**MUMBAI EDUCATIONAL**

**TRUST**

Name : Vedika Bodekar Roll no : 1302

**Distributed System & Cloud Computing Lab**

|  |  |  |
| --- | --- | --- |
| ***Sr No*** | ***Title of Practical*** | ***Date*** |
| 1. | To develop a program for multi-client chat server using Server Socket | 25.08.2023 |
| 2. | To implement Calculator using RPC (Datagram Socket) | 11.09.2023 |
| 3. | To implement Date Time Server using RPC | 11.09.2023 |
| 4. | To implement Hello Server using RMI | 19.09.2023 |
| 5. | To implement Calculator using RMI | 19.09..2023 |
| 6 | To implement Date Time using RMI | 19.09.2023 |
| 7. | To implement GUI Calculator using RMI | 25.09.2023 |
| 8. | Using MySQL, create a Library database. Create table Book (Book\_id, Book\_name, Book\_author) and retrieve the Book information from Library database using Remote Object Communication | 16.10.2023 |
| 9. | Using MySQL create Elecrtic\_Bill database. Create table Bill(consumer\_no,billduedate, Billamt) and retrieve the bill information from Bill table using Remote Object Communication | 16.10.2023 |
| 10. | Implementation of Shared Memory | 30.10.2023 |
| 11. | Implementation of mutual exclusion using token Ring | 2.11.2023 |
| 12. | To develop Application for windows using Windows Azure Platform Training Kit and Visual Studio. | 6.11.2023 |
| 13. | To develop Application using Google App Engine | 6.11.2023 |
| 14 | To implement Identity Management in GCP | 6.11.2023 |
| 15 | Implement Storage as Service on Google Cloud | 6.11.2023 |

**MUMBAI EDUCATIONAL**

**TRUST**

# Practical 1:

**Aim: To develop a program for multi-client chat server using Server Socket Server.java**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package server;

import java.io.DataInputStream; import java.io.DataOutputStream; import java.io.IOException; import java.net.ServerSocket; import java.net.Socket;

import java.util.Scanner;

import java.util.StringTokenizer; import java.util.Vector;

/\*\*

* @author mcamock

\*/

public class Server {

//vector to store active clients

static Vector<ClientHandler> ar = new Vector<>();

Server() { try {

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

while (true) {

**MUMBAI EDUCATIONAL**

**TRUST**

s = ss.accept();

System.out.println("New Client request received: " + s); DataInputStream dis = new DataInputStream(s.getInputStream()); DataOutputStream dos = new DataOutputStream(s.getOutputStream()); String username = dis.readUTF();

System.out.println("Creating a new handler for this client ");

ClientHandler mtch = new ClientHandler(s, username, dis, dos); Thread t = new Thread(mtch);

System.out.println("Adding this client to active client"); System.out.println(username+ " has logged in"); ar.add(mtch);

t.start();

}

} catch (Exception ex) {

}

}

class ClientHandler implements Runnable {

Scanner scn = new Scanner(System.in); private String uname;

final DataInputStream dis; final DataOutputStream dos; Socket s;

boolean isloggedin;

public ClientHandler(Socket s, String uname, DataInputStream dis, DataOutputStream dos) {

this.dis = dis; this.dos = dos; this.uname = uname; this.s = s; this.isloggedin = true;

}

**MUMBAI EDUCATIONAL**

**TRUST**

@Override public void run() {

String received; try {

for (ClientHandler mc : Server.ar) { for (ClientHandler cc : Server.ar) {

mc.dos.writeUTF(cc.uname + " has logged in");

}

}

} catch (Exception ex) {

}

while (true) { try {

received = dis.readUTF(); System.out.println(received); if (received.equals("logout")) {

System.out.println(this.uname + " has logged out"); this.isloggedin = false;

this.dis.close(); this.dos.close(); this.s.close(); break;

}

StringTokenizer st = new StringTokenizer(received, "#"); String MsgToSend = st.nextToken();

String recipient = st.nextToken(); for (ClientHandler mc : Server.ar) {

if (mc.uname.equals(recipient) && mc.isloggedin == true) { mc.dos.writeUTF(this.uname + "SAYS" + MsgToSend); break;

}

}

} catch (Exception e) {

**MUMBAI EDUCATIONAL**

**TRUST**

e.printStackTrace();

}

}

}

}

public static void main(String[] args) { Server cs = new Server();

}

}

**Client.java**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package client;

import java.io.DataInputStream; import java.io.DataOutputStream; import java.io.IOException; import java.net.InetAddress; import java.net.Socket;

import java.net.UnknownHostException; import java.util.Scanner;

/\*\*

* @author mcamock

\*/

public class Client {

final static int ServerPort = 1234;

**MUMBAI EDUCATIONAL**

**TRUST**

public static void main(String[] args) throws UnknownHostException, IOException {

Scanner sc = new Scanner(System.in);

InetAddress ip = InetAddress.getByName("localhost"); Socket s = new Socket(ip, ServerPort);

DataInputStream dis = new DataInputStream(s.getInputStream()); DataOutputStream dos = new DataOutputStream(s.getOutputStream()); Thread sendMessage;

sendMessage = new Thread(new Runnable() { @Override

public void run() {

System.out.println("Enter username to login"); while (true) {

String msg = sc.nextLine(); try {

dos.writeUTF(msg);

if (msg.equals("logout")) { s.close(); System.exit(0);

}

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

}

}

}

});

Thread readMessage = new Thread(new Runnable() { @Override

public void run() { while (true) {

try {

String msg = dis.readUTF(); System.out.println(msg);

} catch (IOException ex) {

**MUMBAI EDUCATIONAL**

**TRUST**

ex.printStackTrace();

}

}

}

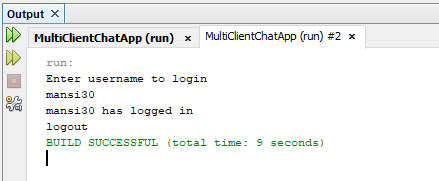
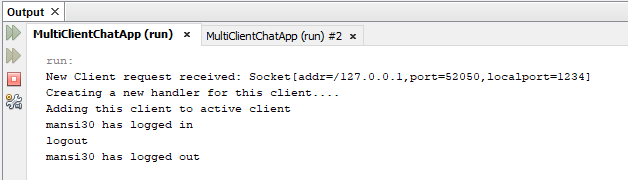
});

readMessage.start(); sendMessage.start();

}

}

**Output:**



**MUMBAI EDUCATIONAL**

**TRUST**

# Practical 2

**Aim: To implement Calculator using RPC (Datagram Socket) Server.java**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package rpcServer;

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

import java.util.StringTokenizer;

/\*\*

* @author mcamock

\*/

public class RPCServer { DatagramSocket dsr; DatagramPacket dpr;

String str,methodName,result; int val1,val2;

RPCServer() throws SocketException, IOException{ try{

dsr = new DatagramSocket(1200); while(true){

byte[] br = new byte[4096];

dpr = new DatagramPacket(br,br.length); dsr.receive(dpr);

str = new String(dpr.getData(),0,dpr.getLength()); if(str.equalsIgnoreCase("exit")){

break;

}

else{

**MUMBAI EDUCATIONAL**

**TRUST**

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

{

methodName = st.nextToken();

val1 = Integer.parseInt(st.nextToken()); val2 = Integer.parseInt(st.nextToken());

}

System.out.println("\n Client Selected\""+str+"\"Method: "); System.out.println("\nFirst Value: "+val1); System.out.println("\nSecond Value: "+val2); 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 bs[] = result.getBytes();

DatagramSocket dss = new DatagramSocket(); DatagramPacket dps = new

DatagramPacket(bs,bs.length,InetAddress.getLocalHost(),1300); System.out.println("Result: "+result+"\n"); dss.send(dps);

}

}

}

catch(Exception ex){ ex.printStackTrace();

}

}

**MUMBAI EDUCATIONAL**

**TRUST**

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 float div(float val1, float val2) { return val1/val2;

}

public static void main(String[] args) throws IOException{ new RPCServer();

}

}

**Client.java**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package rpcClient;

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

import java.util.Scanner; import rpcServer.RPCServer;

/\*\*

* @author mcamock

\*/

**MUMBAI EDUCATIONAL**

**TRUST**

public class RPCClient {

RPCClient(){ try{

System.out.println("\nRPC Client"); System.out.println(" \n");

DatagramSocket dsr = new DatagramSocket(1300); while(true){

System.out.println("Enter Method Name "+ " with Parameter like add 3 4/sub 4

3"+"\n");

Scanner br = new Scanner(System.in); String str = br.nextLine();

byte b[] = str.getBytes();

DatagramSocket dss = new DatagramSocket(); DatagramPacket dp = new

DatagramPacket(b,b.length,InetAddress.getLocalHost(),1200); dss.send(dp);

if(str.equals("exit")) break;

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

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

}

}

catch(Exception ex){ ex.printStackTrace();

}

}

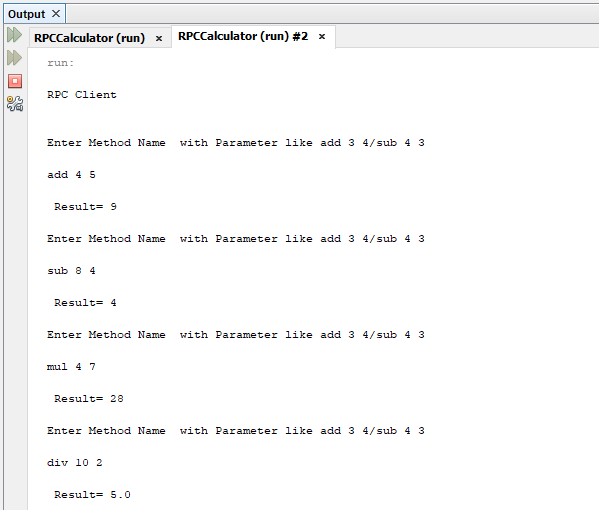
public static void main(String[] args) throws IOException{ new RPCClient();

}

}

**Output:**

**MUMBAI EDUCATIONAL**

**TRUST**

**MUMBAI EDUCATIONAL**

**TRUST**

# Practical 3 Aim: To implement Date Time Server using RPC

**Server.java**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package dateServer;

import java.io.DataInputStream; import java.io.DataOutputStream; import java.net.ServerSocket; import java.net.Socket;

import java.text.SimpleDateFormat; import java.util.Date;

/\*\*

* @author mcamock

\*/

public class DateServer { DateServer() throws Exception{

ServerSocket ss = new ServerSocket(4000); System.out.println("Date Time Server Started"); Socket s = ss.accept();

DataInputStream dis = new DataInputStream(s.getInputStream()); DataOutputStream dos = new DataOutputStream(s.getOutputStream()); while(true){

String str = dis.readUTF(); if(str.equals("date"))

dos.writeUTF("Date: "+mydate()); else if(str.equals("time"))

**MUMBAI EDUCATIONAL**

**TRUST**

dos.writeUTF("Time: "+mytime()); else break;

}

s.close();

ss.close();

}

public String mydate(){

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

}

public String mytime(){

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

}

public static void main(String[] args) throws Exception{ DateServer d = new DateServer();

}

}

**Client.java**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package dateClient;

import java.io.DataInputStream; import java.io.DataOutputStream; import java.net.Socket;

import java.util.Scanner;

/\*\*

* @author mcamock

\*/

public class DateClient {

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

**MUMBAI EDUCATIONAL**

**TRUST**

{

Socket s = new Socket("localhost",4000);

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

while(true){

System.out.println("Enter date/time/exit"); String str = sc.nextLine(); if(str.equals("exit")){

dos.writeUTF(str); break;

}

dos.writeUTF(str); System.out.println(dis.readUTF());

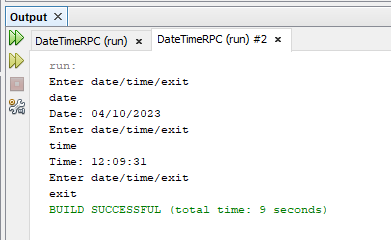
}

s.close();

}

}

**Output:**



**MUMBAI EDUCATIONAL**

**TRUST**

# Practical 4 Aim: To implement Hello Server using RMI

**HelloInterface.java**

package hello;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface HelloInterface extends Remote

{

public String say() throws RemoteException;

}

# HelloClient.java

package hello;

import java.rmi.Naming; public class HelloClient

{

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

{

HelloInterface hello = (HelloInterface) Naming.lookup("//localhost/Hello"); System.out.println(hello.say());

}

}

# HelloServer.java

package hello;

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

**MUMBAI EDUCATIONAL**

**TRUST**

import java.rmi.server.UnicastRemoteObject; import java.util.Scanner;

public class HelloServer extends UnicastRemoteObject implements HelloInterface

{

private String message;

public HelloServer(String msg) throws RemoteException

{

message = msg;

}

@Override public String say() throws RemoteException

{

return(message);

}

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

{

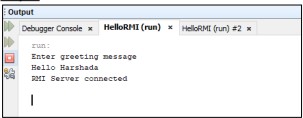
Registry r = LocateRegistry.createRegistry(1099); Scanner sc = new Scanner(System.in); System.out.println("Enter greeting message"); String s = sc.nextLine();

r.rebind("Hello", new HelloServer(s)); System.out.println("RMI Server connected");

}

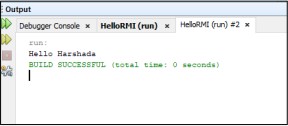
}

**Output:**



**MUMBAI EDUCATIONAL**

**TRUST**



**MUMBAI EDUCATIONAL**

**TRUST**

# Practical 5 Aim: To implement Calculator using RMI

**CalciInterface.java**

package rmicalculator; import java.rmi.Remote;

import java.rmi.RemoteException;

public interface CalciInterface extends Remote{

double Add(double a, double b) throws RemoteException; double Sub(double a, double b) throws RemoteException; double Multiply(double a, double b) throws RemoteException; double Divide(double a, double b) throws RemoteException;

}

**RMICalciImp.java**

package rmicalculator;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class RMICalciImp extends UnicastRemoteObject implements CalciInterface{ RMICalciImp() throws Exception{}

@Override

public double Add(double a, double b) throws RemoteException { return a+b;

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public double Sub(double a, double b) throws RemoteException { return a-b;

**MUMBAI EDUCATIONAL**

**TRUST**

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public double Multiply(double a, double b) throws RemoteException { return a\*b;

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public double Divide(double a, double b) throws RemoteException { return a/b;

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

}

**RMICalciServer.java**

package rmicalculator;

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

public class RMIServer {

public static void main(String[] args){ try{

RMICalciImp rmics = new RMICalciImp(); Registry r = LocateRegistry.createRegistry(1099); r.rebind("Calci", rmics);

System.out.println("RMI server Started");

}

catch (Exception ex){}

}

}

**MUMBAI EDUCATIONAL**

**TRUST**

**RMICalciClient.java** import java.rmi.Naming; import java.util.Scanner; public class RMICalciClient

{

public static void main(String[] args)

{

String str; Scanner sc = new Scanner(System.in); try

{

CalciInterface cclient;

cclient = (CalciInterface) Naming.lookup("//localhost/Calci"); while(true)

{

System.out.println(" Calculator "); System.out.println(" 1. ADD "); System.out.println(" 2. SUBTRACT "); System.out.println(" 3. MULTIPLY "); System.out.println(" 4. DIVIDE "); System.out.println(" 5. EXIT" ); System.out.println(" ENTER YOUR CHOICE: " ); int ch = sc.nextInt();

if(ch > 5)

System.out.println("Invalid option");

else

System.out.println("Enter two values"); double val1 = sc.nextDouble();

double val2 = sc.nextDouble();

switch(ch)

**MUMBAI EDUCATIONAL**

**TRUST**

{

case 1:

System.out.println("Addition of two numbers: " + cclient.Add(val1, val2));

break; case 2:

System.out.println("Subtraction of two numbers: "

+ cclient.Sub(val1, val2)); break;

case 3:

System.out.println("Multiplication of two numbers:"

+ cclient.Multiply(val1, val2)); break;

case 4:

System.out.println("Division of two numbers:" + cclient.Divide(val1, val2));

break; case 5:

System.exit(0);

}

}

}

catch(Exception e)

{

e.printStackTrace();

}

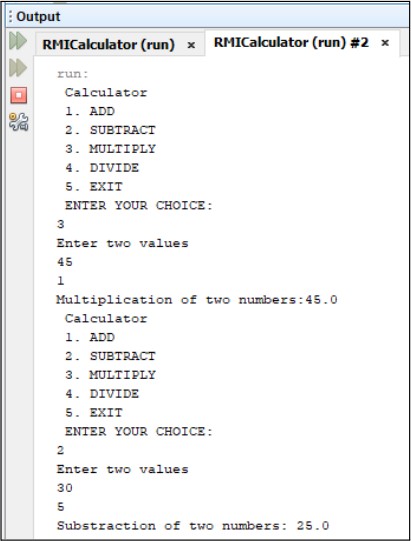
}

}

**Output:**

**MUMBAI EDUCATIONAL**

**TRUST**



**MUMBAI EDUCATIONAL**

**TRUST**

# Practical 6 Aim: To implement Date Time using RMI

**DateTimeInterface.java**

package datetimermi;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface DateTimeInterface extends Remote { String getDate() throws RemoteException;

String getTime() throws RemoteException;

}

# DateTimeServerImpl.java

package datetimermi;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject; import java.text.SimpleDateFormat;

import java.util.Date;

public class DateTimeServerImpl extends UnicastRemoteObject implements DateTimeInterface {

protected DateTimeServerImpl() throws RemoteException { super();

}

@Override

public String getDate() throws RemoteException {

**MUMBAI EDUCATIONAL**

**TRUST**

return "Date: " + new SimpleDateFormat("dd/MM/yyyy").format(new Date());

}

@Override

public String getTime() throws RemoteException {

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

}

}

# DateTimeServer.java

package datetimermi;

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

public class DateTimeServer {

public static void main(String[] args) { try {

DateTimeInterface dateTimeObj = new DateTimeServerImpl();

Registry registry = LocateRegistry.createRegistry(1099); // Default RMI registry port registry.rebind("DateTimeService", dateTimeObj);

System.out.println("DateTime Server is ready.");

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

}

}

}

**MUMBAI EDUCATIONAL**

**TRUST**

# DateTimeClient.java

package datetimermi;

/\*\*

\*

\* @author MansiPuthran

\*/

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

public class DateTimeClient {

public static void main(String[] args) { try {

Registry registry = LocateRegistry.getRegistry("localhost", 1099);

DateTimeInterface dateTimeObj = (DateTimeInterface) registry.lookup("DateTimeService");

Scanner sc = new Scanner(System.in); while (true) {

System.out.println("Enter date/time/exit"); String str = sc.nextLine();

if (str.equals("exit")) { break;

}

if (str.equals("date")) {

**MUMBAI EDUCATIONAL**

**TRUST**

System.out.println(dateTimeObj.getDate());

} else if (str.equals("time")) { System.out.println(dateTimeObj.getTime());

} else {

System.out.println("Invalid input. Enter date/time/exit");

}

}

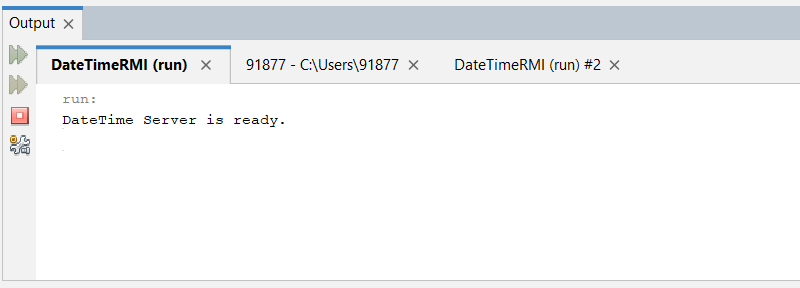
} catch (Exception e) { e.printStackTrace();

}

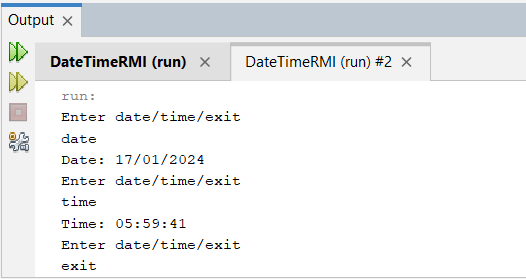
}

}

**Output:**



**MUMBAI EDUCATIONAL**

**TRUST**

**Practical 7**

**MUMBAI EDUCATIONAL**

**TRUST**

# Aim: To implement GUI Calculator using RMI

**CalciInterface.java**

package rmicalculator; import java.rmi.Remote;

import java.rmi.RemoteException;

public interface CalciInterface extends Remote{

double Add(double a, double b) throws RemoteException; double Sub(double a, double b) throws RemoteException; double Multiply(double a, double b) throws RemoteException; double Divide(double a, double b) throws RemoteException;

}

**RMICalciImp.java**

package rmicalculator;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class RMICalciImp extends UnicastRemoteObject implements CalciInterface{ RMICalciImp() throws Exception{}

@Override

public double Add(double a, double b) throws RemoteException { return a+b;

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public double Sub(double a, double b) throws RemoteException { return a-b;

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

**MUMBAI EDUCATIONAL**

**TRUST**

}

@Override

public double Multiply(double a, double b) throws RemoteException { return a\*b;

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public double Divide(double a, double b) throws RemoteException { return a/b;

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

}

**RMICalciServer.java**

package rmicalculator;

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

public class RMIServer {

public static void main(String[] args){ try{

RMICalciImp rmics = new RMICalciImp(); Registry r = LocateRegistry.createRegistry(1099); r.rebind("Calci", rmics);

System.out.println("RMI server Started");

}

catch (Exception ex){}

}

}

**RMIGUIClient.java**

**MUMBAI EDUCATIONAL**

**TRUST**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package rmicalculator;

import java.awt.Container; import java.awt.GridLayout;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.rmi.Naming;

import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JPanel; import javax.swing.JTextField;

/\*\*

* @author mcamock

\*/

public class RMIGUIClient extends JFrame implements ActionListener{ double n1 = 0.0;

double n2 = 0.0; double d1;

JButton b[] = new JButton[21]; JTextField tf;

Container con; int button,j;

String str, num=""; JPanel tp,bp;

public RMIGUIClient(){ setTitle("Calculator"); tp = new JPanel(); bp = new JPanel();

**MUMBAI EDUCATIONAL**

**TRUST**

tf = new JTextField(10);

tf.setEditable(false);

con = getContentPane(); bp.setLayout(new GridLayout(5,4)); tp.add(tf);

con.add(tp,"North"); for(int i=0; i<10;i++)

{

b[i] = new JButton(""+i);

}

b[10] = new JButton("+");

b[11] = new JButton("-");

b[12] = new JButton("\*");

b[13] = new JButton("/"); b[14] = new JButton("clear"); b[15] = new JButton(".");

b[16] = new JButton("="); for(int i=0;i<17;i++){

b[i].addActionListener(this); bp.add(b[i]);

}

con.add(bp,"Center"); setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

@Override

public void actionPerformed(ActionEvent ae) { str = ae.getActionCommand();

for(int i=0;i<10;i++){ if(ae.getSource()==b[i])

{

num = num+str; tf.setText(num); System.out.println(num);

//System.out.println(num);

}

}

**MUMBAI EDUCATIONAL**

**TRUST**

if(ae.getSource()==b[15])

{

num = num+str; tf.setText(num);

}

for(int i =10;i<14;i++){ if(ae.getSource()==b[i])

{

button = i-9; if(num!="")

{

n1 = Double.parseDouble(num); num=""; System.out.println("N1="+n1);

System.out.println("Operator="+b[i].getText());

}

}

}

if(ae.getSource()==b[16])

{

n2 = Double.parseDouble(num); System.out.println("N2= "+n2); num="";

try{

CalciInterface cclient;

cclient =(CalciInterface) Naming.lookup("//localhost/Calci"); switch(button)

{

case 1: d1= cclient.Add(n1, n2); break;

case 2: d1= cclient.Sub(n1, n2); break;

case 3: d1= cclient.Multiply(n1, n2); break;

case 4: d1= cclient.Divide(n1, n2); break;

default: d1=0.0;

**MUMBAI EDUCATIONAL**

**TRUST**

}

str = String.valueOf(d1);

n1 = d1; tf.setText(str);

}

catch(Exception ex){

}

}

if(ae.getSource()==b[14]){

tf.setText(""); num=""; n1=0.0; n2=0.0;

button=0;

}

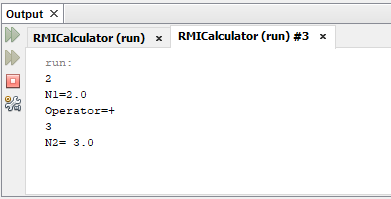
}

public static void main(String[] args){ JFrame f = new RMIGUIClient(); f.setSize(300, 300); f.setVisible(true);

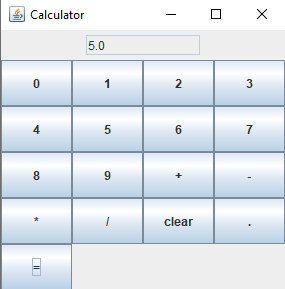
}

}

**Output:**



**MUMBAI EDUCATIONAL**

**TRUST**

# Practical 8

**Aim: Using MySQL create Library database. Create table Book (Book\_id, Book\_name, Book\_author) and retrieve the Book information from Library database using Remote Object Communication.**

**LibraryInterface.java**

/\*

* To change this license header, choose License Headers in Project Properties.

**MUMBAI EDUCATIONAL**

**TRUST**

* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package rmilibrary;

import java.rmi.Remote;

import java.rmi.RemoteException;

/\*\*

* ​
* @author mcamock

\*/

public interface LibraryInterface extends Remote{ public String getData() throws RemoteException;

}

**LibraryServer.java**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package rmilibrary;

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

import java.rmi.server.UnicastRemoteObject; import java.sql.Connection;

import java.sql.DriverManager; import java.sql.ResultSet;

import java.sql.ResultSetMetaData; import java.sql.SQLException; import java.sql.Statement;

import java.util.logging.Level; import java.util.logging.Logger;

import javax.naming.spi.DirStateFactory.Result;

/\*\*

**MUMBAI EDUCATIONAL**

**TRUST**

* ​
* @author mcamock

\*/

public class LibraryServer extends UnicastRemoteObject implements LibraryInterface{

int count =0, i=0; String str, columname;

LibraryServer() throws RemoteException{ super();

}

@Override

public String getData() throws RemoteException { try{

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

DriverManager.getConnection("jdbc:mysql://localhost:3306/dscc","root",""); Statement st = con.createStatement();

ResultSet rs1;

rs1 = st.executeQuery("select \* from library"); ResultSetMetaData rmd = rs1.getMetaData(); str = "";

columname = ""; for(i=1;i<=rmd.getColumnCount();i++){

columname = columname +rmd.getColumnName(i)+"\t\t";

}

while(rs1.next()){ for(i=1;i<=rmd.getColumnCount();i++){

str = str+rs1.getString(i)+"\t\t";

}

str = str+"\n";

}

str = columname+"\n"+str;

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

}

return str;

}

public static void main(String[] args){ try{

Registry reg = LocateRegistry.createRegistry(2099);

**MUMBAI EDUCATIONAL**

**TRUST**

LibraryServer obj = new LibraryServer(); reg.rebind("db", obj); System.out.println("Database server ready");

}

catch(Exception e){

System.out.println(e);

}

}

}

**LibraryClient.java**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package rmilibrary; import java.rmi.Naming;

/\*\*

* ​
* @author mcamock

\*/

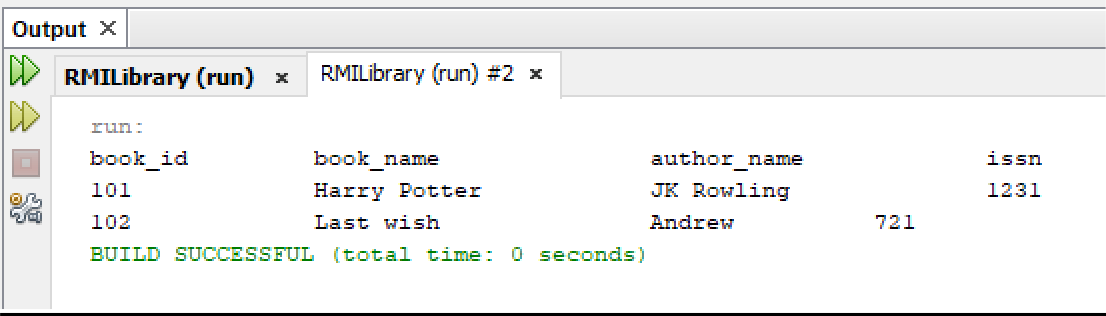
public class LibraryClient {

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

LibraryInterface obj = (LibraryInterface) Naming.lookup("//localhost:2099/db"); System.out.print(obj.getData());

}

}

**Output:**

**MUMBAI EDUCATIONAL**

**TRUST**

**Practical 9**

**Aim: Using MySQL create Elecrtic\_Bill database. Create table Bill(consumer\_no,billduedate, Billamt) and retrieve the bill information from Bill table using Remote Object Communication**

**ElectricBillInterface.java**

package electricbillrmi; import java.rmi.Remote;

**MUMBAI EDUCATIONAL**

**TRUST**

import java.rmi.RemoteException;

public interface ElectricBillInterface extends Remote { String getBillData() throws RemoteException;

}

# ElectricBillServerImpl.java

package electricbillrmi;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject; import java.sql.Connection;

import java.sql.DriverManager; import java.sql.ResultSet;

import java.sql.ResultSetMetaData; import java.sql.Statement;

public class ElectricBillServerImpl extends UnicastRemoteObject implements ElectricBillInterface {

public ElectricBillServerImpl() throws RemoteException { super();

}

@Override

public String getBillData() throws RemoteException { String result = "";

try {

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

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dscc", "root", "");

Statement st = con.createStatement();

**MUMBAI EDUCATIONAL**

**TRUST**

ResultSet rs = st.executeQuery("select \* from Bill"); ResultSetMetaData rmd = rs.getMetaData();

// Build column names String columnNames = "";

for (int i = 1; i <= rmd.getColumnCount(); i++) {

columnNames = columnNames + rmd.getColumnName(i) + "\t\t";

}

// Build result string while (rs.next()) {

for (int i = 1; i <= rmd.getColumnCount(); i++) { result = result + rs.getString(i) + "\t\t\t";

}

result = result + "\n";

}

result = columnNames + "\n" + result;

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

}

return result;

}

}

# ElectricBillServer.java

package electricbillrmi;

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

**MUMBAI EDUCATIONAL**

**TRUST**

public class ElectricBillServer {

public static void main(String[] args) { try {

ElectricBillInterface electricBillObj = new ElectricBillServerImpl();

Registry registry = LocateRegistry.createRegistry(2098); registry.rebind("ElectricBillService", electricBillObj);

System.out.println("Electric Bill Server is ready.");

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

}

}

}

# ElectricBillClient.java

package electricbillrmi; import java.rmi.Naming;

public class ElectricBillClient {

public static void main(String[] args) { try {

ElectricBillInterface electricBillObj = (ElectricBillInterface) Naming.lookup("//localhost:2098/ElectricBillService");

System.out.print(electricBillObj.getBillData());

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

}

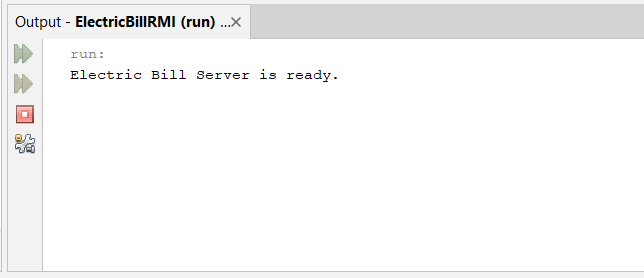
}

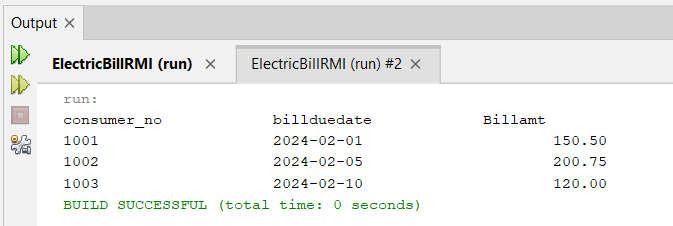
**MUMBAI EDUCATIONAL**

**TRUST**

}

# Output:





**Practical 10 Aim: Implementation of Shared Memory SharedMemoryServer:**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

\*/

package sharedmemory;

import java.io.BufferedReader; import java.io.InputStreamReader;

**MUMBAI EDUCATIONAL**

**TRUST**

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

* ​
* @author mcamock

\*/

public class SharedMemoryServer { static int a = 50;

static int count = 0;

public static int getA(PrintStream cout){ count++;

cout.println(a); return a;

}

public static void main(String[] args) throws Exception{ ServerSocket ss = new ServerSocket(2000); Socket sk = ss.accept();

BufferedReader cin = new BufferedReader(new InputStreamReader(sk.getInputStream())); PrintStream cout = new PrintStream(sk.getOutputStream());

while(true){

System.out.println("Client from "+sk.getInetAddress().getHostAddress()+" accepted"); String sr = cin.readLine();

System.out.println(sr); if(sr.equalsIgnoreCase("show")){

getA(cout);

}

else{

cout.println("Check Syntax");

}

System.out.println("Count: "+count);

}

}

}

**SharedMemoryClient.java:**

/\*

* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.

**MUMBAI EDUCATIONAL**

**TRUST**

\*/

package sharedmemory;

import java.io.BufferedReader; import java.io.InputStreamReader; import java.io.PrintStream;

import java.net.ServerSocket; import java.net.Socket; import java.util.Scanner;

/\*\*

* ​
* @author mcamock

\*/

public class SharedMemoryClient {

public static void main(String[] args) throws Exception{ Socket sk = new Socket("localhost",2000);

BufferedReader sin = new BufferedReader(new InputStreamReader(sk.getInputStream())); PrintStream sout = new PrintStream(sk.getOutputStream());

Scanner stdin = new Scanner(System.in); String sr,sw;

while(true){ System.out.println("type show"); sw = stdin.nextLine(); sout.println(sw);

sr = sin.readLine(); System.out.println("Answer: "+sr);

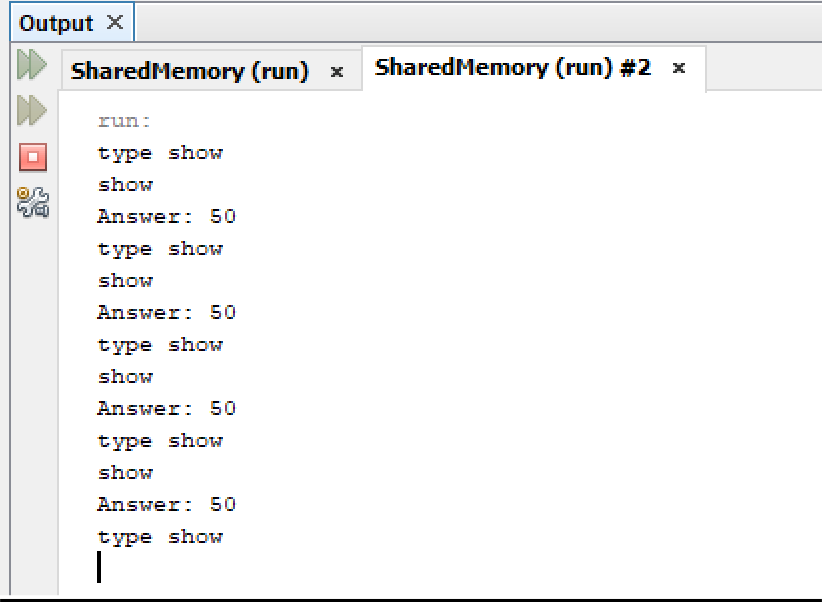
}

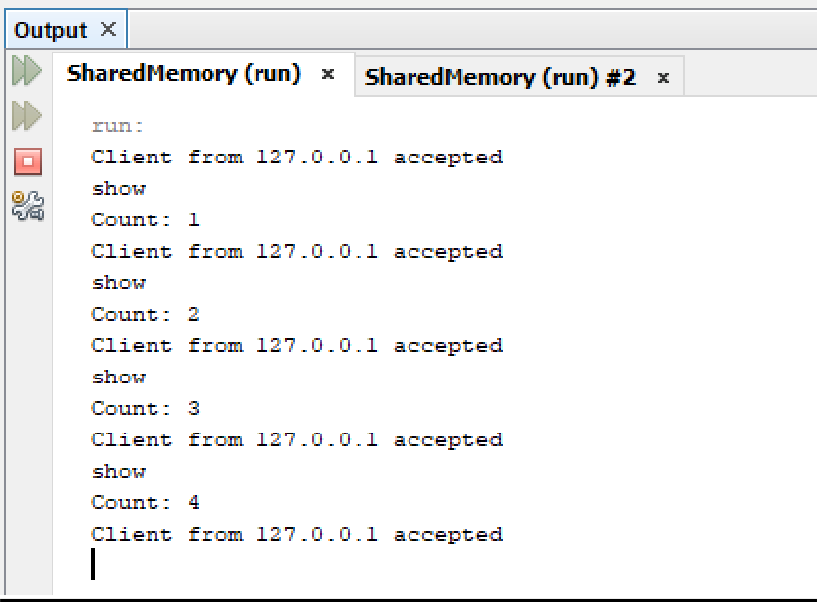
}

}

**Output:**

**MUMBAI EDUCATIONAL**

**TRUST**



**Practical 11**

**Aim: Implementation of mutual exclusion using token Ring**

**MUMBAI EDUCATIONAL**

**TRUST**

# TokenRingServer.java

package tokenringapp;

import java.net.\*; //DatagramSocket and DatagramPacket

/\*\*

\*

* @author MansiPuthran

\*/

public class TokenRingServer { public static DatagramSocket ds; public static DatagramPacket dp;

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

{

ds = new DatagramSocket(2000);

while(true)

{

//pre-allocated space for receiving message byte[] buffer = new byte[1024];

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

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

}

}

}

**TokenRingClient1.java**

**MUMBAI EDUCATIONAL**

**TRUST**

package tokenringapp;

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

/\*\*

* @author MansiPuthran

\*/

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) //the current process has token

{

System.out.println("Do you want to enter criticalsection...(yes/no)? "); br = new BufferedReader(new InputStreamReader(System.in)); String choice = br.readLine();

//if the choice is yes, the process or client will write data to TokenRingServer

**MUMBAI EDUCATIONAL**

**TRUST**

if(choice.equalsIgnoreCase("yes"))

{

System.out.println("the client or process is ready to write ");

System.out.println("Enter the message");

br = new BufferedReader(new InputStreamReader(System.in)); String message = "Client--> "+br.readLine();

dp = new DatagramPacket(message.getBytes(),message.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 critical section"); String msg1="Token";//this msg will pass to the next process

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("PreviousProcessMsg is"+prevProcessMsg); if(prevProcessMsg.equals("Token"))

{

**MUMBAI EDUCATIONAL**

**TRUST**

hasToken=true;

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

}

}

}

}

}

# TokenRingClient2.java

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

/\*\*

\*

* @author MansiPuthran

\*/

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)//the current procss has token

{

System.out.println("do you want to enter CS..(yes/no)");

**MUMBAI EDUCATIONAL**

**TRUST**

br=new BufferedReader(new InputStreamReader(System.in));//yes or no String choice =br.readLine();

//if the choice is yes, the process or client will write data to the TokenRingServer if(choice.equalsIgnoreCase("yes"))

{

System.out.println("the client or process is ready to write");

System.out.println("enter the message");

br=new BufferedReader(new InputStreamReader(System.in)); String msg = "Client2-->"+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 critical section"); String msg1="Token";//this msg will pass to the next process 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());

**MUMBAI EDUCATIONAL**

**TRUST**

System.out.println("PreviousProcessMsg is"+prevProcessMsg); if(prevProcessMsg.equals("Token"))

{

hasToken=true;

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

}

}

}

}

}

# TokenRingClient3.java

package tokenringapp;

//if the process doesn't want to the enter to the critical section

//it will pass the token to the next process import java.net.\*;

import java.io.\*;

/\*\*

\*

* @author MansiPuthran

\*/

public class TokenRingClient3 {

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

**MUMBAI EDUCATIONAL**

**TRUST**

public static void main(String[] args) throws Exception { boolean hasToken = true;

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

if (hasToken == true)//the current procss has token

{

System.out.println("do you want to enter CS..(yes/no)");

br = new BufferedReader(new InputStreamReader(System.in)); String choice = br.readLine();

//if the choice is yes, the process or client will write data to the TokenRingServer

if (choice.equalsIgnoreCase("yes")) {

System.out.println("the client or process is ready to write");

System.out.println("enter the message");

br = new BufferedReader(new InputStreamReader(System.in)); String msg = "Client3-->" + 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 critical section"); String msg1 = "Token";//this msg will pass to the next process dp = new DatagramPacket(msg1.getBytes(), msg1.length(),

InetAddress.getLocalHost(), 2001);

**MUMBAI EDUCATIONAL**

**TRUST**

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("PreviousProcessMsg is" + prevProcessMsg); if (prevProcessMsg.equals("Token")) {

hasToken = true;

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

}

}

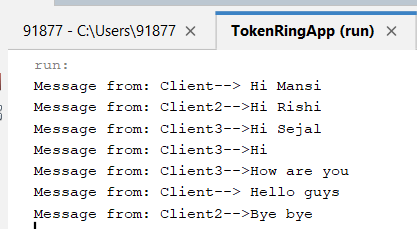
}

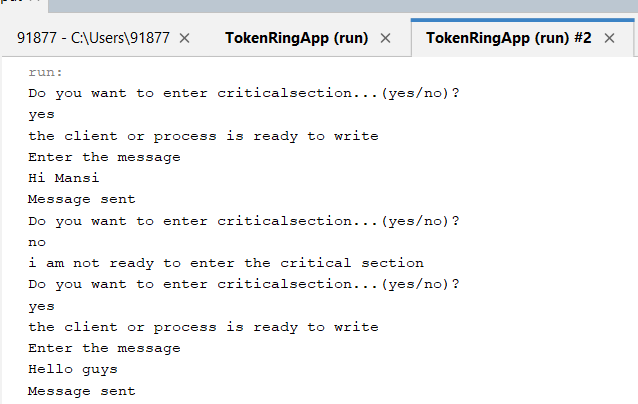
}

}

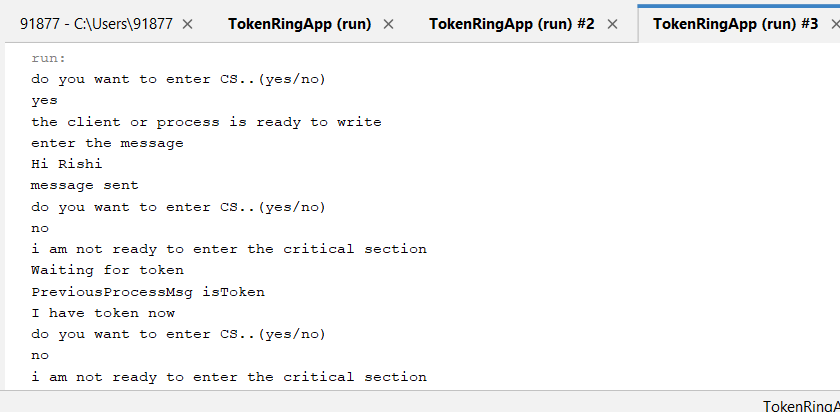
**Output:**

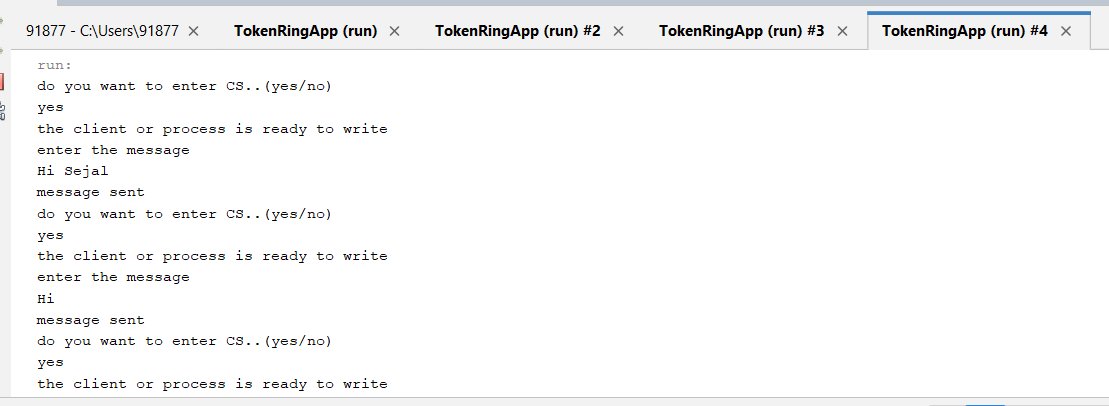
**MUMBAI EDUCATIONAL**

**TRUST**



**MUMBAI EDUCATIONAL**

**TRUST**



**MUMBAI EDUCATIONAL**

**TRUST**

**Practical 12**

**Aim: To develop Application for windows using Windows Azure Platform Training Kit and Visual Studio.**

**Code:**

**WebForm1.aspx**

**<!DOCTYPE html>**

**<html xmlns=”**[**http://www.w3.org/1000/xhtml**](http://www.w3.org/1000/xhtml)**”>**

# <head runat=”server”>

**<title></title>**

**</head>**

**<body>**

# <form id=”form1” runat=”server”>

**<div>**

**Hello World !! This is SYMCA**

**</div>**

**</form>**

**</body>**

**</html> Output:**

**MUMBAI EDUCATIONAL**

**TRUST**



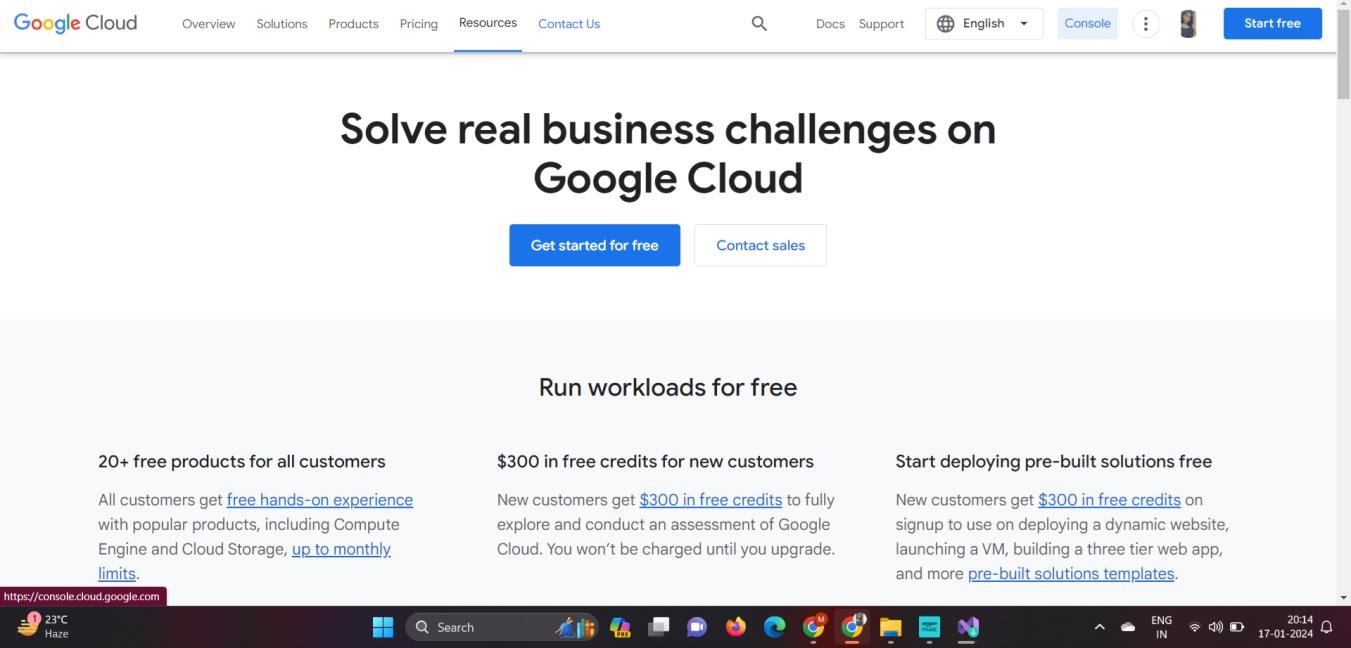
**MUMBAI EDUCATIONAL**

**TRUST**

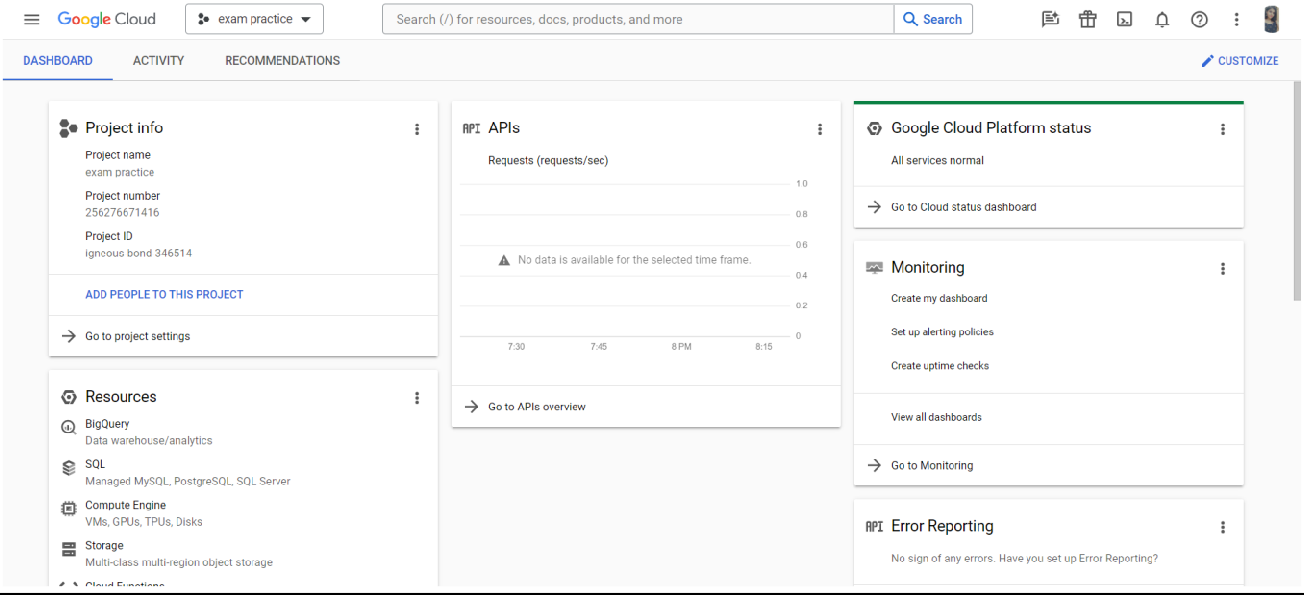
# Practical 13

**Aim: To develop Application using Google App Engine**

Click on Console on top right



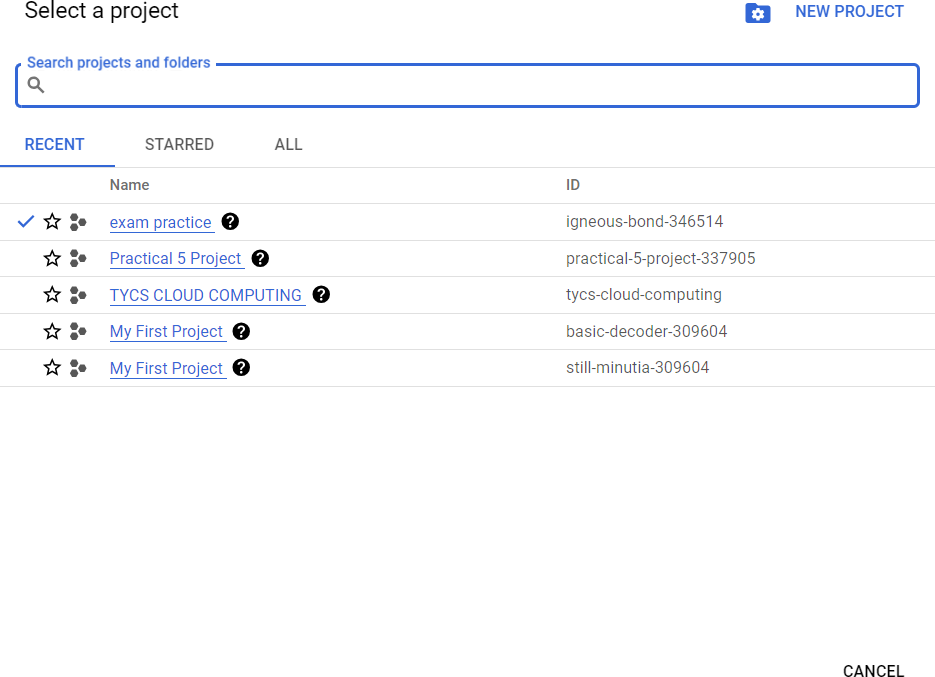
Click over current project next to GCP



**MUMBAI EDUCATIONAL**

**TRUST**

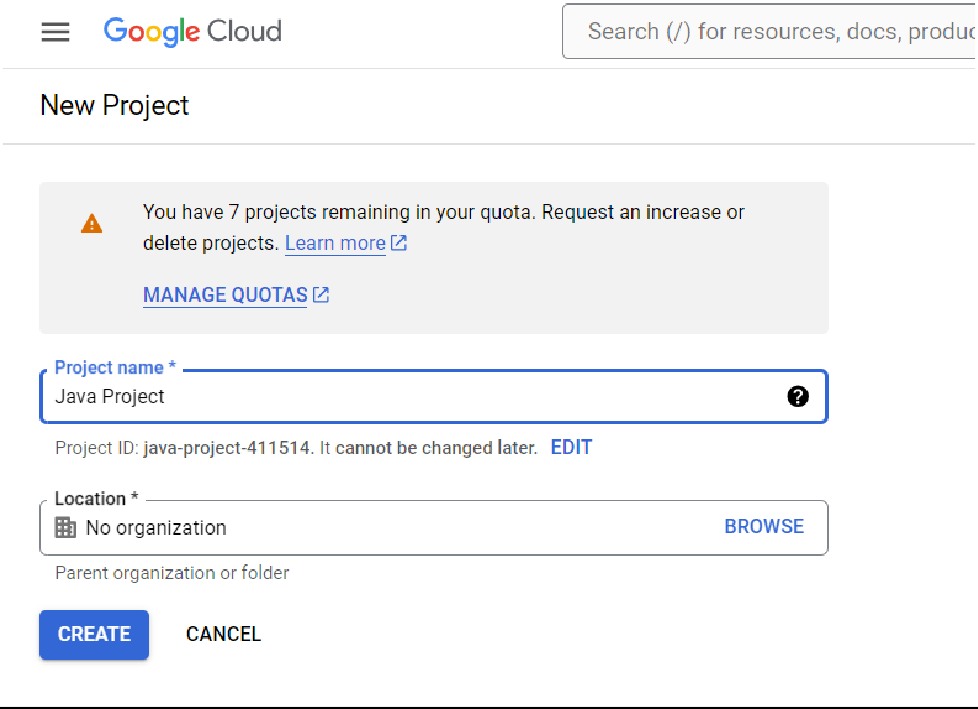
Click on New Project



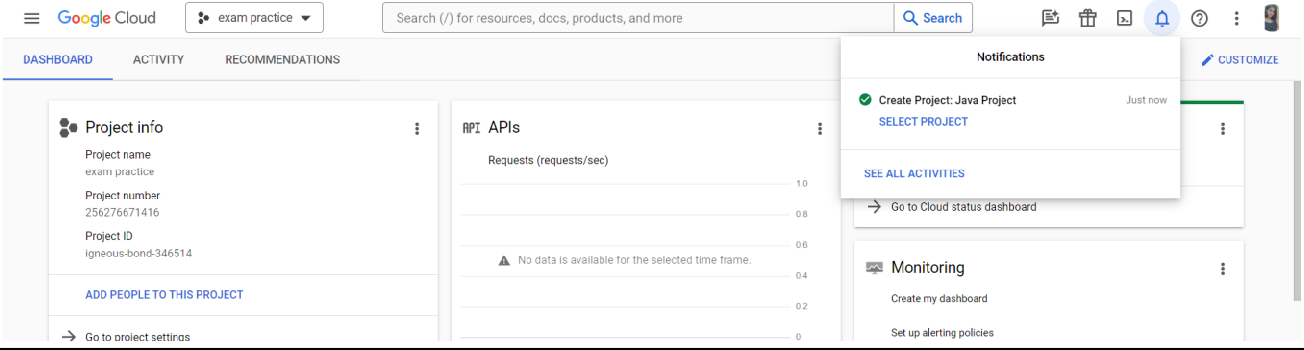
Give some useful name for project and click on create

**MUMBAI EDUCATIONAL**

**TRUST**



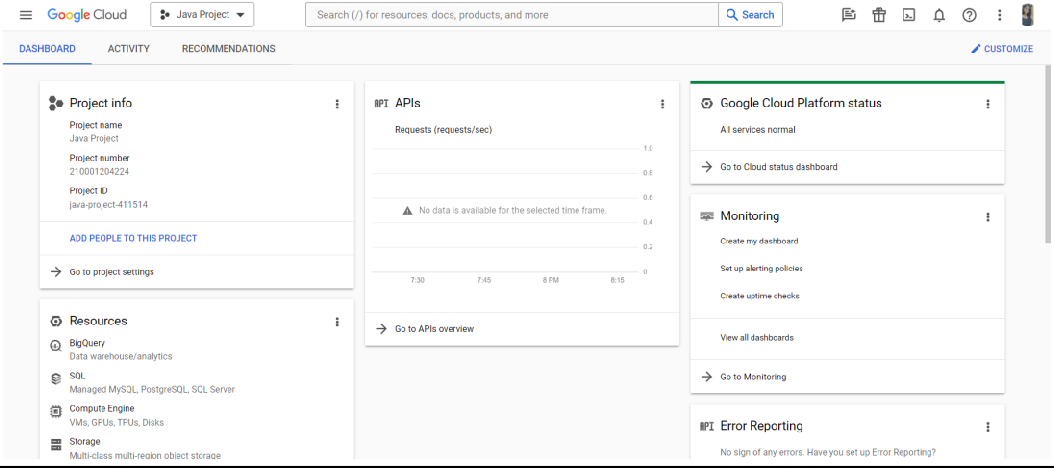
Now the project is created but still we’ve the old project selected so click on current project next to GCP and select the newly created project



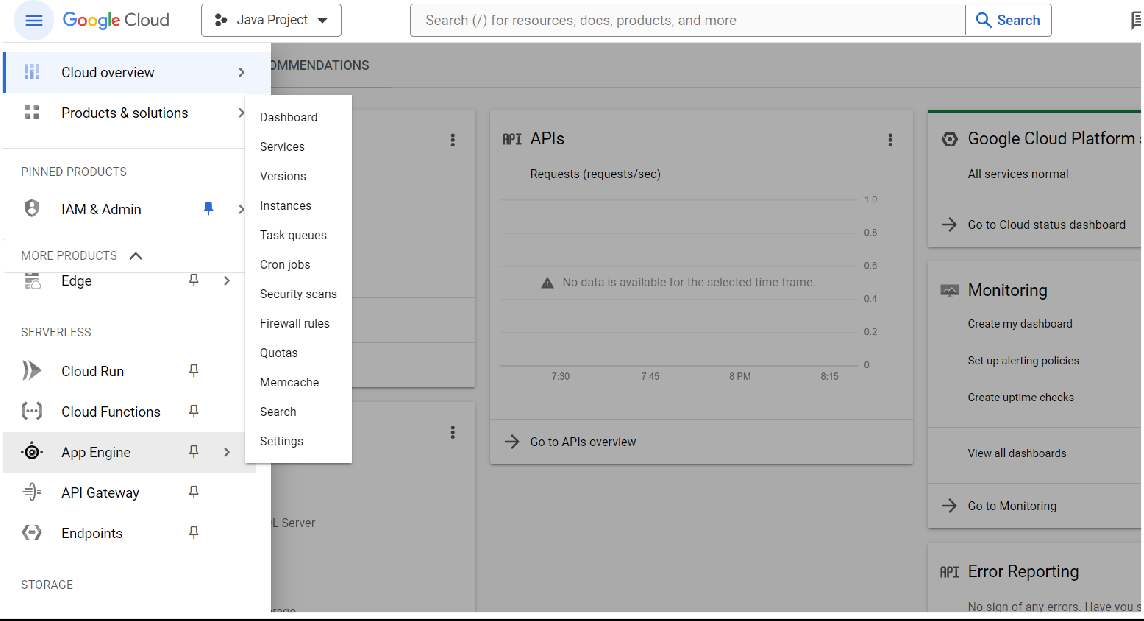
Now the project is selected, the interface will look like as below

**MUMBAI EDUCATIONAL**

**TRUST**



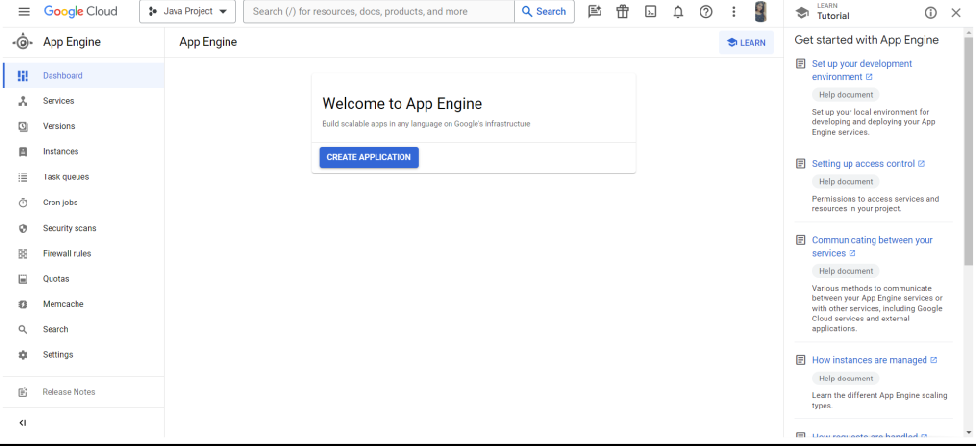
Select App engine from left panel



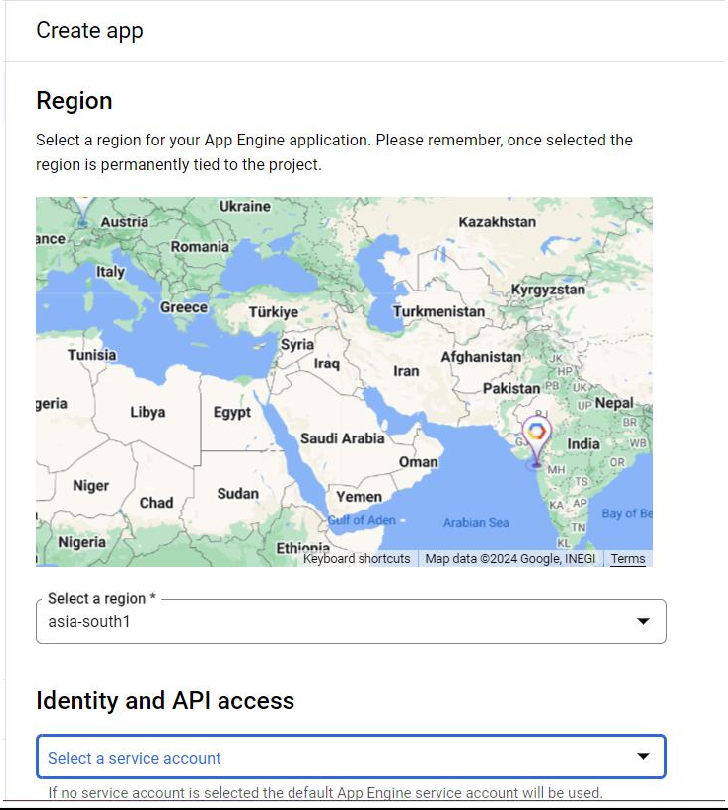
The interface after clicking on App engine will look like below

**MUMBAI EDUCATIONAL**

**TRUST**



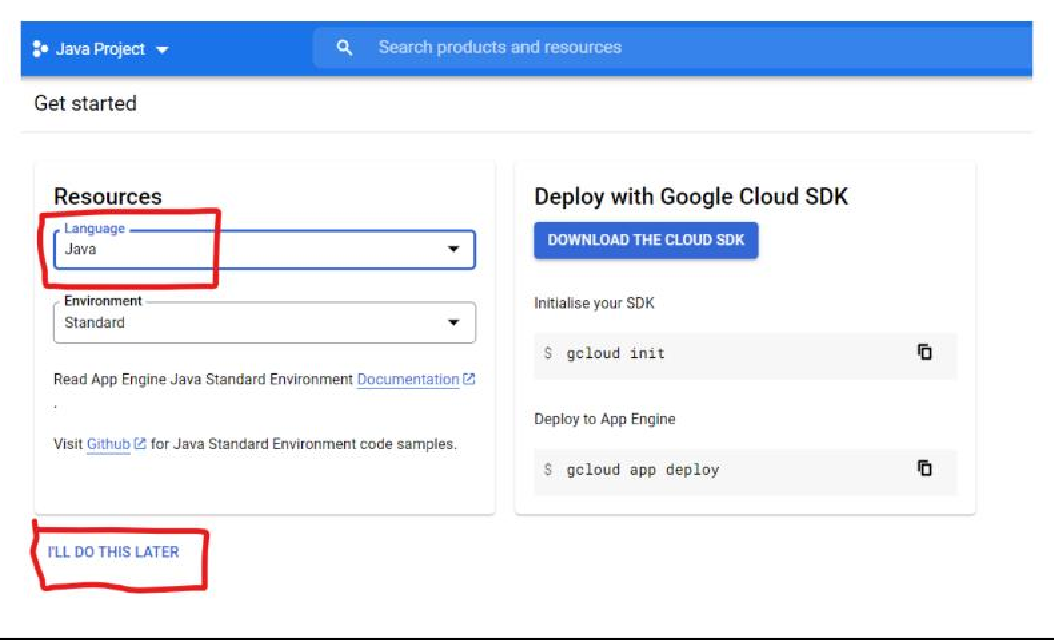
Click on Create Application then select the region in next slide and click Next



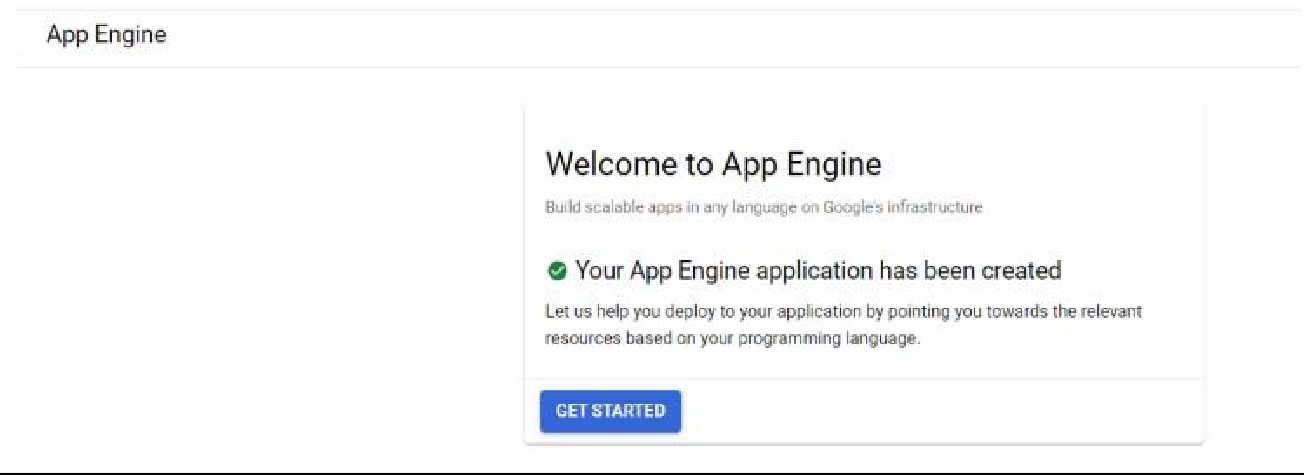
Now select the language and click I’ll do this later

**MUMBAI EDUCATIONAL**

**TRUST**



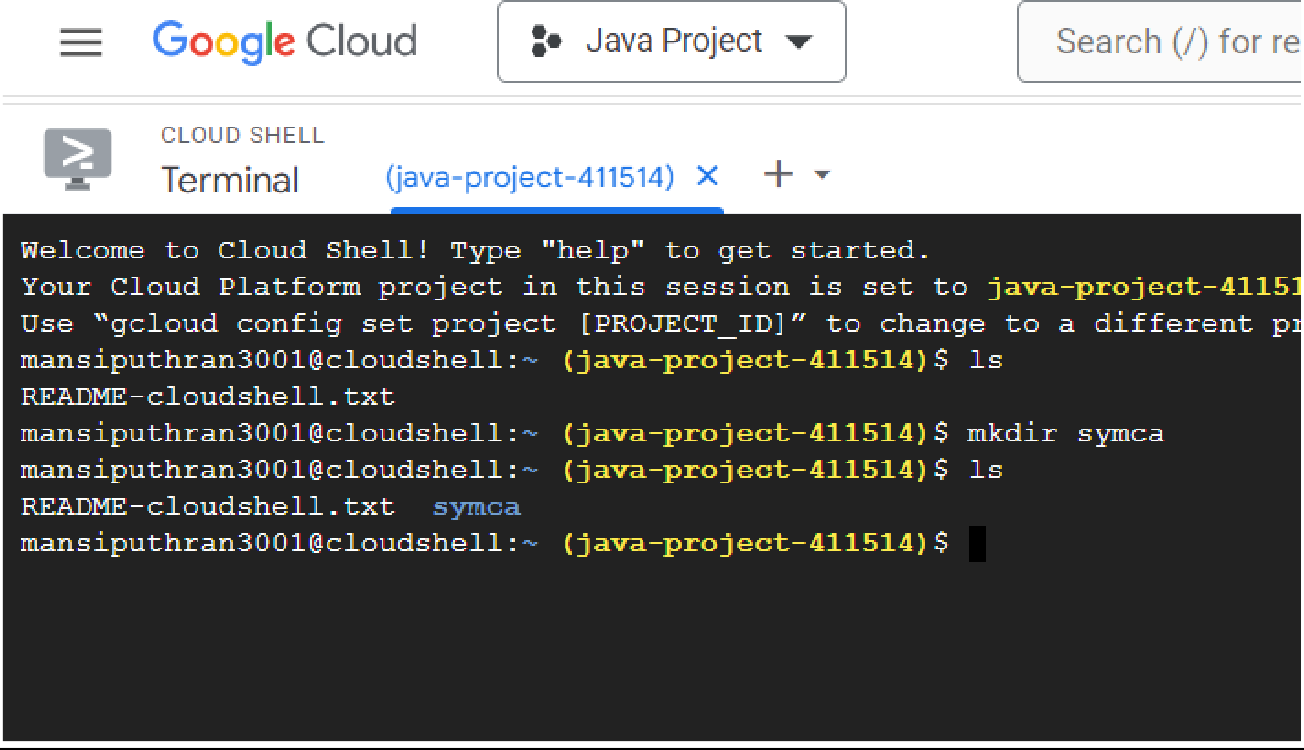
Now the App Engine Application has been created



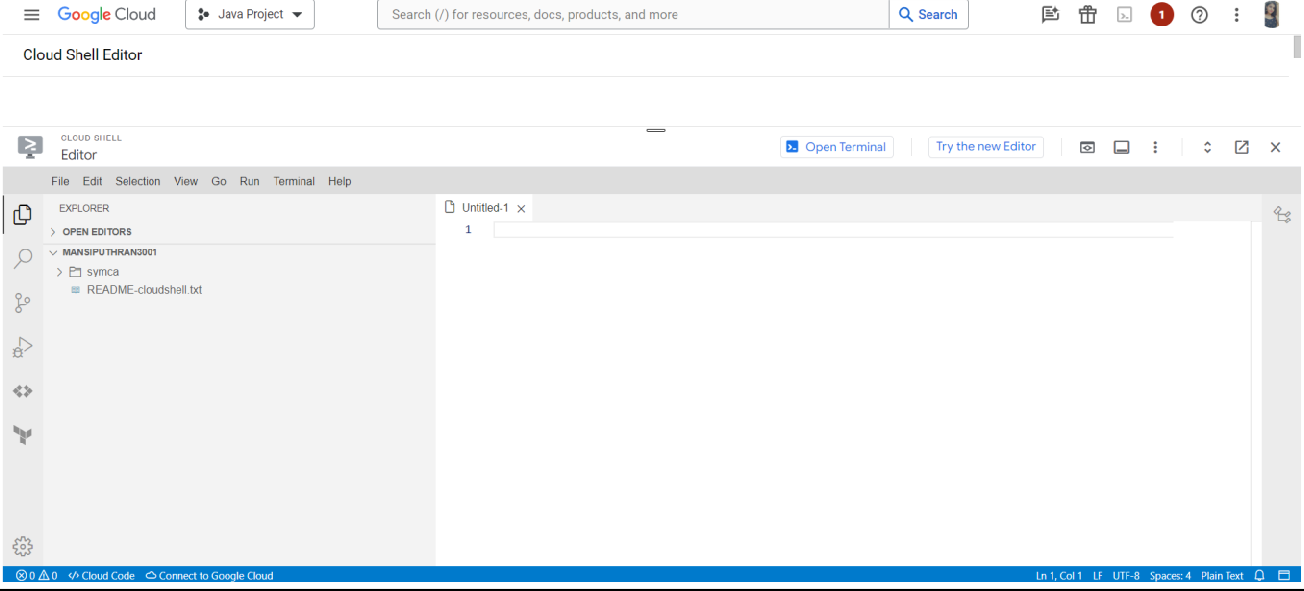
Search Cloud shell and click on Open terminal it will open the command shell to work

**MUMBAI EDUCATIONAL**

**TRUST**

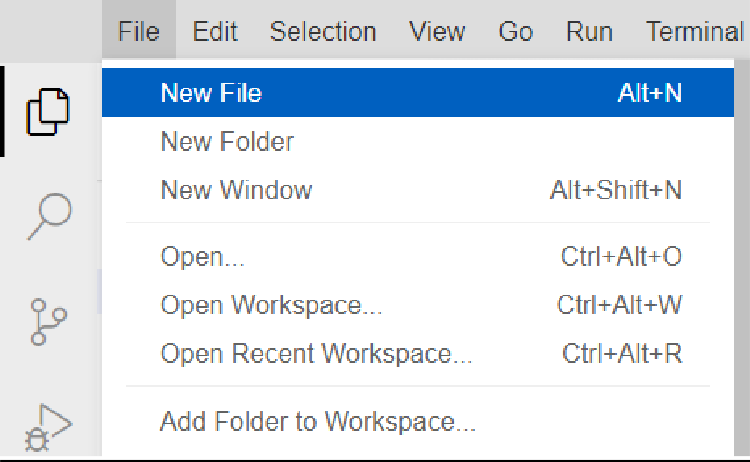


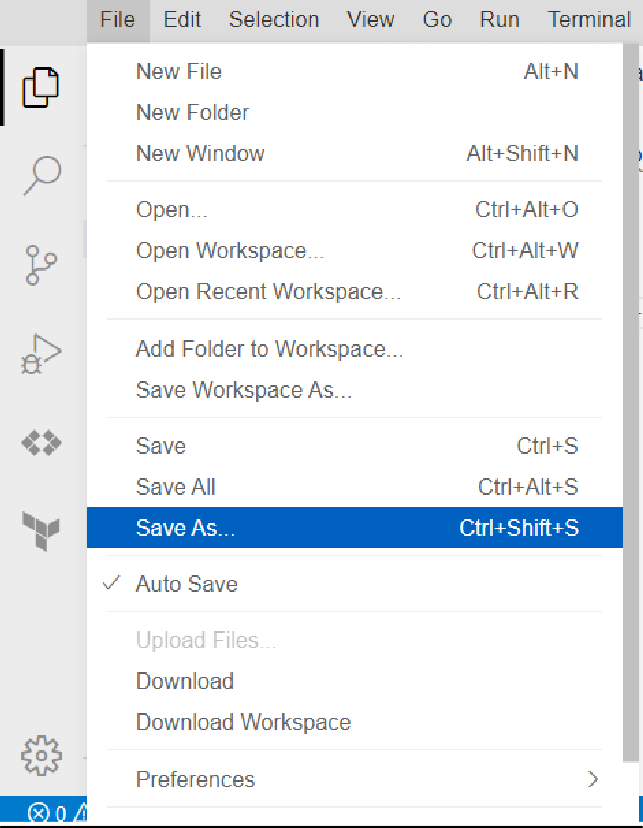
Click on open editor and then click on Legacy Editor



Create Demo.java

**MUMBAI EDUCATIONAL**

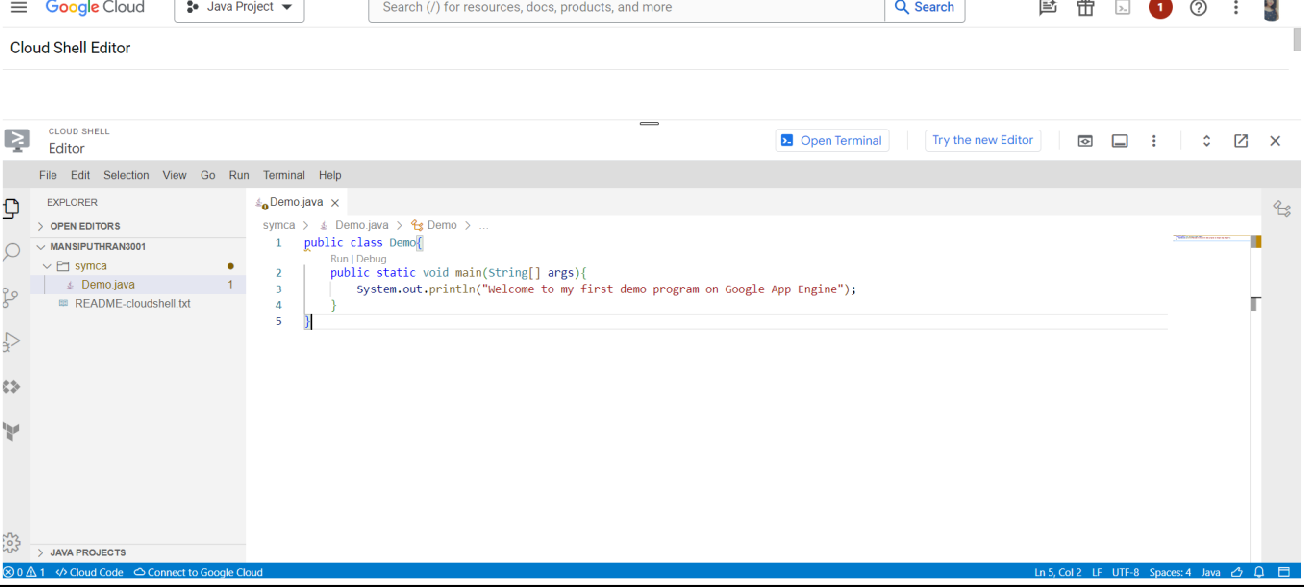
**TRUST**



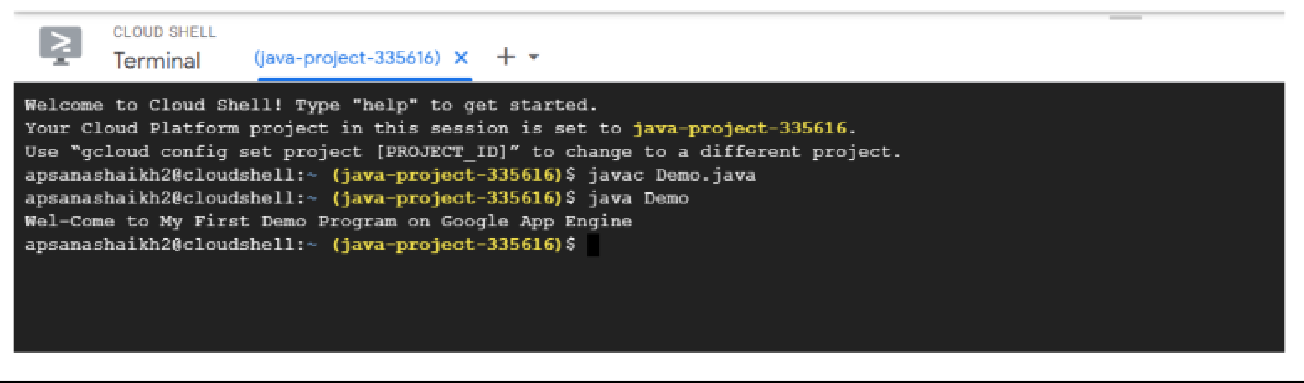
Now write the Demo.java program

**MUMBAI EDUCATIONAL**

**TRUST**



Open the terminal and run the program and see the output

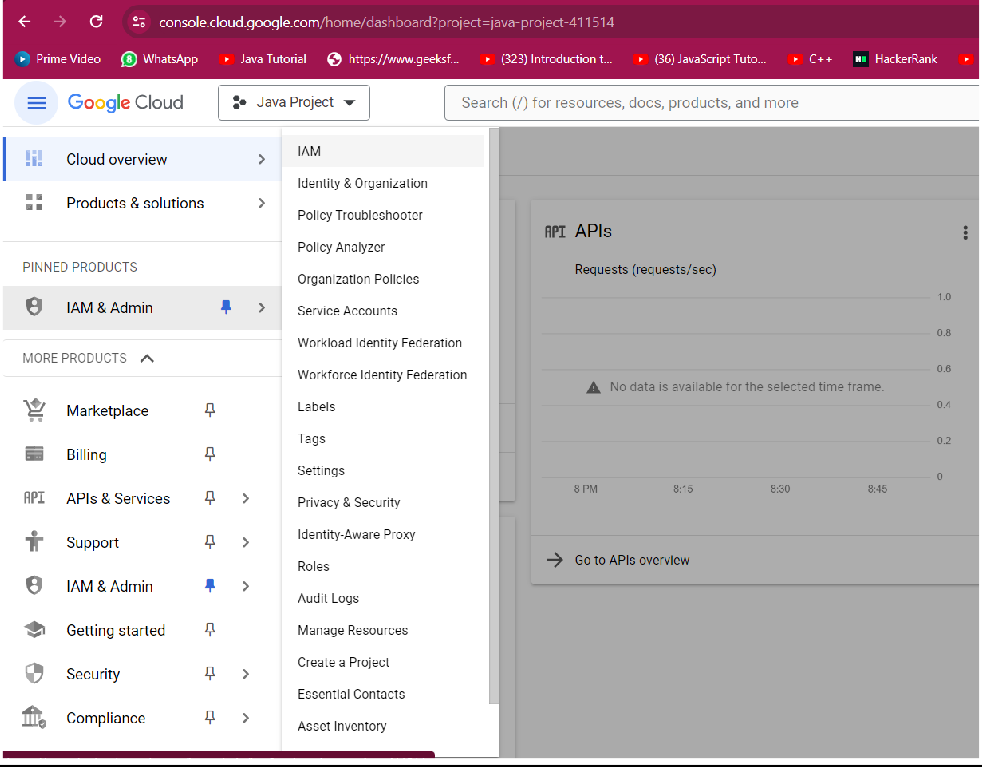


**MUMBAI EDUCATIONAL**

**TRUST**

# Practical 14 Aim: To implement Identity Management in GCP

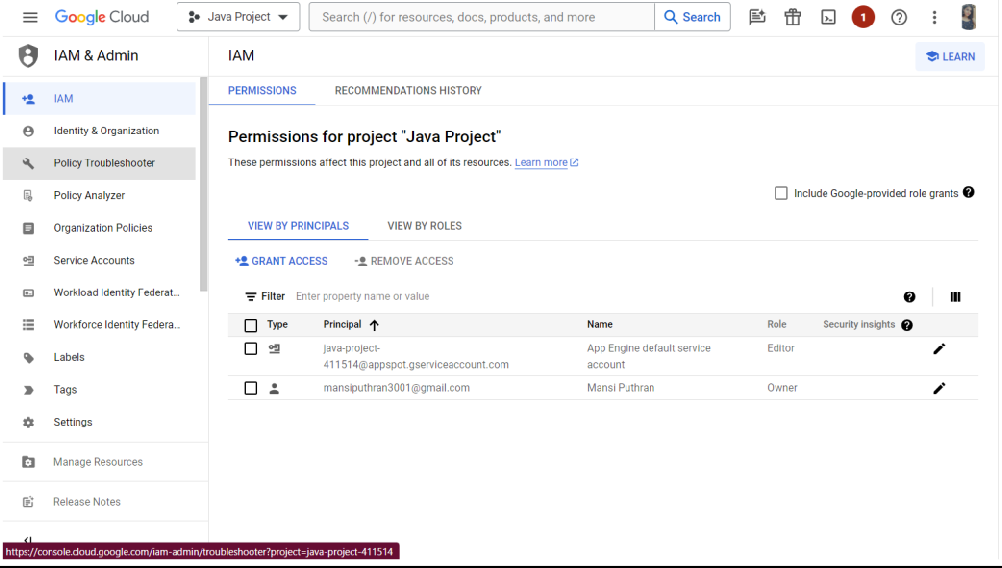
Select a project and select IAM



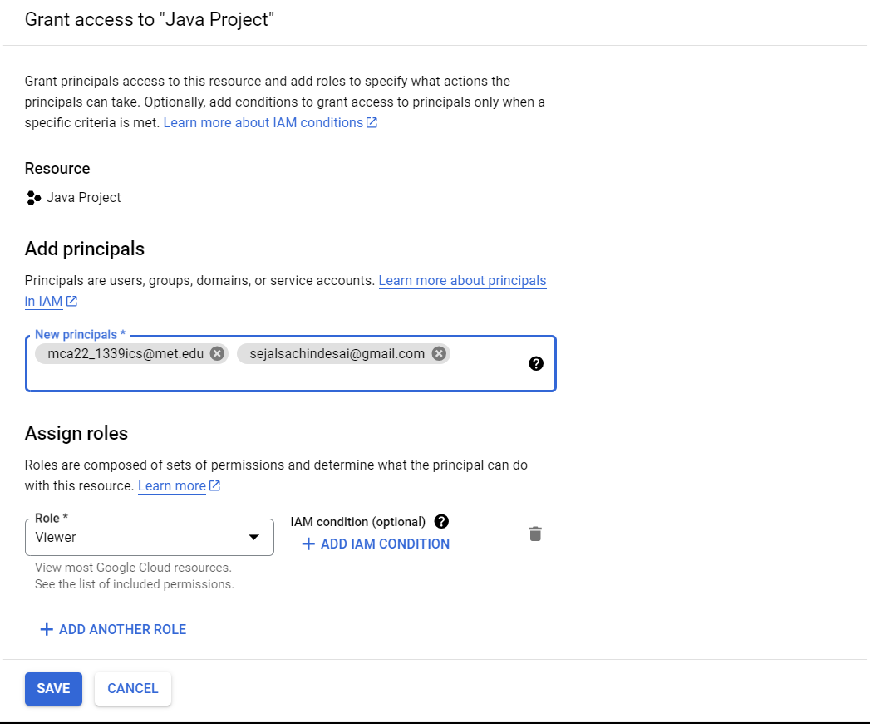
Click on Grant access for you project

**MUMBAI EDUCATIONAL**

**TRUST**



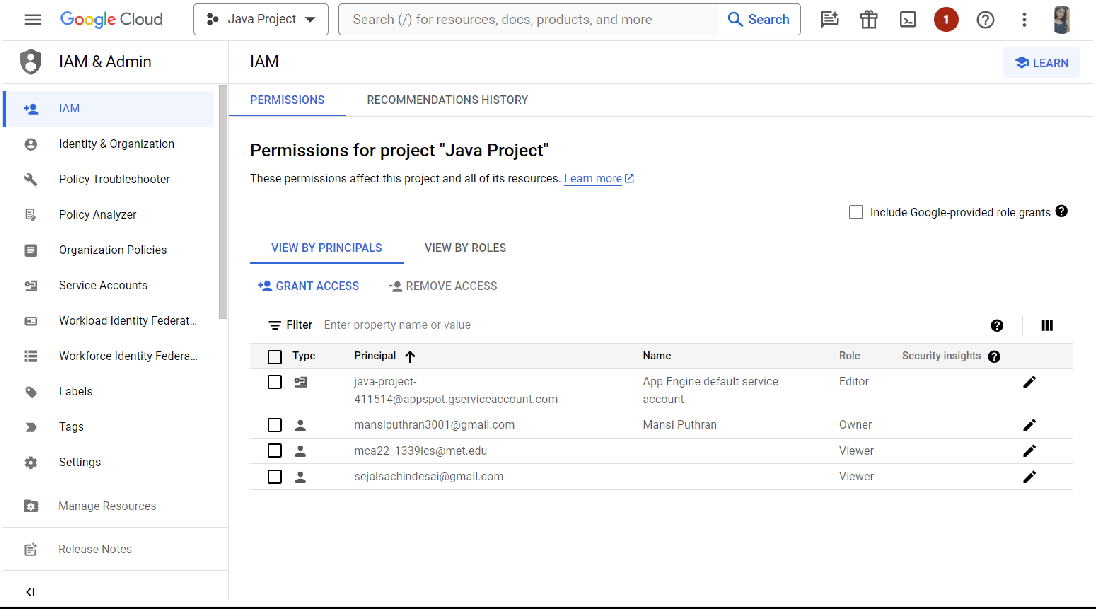
Add new principals and assign roles



It will look like this

**MUMBAI EDUCATIONAL**

**TRUST**



**MUMBAI EDUCATIONAL**

**TRUST**

**Practical 15**

**Aim: Implement Storage as Service on Google Cloud**

**Getting started**

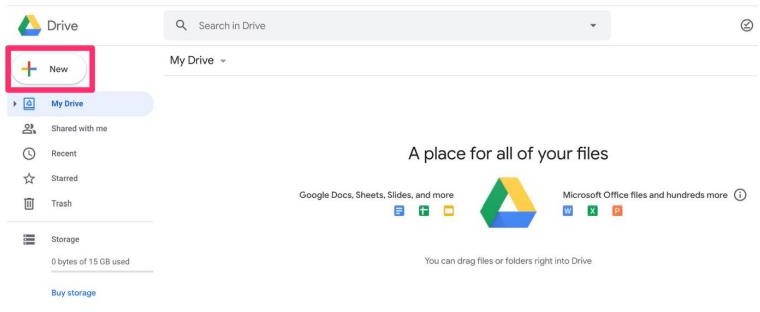
To get started with Google Drive, you'll need to make a Google account, if you don't already have one.

Creating a Google account is free, and gives you access to Google Drive, as well as other Google Services, such as Gmail, Google Calendar, and Google Photos.

Once you've signed up for an account (or signed into an existing account), you can access Google Drive in your browser by going to drive.google.com. This will bring you to Drive's web interface, which is fairly intuitive and easy to navigate.

As you begin to familiarize yourself with all the things you can do with Google Drive, you'll most likely want to first learn how to upload, create, manage and share files.

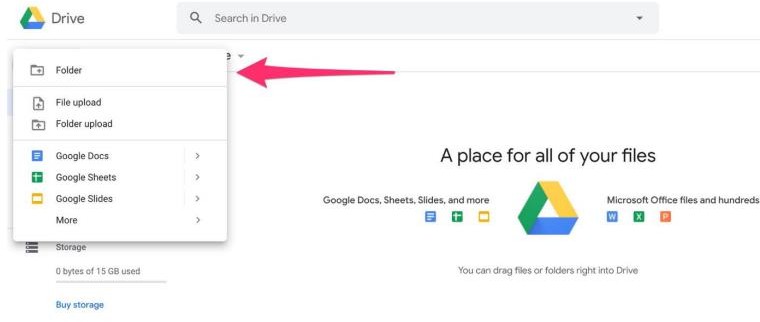
**Uploading and creating files**



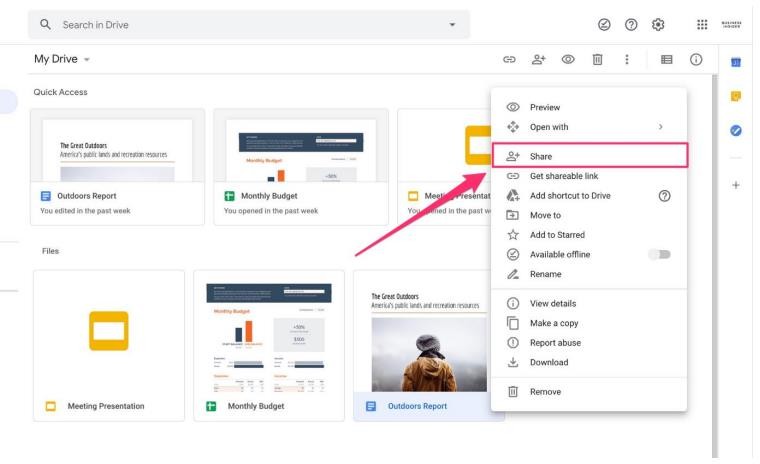
**Managing and organizing files and folders**

**MUMBAI EDUCATIONAL**

**TRUST**



# Sharing files and folders



The biggest draw of Google Drive is the ability to share files and folders with others. From sharing videos to entire Drive folders, Drive makes sharing simple.

To share a Google Drive file or folder, right-click on it to bring up the context menu, then click "Share." From there, you'll be able to add collaborators using their email address, and decide whether people can edit, comment on, or simply just view the file.

**MUMBAI EDUCATIONAL**

**TRUST**

Any files created in Drive have the option to be edited by multiple people at once, meaning you'll be able to collaborate with others in real-time.