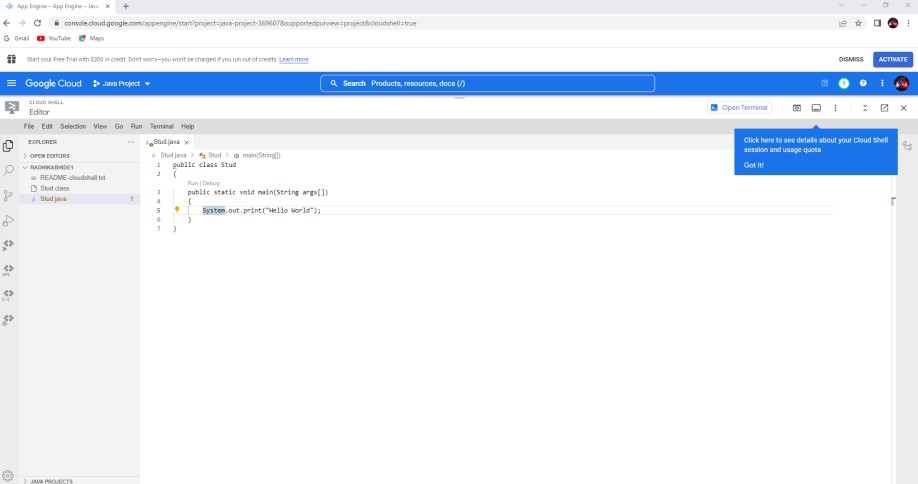
1. Now click on Get Started To start the App Engine



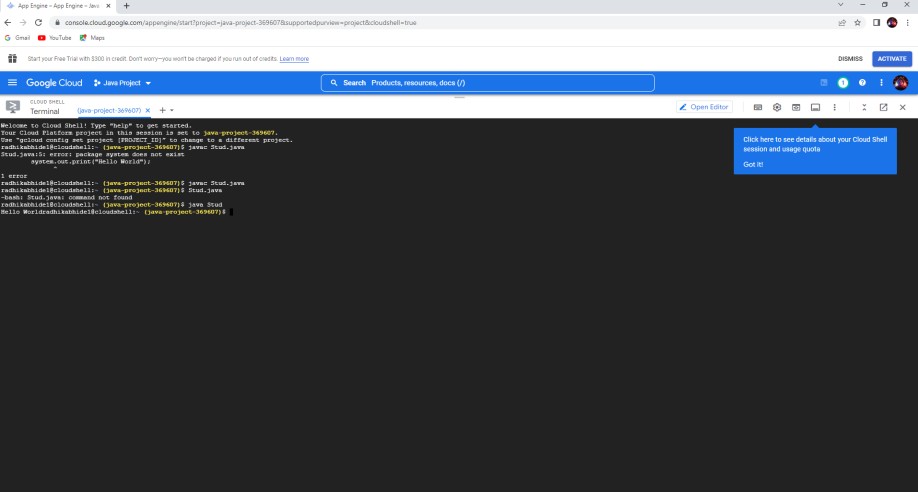


1. Inside App Engine Editor open new folder create new file and type the code:



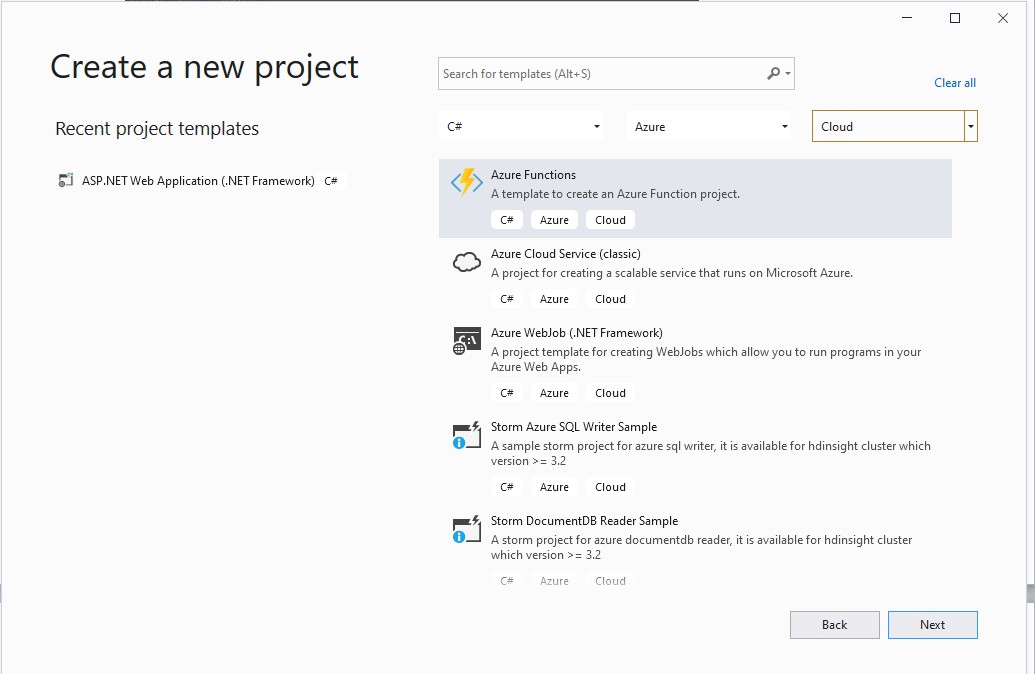


1. After typing the code run the file:

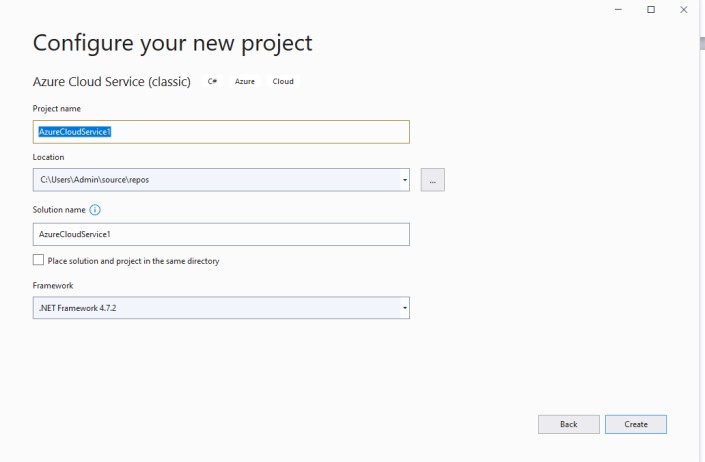


To develop application for windows Azure using windows Azure platform tranining kit and visual studio.

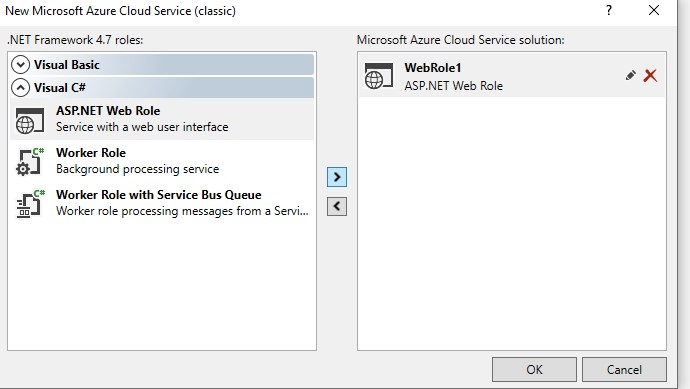
1. open visual studio – language



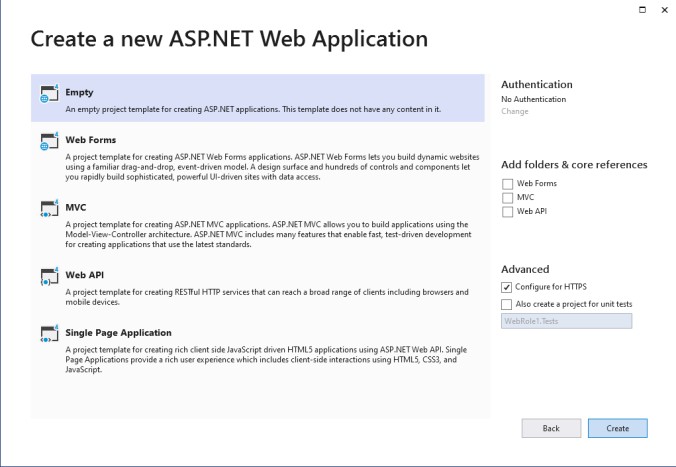
1. Give the proper name to project:



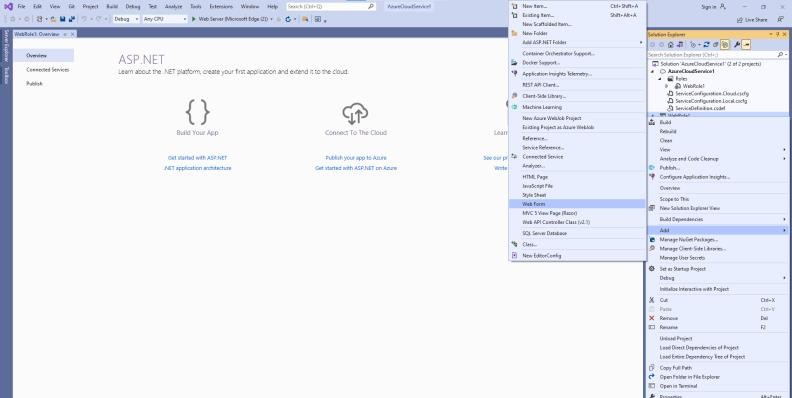
1. Select Asp.Net Web Role and click on Ok:



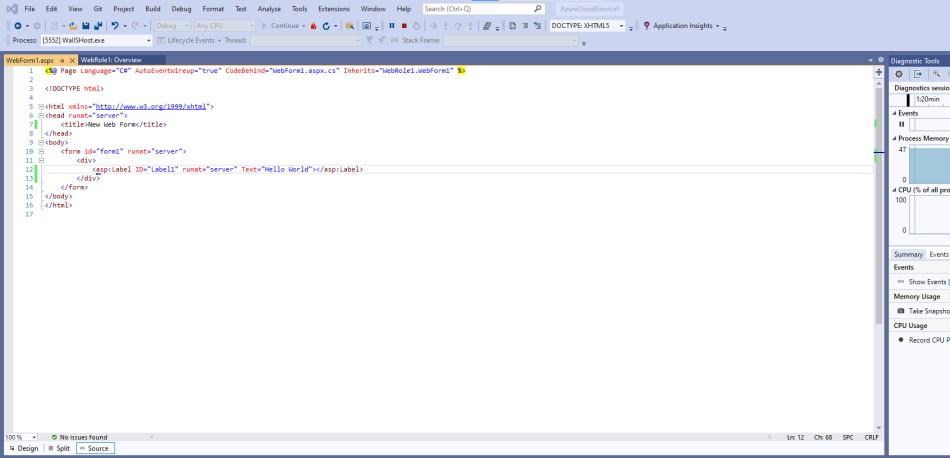
1. Select Empty application:



1. Now click on file and then click on Web Form then click on add:



1. Type the Html form code and run the app:



1. Output

