Distributed System and Cloud Computing Lab

Subject Code: MCAL32

A Practical Journal Submitted in Fulfilment

of the Degree of

MASTER

In

COMPUTER APPLICATION

Year 2023-2024

By

(RAVISHANKAR JAISWAL)

(Application ID: 129211)

(Seat Number: 5000058)

Semester- III

Under the Guidance of

Prof. Trupti Rongare

Institute of Distance and Open Learning

Vidya Nagari, Kalina, Santacruz East – 400098.

University of Mumbai

PCP Center

[Satish Pradhan Dyanasadhana College, Thane]

Institute of Distance and Open Learning,

Vidyanagari, Kalina, Santacruz (E) -400098

CERTIFICATE

This to certify that, **Ravishankar Jaiswal** appearing **master's in computer application (Semester 3) (Application ID: 129211)(Seat Number: 5000058):** has satisfactory completed the prescribed practical of **MCAL32 - Distributed System and Cloud Computing Lab** as laid down by the University of Mumbai for the academic year 2023-24

Teacher in charge	Examiners	Coordinator
Prof. Trupti Rongare		IDOL, MCA University of Mumbai

Date: -

Place: -

Contents

PRACTICAL 1

Aim: Write a program to develop multi-client server application where multiple clients chat with each other concurrently.

STEP 1: Create Server java program and build it.

MultithreadedSocketServer.java

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

public class MultithreadedSocketServer {
   public static void main(String[] args) {
      try {
         // Create server socket on port 8888
         ServerSocket server = new ServerSocket(8888);
         int counter = 0;
         System.out.println("Server Started ....");

         // Keep accepting client connections
         while (true) {
            counter++;
            Socket serverClient = server.accept(); // Accept client connection
            System.out.println(" >> Client No:" + counter + " started!");

            // Create a new thread to handle the client request
            ServerClientThread sct = new ServerClientThread(serverClient, counter);
            sct.start();
         }
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

STEP 2: Create a ServerClientThread Java program which is the part of Multithreaded Server
Socket program

ServerClientThread.java

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

class ServerClientThread extends Thread {
   private Socket serverClient;
   private int clientNo;

   // Constructor to initialize socket and client number
   ServerClientThread(Socket inSocket, int counter) {
      serverClient = inSocket;
      clientNo = counter;
   }
```

```
    public void run() {
        try {
            // Setup input and output streams
            DataInputStream inStream = new DataInputStream(serverClient.getInputStream());
            DataOutputStream outStream = new DataOutputStream(serverClient.getOutputStream());

            String clientMessage = "", serverMessage = "";
            int square;

            // Continuously listen for client messages
            while (!clientMessage.equals("bye")) {
                // Read the client's message
                clientMessage = inStream.readUTF();
                System.out.println("From Client-" + clientNo + ": Number is: " + clientMessage);

                // Calculate the square of the number
                square = Integer.parseInt(clientMessage) * Integer.parseInt(clientMessage);

                // Prepare the server's response message
                serverMessage = "From Server to Client-" + clientNo + " Square of " + clientMessage + " is
" + square;

                // Send the response back to the client
                outStream.writeUTF(serverMessage);
                outStream.flush();
            }

            // Close streams and socket
            inStream.close();
            outStream.close();
            serverClient.close();

        } catch (Exception ex) {
            System.out.println(ex);
        } finally {
            System.out.println("Client-" + clientNo + " exited!");
        }
    }
}
```

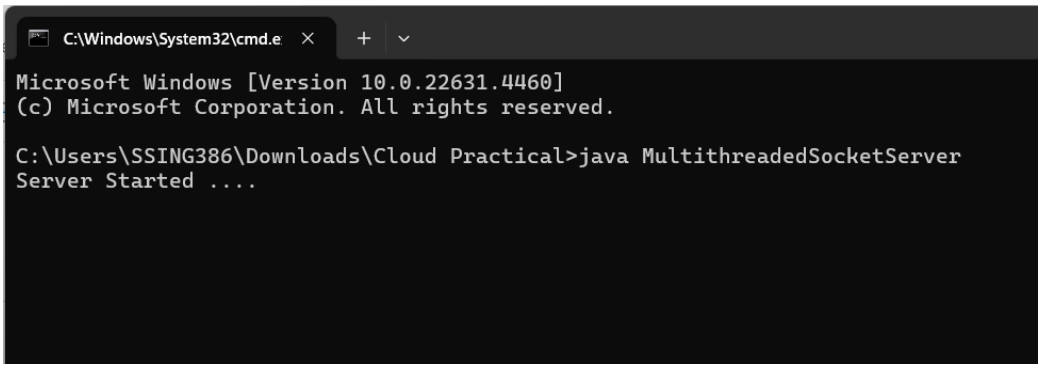
STEP 3: Now Create a Client program TCPClient.java and build it and run

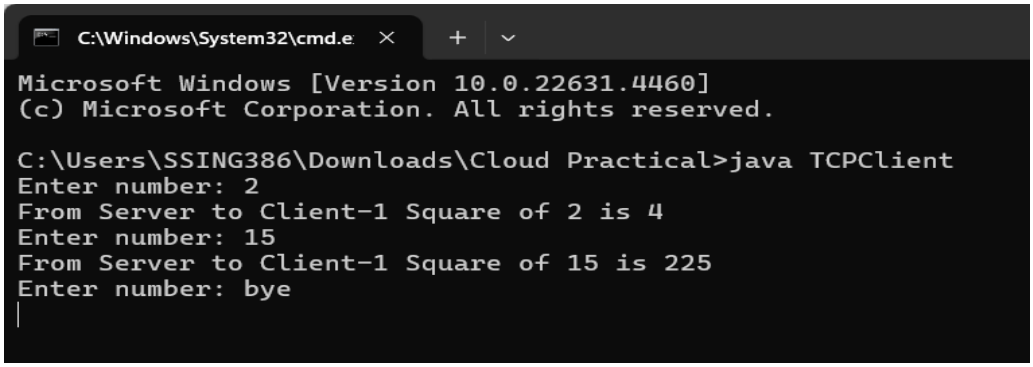
TCPClient.java

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

public class TCPClient {
    public static void main(String[] args) {
        try {
            // Create socket connection to server at localhost:8888
            Socket socket = new Socket("127.0.0.1", 8888);

            // Setup input and output streams
```

```
            DataInputStream inStream = new DataInputStream(socket.getInputStream());
            DataOutputStream outStream = new DataOutputStream(socket.getOutputStream());
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

            String clientMessage = "", serverMessage = "";

            // Keep sending messages until the client types "bye"
            while (!clientMessage.equals("bye")) {
                // Ask for user input (number)
                System.out.print("Enter number: ");
                clientMessage = br.readLine();

                // Send the message to the server
                outStream.writeUTF(clientMessage);
                outStream.flush();

                // Receive the server's response and print it
                serverMessage = inStream.readUTF();
                System.out.println(serverMessage);
            }

            // Close streams and socket
            outStream.close();
            inStream.close();
            socket.close();

        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

OUTPUT:





PRACTICAL 2

AIM: To implement a server calculator using RPC concept.

STEP 1: Create Server java program and build it.

RPCServer.java

```
import java.util.*;
import java.net.*;

class RPCServer {
   DatagramSocket ds;
   DatagramPacket dp;
   String str, methodName, result;
   int val1, val2;

   RPCServer() {
      try {
         ds = new DatagramSocket(1200);  // Server listens on port 1200
         byte b[] = new byte[4096];
         while (true) {
            dp = new DatagramPacket(b, b.length);
            ds.receive(dp);
            str = new String(dp.getData(), 0, dp.getLength());
            System.out.println("Received: " + str);  // Debug output
            if (str.equalsIgnoreCase("q")) {
               System.exit(1);
            } else {
               StringTokenizer st = new StringTokenizer(str, " ");
               methodName = st.nextToken();
               val1 = Integer.parseInt(st.nextToken());
               val2 = Integer.parseInt(st.nextToken());
            }

            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);
            } else {
               result = "Invalid method name!";
            }

            byte b1[] = result.getBytes();
            DatagramSocket ds1 = new DatagramSocket();
            DatagramPacket dp1 = new DatagramPacket(b1, b1.length, InetAddress.getLocalHost(),
1300);
            ds1.send(dp1);
```

```
            System.out.println("Sent result: " + result);  // Debug output
        }
    } 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) {
    if (val2 == 0) {
        return Integer.MAX_VALUE; // To avoid division by zero errors
    }
    return val1 / val2;
}

public static void main(String[] args) {
    new RPCServer();
}
}
```

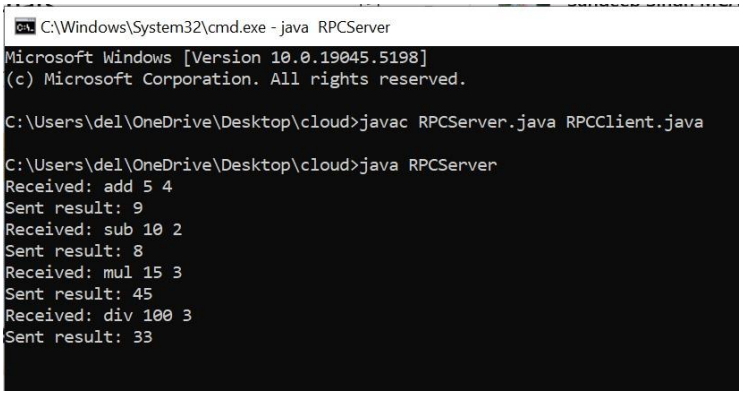
STEP 2: Now Create a Client program RPCClient.java and build it and run

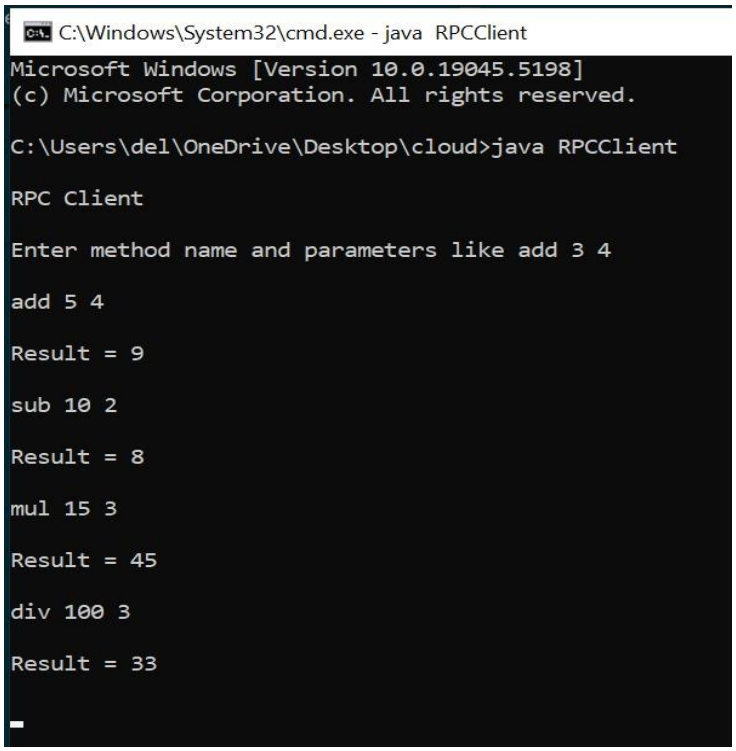
RPCClient.java

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

class RPCClient {
    RPCClient() {
        try {
            InetAddress ia = InetAddress.getLocalHost();
            DatagramSocket ds = new DatagramSocket();
            DatagramSocket ds1 = new DatagramSocket(1300);
            System.out.println("\nRPC Client\n");
            System.out.println("Enter method name and parameters like add 3 4\n");
            while (true) {
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                String str = br.readLine();
                byte b[] = str.getBytes();
                DatagramPacket dp = new DatagramPacket(b, b.length, ia, 1200);
                ds.send(dp);

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

```
            String result = new String(dp.getData(), 0, dp.getLength());
            System.out.println("\nResult = " + result + "\n");

            // Exit condition
            if (result.equals("q")) {
                System.out.println("Server exited");
                break;
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
  }

  public static void main(String[] args) {
    new RPCClient();
  }
}
```

OUTPUT:





AIM: The program demonstrates basic client-server communication in Java using TCP sockets, where the server sends the current date to the client upon connection.

Server Code (DateServer.java):

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

class DateServer {
    public static void main(String[] args) throws Exception {
        // Create a server socket that listens on port 5217
        ServerSocket serverSocket = new ServerSocket(5217);
        System.out.println("Server is waiting for connection...");

        // Continuously accept client connections
        while (true) {
            // Accept a client connection
            Socket clientSocket = serverSocket.accept();

            // Create an output stream to send the current date to the client
            DataOutputStream out = new DataOutputStream(clientSocket.getOutputStream());
            out.writeBytes("Server Date: " + (new Date()).toString() + "\n");

            // Close the output stream and socket
            out.close();
            clientSocket.close();
        }
    }
}
```

Client Code (DateClient.java):

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

class DateClient {
    public static void main(String[] args) throws Exception {
        // Create a socket to connect to the server on localhost at port 5217
        Socket socket = new Socket(InetAddress.getLocalHost(), 5217);

        // Create an input stream to receive data from the server
        BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

        // Read and print the server's response (date)
        System.out.println(in.readLine());

        // Close the input stream and socket
        in.close();
        socket.close();
    }
}
```

OUTPUT:

```
C:\Users\del\OneDrive\Desktop\cloud>java DateServer
Server is waiting for connection...
Connection Successful...
```

```
Microsoft Windows [Version 10.0.19045.5198]
(c) Microsoft Corporation. All rights reserved.

C:\Users\del\OneDrive\Desktop\cloud>java DateClient
Server Date: Fri Nov 29 12:00:06 IST 2024

C:\Users\del\OneDrive\Desktop\cloud>
```

PRACTICAL 4

AIM: Demonstrate an sample RMI Java application.

STEP 1: Create the Remote Interface (Hello.java)
This interface defines the remote method that the server will implement, and the client will invoke.

Hello.java

```
import java.rmi.*;
public interface Hello extends Remote
{
 void printMsg() throws RemoteException;
}
```

STEP 2: Implement the Remote Interface (ImplExample.java)
This is the implementation class where the remote method printMsg is implemented. The server uses this class to provide the required functionality.

ImplExample.java

```
public class ImplExample implements Hello
{

 // Implementing the interface method
 public void printMsg()
 {
 System.out.println("This is an example RMI program....");
 }
}
```

STEP 3: Create the Server Code (Server.java)
The server registers the remote object with the RMI registry, so that it can be looked up by the client.

Server.java

```
import java.rmi.registry.*;
import java.rmi.*;
import java.rmi.server.*;
public class Server extends ImplExample {
 public Server() {}
 public static void main(String args[]) {
 try {

 ImplExample obj = new ImplExample();
 Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
 Registry registry = LocateRegistry.getRegistry();
 registry.bind("Hello", stub);
 System.err.println("Server ready");
 } catch (Exception e) {
 System.err.println("Server exception: " + e.toString());
```

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

STEP 4: Create the Client Code (Client.java)
The client looks up the remote object in the RMI registry and invokes the remote method.

Client.java

```
import java.rmi.registry.*;
public class Client {
 private Client() {}
 public static void main(String[] args) {
 try {
 // Getting the registry
 Registry registry = LocateRegistry.getRegistry(null);

 // Looking up the registry for the remote object
 Hello stub = (Hello) registry.lookup("Hello");
 stub.printMsg();
 System.out.println("Hello is running with RMI");

 // System.out.println("Remote method invoked");
 } catch (Exception e) {
          System.err.println("Client exception: " + e.toString());
 e.printStackTrace();
 }
 }
}
```

OUTPUT:

```
C:\Users\del\OneDrive\Desktop\cloud\RMI\cloud>javac *.java

C:\Users\del\OneDrive\Desktop\cloud\RMI\cloud>rmiregistry
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release
```

```
C:\Users\del\OneDrive\Desktop\cloud\RMI\cloud>java Server
Server ready
This is an example RMI program....
```

```
C:\Users\del\OneDrive\Desktop\cloud\RMI\cloud>java Client
Hello is running with RMI

C:\Users\del\OneDrive\Desktop\cloud\RMI\cloud>_
```

PRACTICAL 5

Aim: Book information from library database using remote object communication concept.

Code:
• Library.java

```
import java.io.Serializable;

public class Library implements Serializable {

        private int BookID;     private String BookName;
                private String BookAuthor;

    public int getBookID () {
            return BookID ;
                }

                public void setBookID(int bookID) {
                        BookID = bookID;
                }

                public String getBookName() {
                        return BookName;
                }

                public void setBookName(String bookName) {
                        BookName = bookName;
                }

 public String getBookAuthor() {   return BookAuthor;
                }

                public void setBookAuthor(String bookAuthor) {
                        BookAuthor = bookAuthor;
                }
}
```

Hello.java

```
import java.rmi.Remote;
import java.util. * ;

public interface Hello extends Remote {
  public List < Library > getBookInfo() throws Exception;
}
```

ImplExample.java import java.sql.*;

```
import java.util.*;

public class ImplExample implements Hello {

                @Override
```

```
public List<Library> getBookInfo() throws Exception {

        List<Library> list = new ArrayList<>();
        String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
        String DB_URL = "jdbc:mysql://localhost:3306/test";
        String USER = "DATABASE_USERNAME";
        String PASS = " DATABASE_PASSWORD ";
        Connection conn = null;
        Statement stmt = null;
        Class.forName("com.mysql.cj.jdbc.Driver");
    System.out.println("Connecting to a selected database...");
conn = DriverManager.getConnection(DB_URL, USER, PASS);
    System.out.println("Connected database successfully...");
System.out.println("Creating statement...");        stmt = conn.createStatement();
    String sql = "SELECT * FROM books";
        ResultSet rs = stmt.executeQuery(sql);

        while (rs.next()) {
        int id = rs.getInt("Book_id");                  String name =
rs.getString("Book_name");
        String author = rs.getString("Book_author");
Library info = new Library();                   info.setBookID(id);
info.setBookName(name);                         info.setBookAuthor(author);
list.add(info);
        }
        rs.close();
        return list;
    }

}
```

• Server.java import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry; import java.rmi.server.UnicastRemoteObject;

```
public class Server extends ImplExample {
    public Server() {
    }

    public static void main(String args[]) {
        try {
            ImplExample obj = new ImplExample();
        Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
        Registry registry = LocateRegistry.createRegistry(6666);
        registry.rebind("bookInfo", stub);
System.err.println("Server ready");
        }
        catch (Exception e) {
        System.err.println("Server exception: " + e.toString());
e.printStackTrace();
        }
    }
}
```

• Client.java import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry; import java.util.List;

```
public class Client {
		private Client() {
		}

		public static void main(String[] args) throws Exception {
			try {
				Registry registry = LocateRegistry.getRegistry("localhost",
6666);
				Hello stub = (Hello) registry.lookup("bookInfo");

				List<Library> list = (List) stub.getBookInfo();
				for (Library l : list) {
					System.out.println("Book ID: " + l.getBookID());
					System.out.println("Book Name: " + l.getBookName());
					System.out.println("Book Author: " + l.getBookAuthor());
					System.out.println("----------------------------------------------
");
				}

			} catch (Exception e) {
			System.err.println("Client exception: " + e.toString());
		e.printStackTrace();
			}
		}
}
```

Output:

```
Server [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (Oct 7, 2024, 1:12:55 AM) [pid: 7176]
Server ready
Connecting to a selected database...
Connected database successfully...
Creating statement...
<terminated> Client [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (Oct 7, 2024, 1:13:01 AM – 1:13:04 AM) [pid: 12116]
Book ID: 1001
Book Name: Atomic Habits
Book Author: James Clear
----------------------------------------------
Book ID: 1002
Book Name: Life's Amazing Secrets
Book Author: Gaur Gopal Das
----------------------------------------------
Book ID: 1003
Book Name: Rich Dad Poor Dad
Book Author: Robert Kiyosaki, Sharon Lechter
----------------------------------------------
Book ID: 1004
Book Name: Atomic Habits
Book Author: James Clear
----------------------------------------------
```

Book_id	Book_name	Book_author
1001	Atomic Habits	James Clear
1002	Life's Amazing Secrets	Gaur Gopal Das
1003	Rich Dad Poor Dad	Robert Kiyosaki, Sharon Lechter
1004	Atomic Habits	James Clear
NULL	NULL	NULL

PRACTICAL 6

Aim: Electric bill database using remote object communication concept.

Code:

• Electric.java

```
public class  Electric implements java.io.Serializable {

        private float BillAmount;        private String CustomerName;
                private String BillDueDate;

                public float getBillAmount() {
                        return BillAmount;
                }

                public void setBillAmount(float billAmount) {
                        BillAmount = billAmount;
                }

 public String getCustomerName() {   return CustomerName;
                }

                public void setCustomerName(String customerName) {
                        CustomerName = customerName;
                }

 public String getBillDueDate() {   return BillDueDate;
                }

                public void setBillDueDate(String billDueDate) {
                        BillDueDate = billDueDate;
                }
}

Hello.java import java.rmi.Remote;
import java.util.List;

public interface Hello extends Remote {
        public List<Electric> getBillInfo() throws Exception; }

ImplExample.java import java.sql.*; import java.util.*;

public class ImplExample implements Hello {

        public List<Electric> getBillInfo() throws Exception {                    List<Electric> list
= new ArrayList<Electric>();

                String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";                    String
DB_URL = "jdbc:mysql://localhost:3306/test";

                        String USER = " DATABASE_USERNAME";
                        String PASS = " DATABASE_PASSWORD ";

                        Connection conn = null;
                        Statement stmt = null;

                        Class.forName("com.mysql.cj.jdbc.Driver");
                System.out.println("Connecting to a selected database...");
        conn = DriverManager.getConnection(DB_URL, USER, PASS);
                        System.out.println("Connected database successfully...");
```

```
                        System.out.println("Creating statement...");

                stmt = conn.createStatement();                  String sql = "SELECT * FROM
electricbill";
                        ResultSet rs = stmt.executeQuery(sql);

                while (rs.next()) {                             float amount = rs.getFloat("BillAmount");
                        String name = rs.getString("ConsumerName");
                        String Date = rs.getString("BillDuedate");                      Electric
info = new Electric();                  info.setBillAmount(amount);
        info.setCustomerName(name);                             info.setBillDueDate(Date);
                                list.add(info);
                        }
                        rs.close();
                        return list;
                }
        }
```

• Server.java import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry; import java.rmi.server.UnicastRemoteObject;

```
public class Server extends ImplExample {
                public Server() {
                }

                public static void main(String args[]) {
                        try {
                                ImplExample obj = new ImplExample();
                        Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
                Registry registry = LocateRegistry.createRegistry(6666);
        registry.rebind("billinfo", stub);                      System.err.println("Server
ready");
                        } catch (Exception e) {
                        System.err.println("Server exception: " + e.toString());
        e.printStackTrace();
                        }
                }
}
```

• Client.java import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry; import java.util.List;

```
public class Client {
                private Client() {
                }

                public static void main(String[] args) throws Exception {
                        try {
                                Registry registry = LocateRegistry.getRegistry("localhost",
6666);
                        Hello stub = (Hello) registry.lookup("billinfo");
        List<Electric> list = (List) stub.getBillInfo();
                                for (Electric l : list) {
```

```
                                        System.out.println("Customer name: " +
l.getCustomerName());
                                        System.out.println("Bill Due Date: " +
l.getBillDueDate());
                                        System.out.println("Bill Amount: " + l.getBillAmount());
                                        System.out.println("---------------------------------------------
");

                                }
                        } catch (Exception e) {
                        System.err.println("Client exception: " + e.toString());
        e.printStackTrace();
                        }
                }
}
```

Output:

```
Server (1) [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (Oct 7, 2024, 1:33:59 AM) [pid: 12004]
Server ready
Connecting to a selected database...
Connected database successfully...
Creating statement...
<terminated> Client (1) [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (Oct 7, 2024, 1:34:03 AM – 1:34:05 AM) [pid: 16020]
Customer name: James Clear
Bill Due Date: 07-Oct-2024
Bill Amount: 302.56
---------------------------------------------
Customer name: Robert Kiyosaki
Bill Due Date: 08-Oct-2024
Bill Amount: 1000.6
---------------------------------------------
Customer name: Sharon Lechter
Bill Due Date: 09-Oct-2024
Bill Amount: 658.0
---------------------------------------------
```

	ConsumerName	BillDuedate	BillAmount
▶	James Clear	07-Oct-2024	302.56
	Robert Kiyosaki	08-Oct-2024	1000.6
	Sharon Lechter	09-Oct-2024	658

PRACTICAL 7

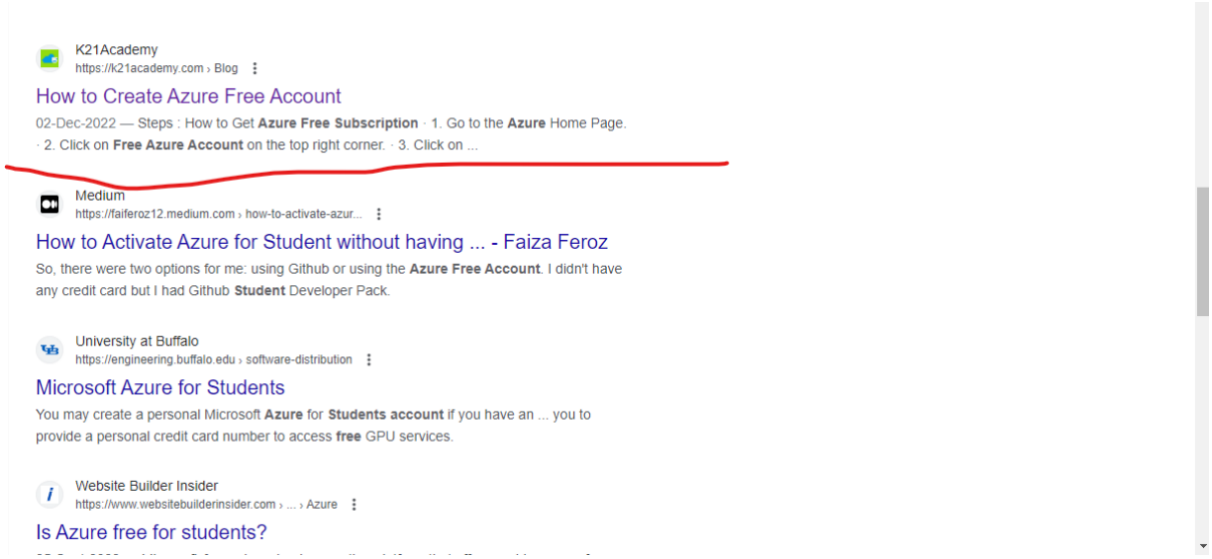
Aim: implementation of cloud computing services.

Implementation of cloud computing services:

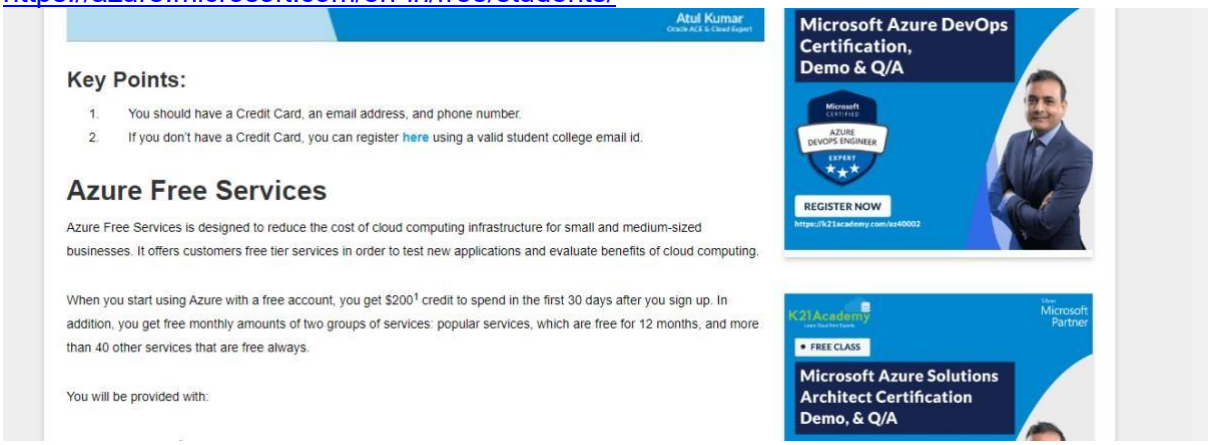
Infrastructure as a service (IaaS):

• Create a storage account:

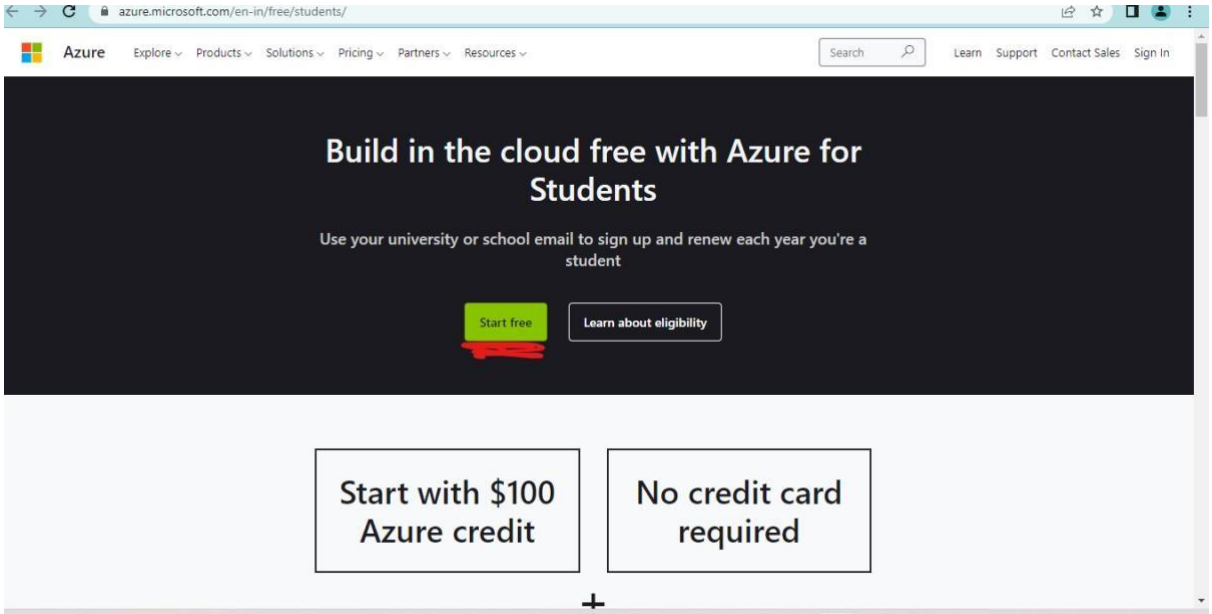
A storage account is an Azure Resource Manager resource. Resource Manager is the deployment and management service for Azure. For more information, see Azure Resource Manager overview. https://k21academy.com/microsoft-azure/create-free-microsoft-azure-trial-account/



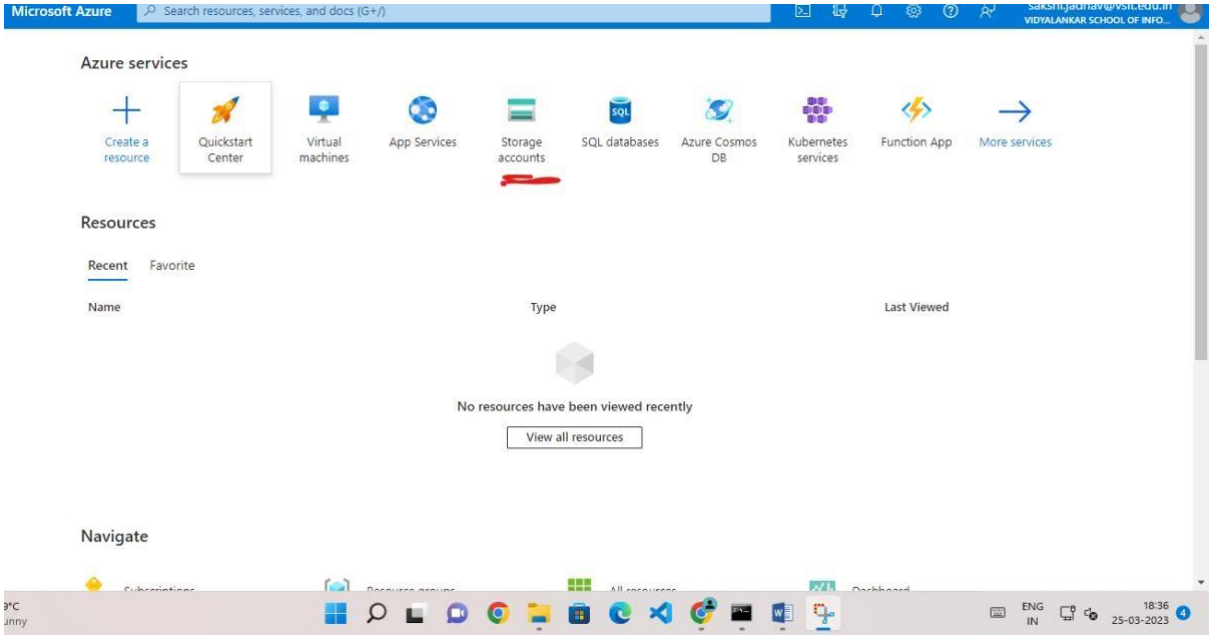
https://azure.microsoft.com/en-in/free/students/



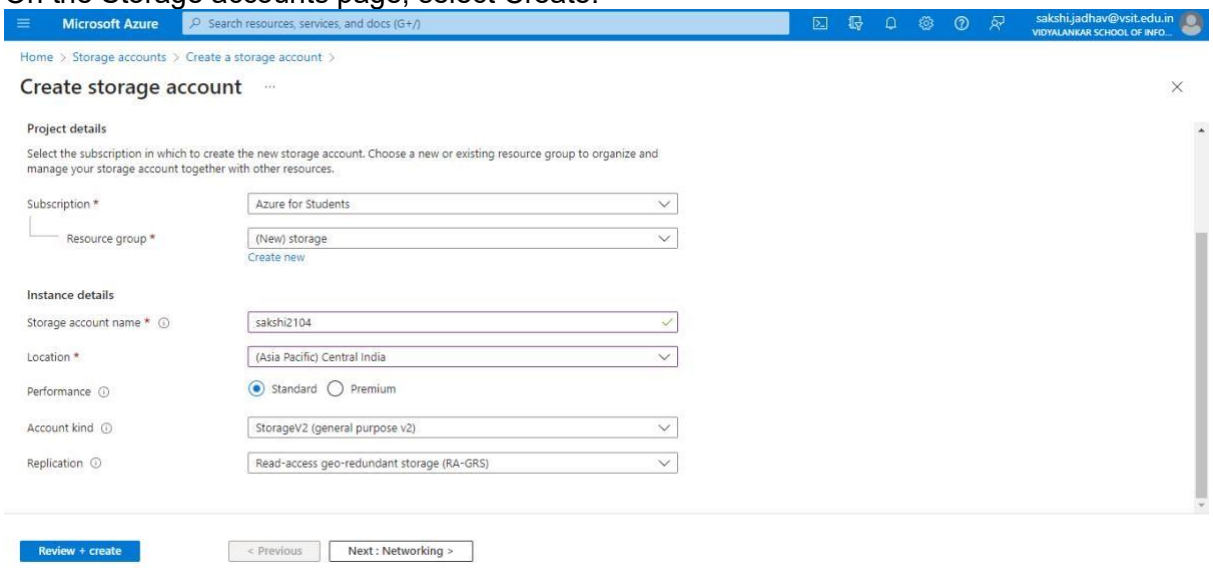
Click on Start Free.

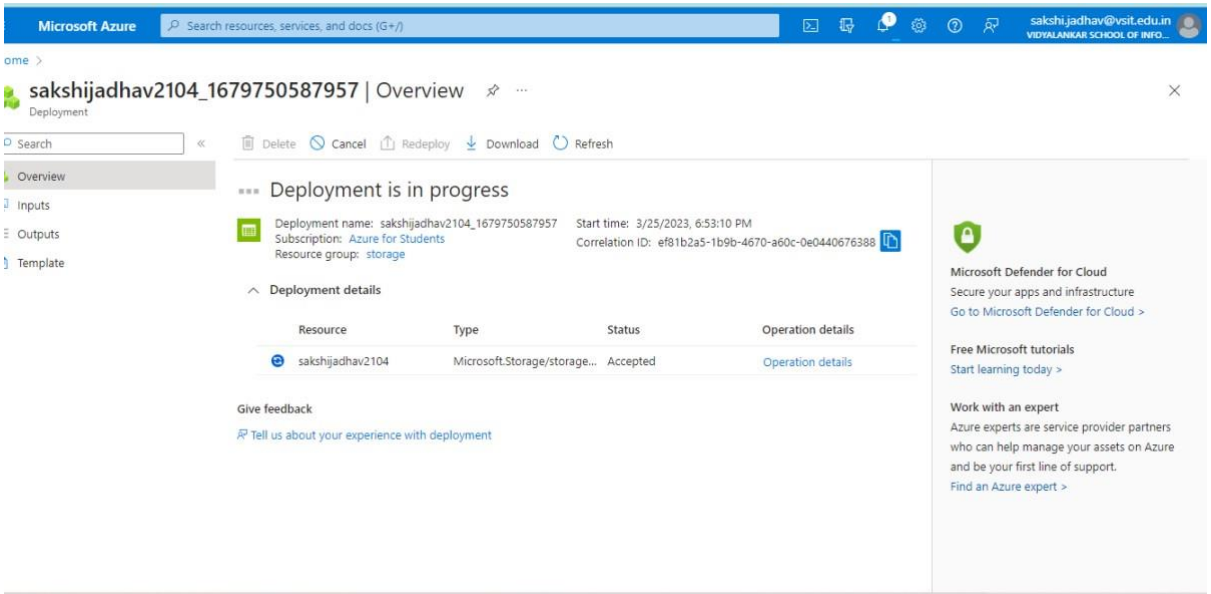


From the left portal menu, select Storage accounts to display a list of your storage accounts. If the portal menu isn't visible, click the menu button to toggle it on.

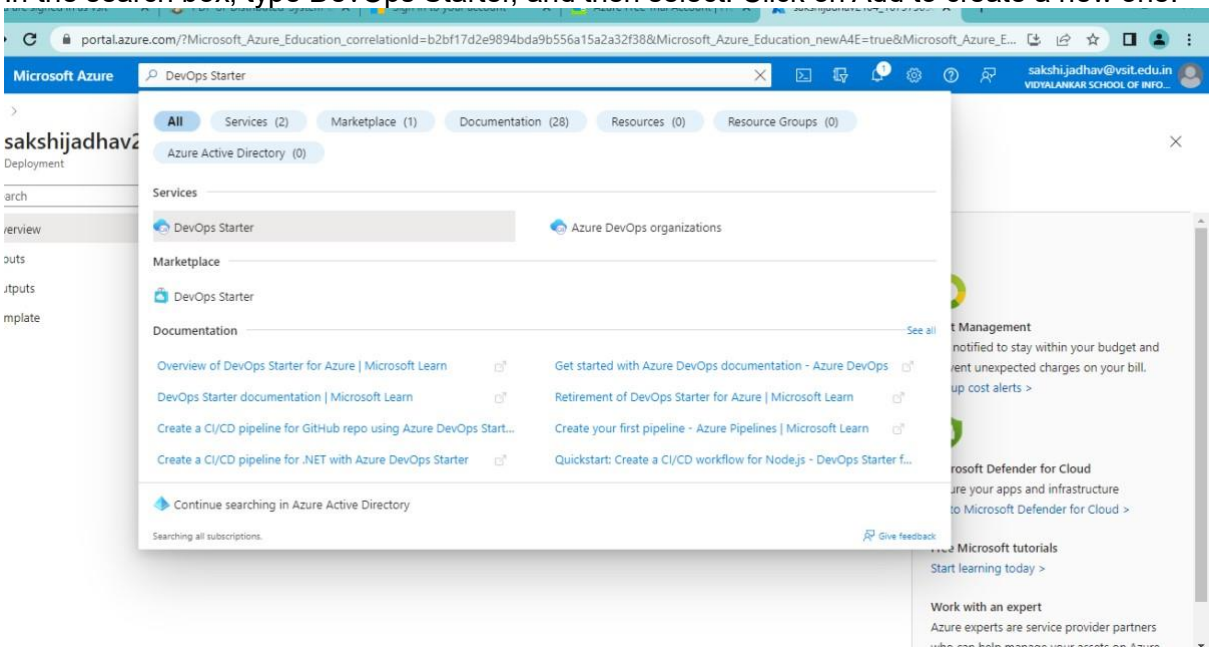


On the Storage accounts page, select Create.

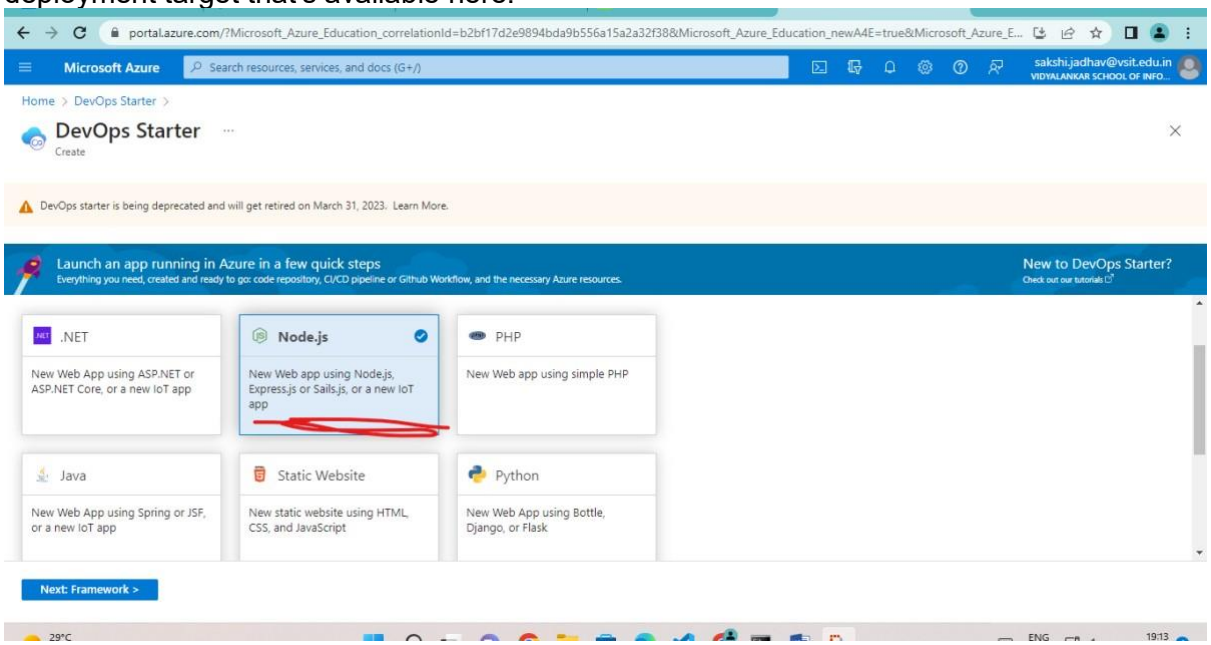


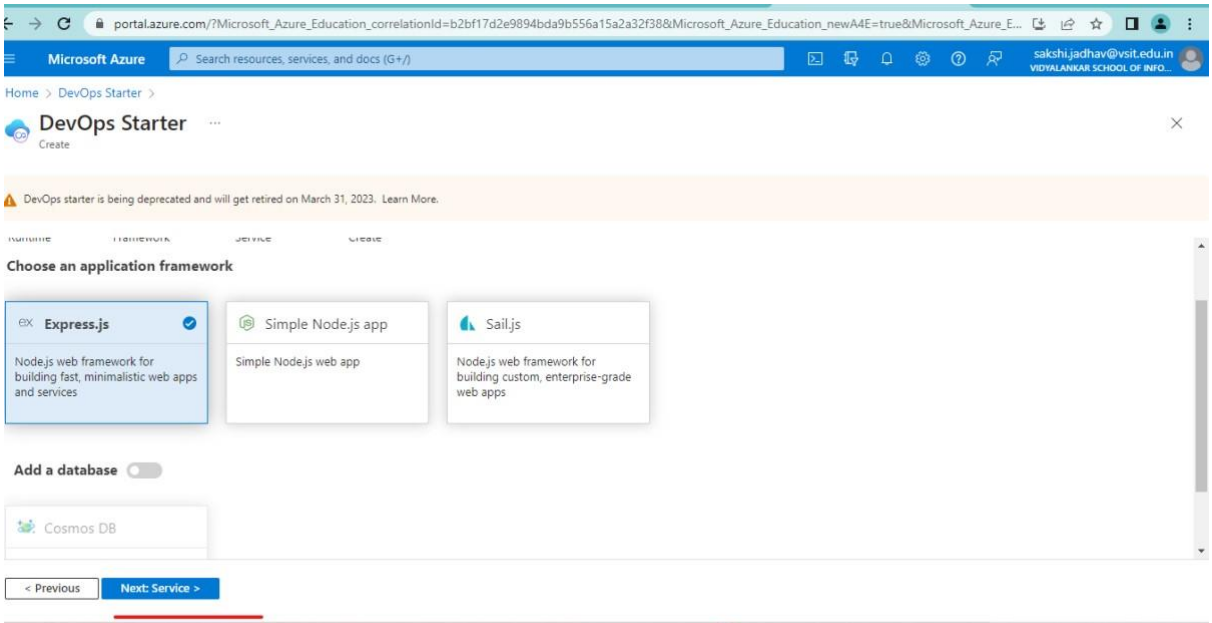


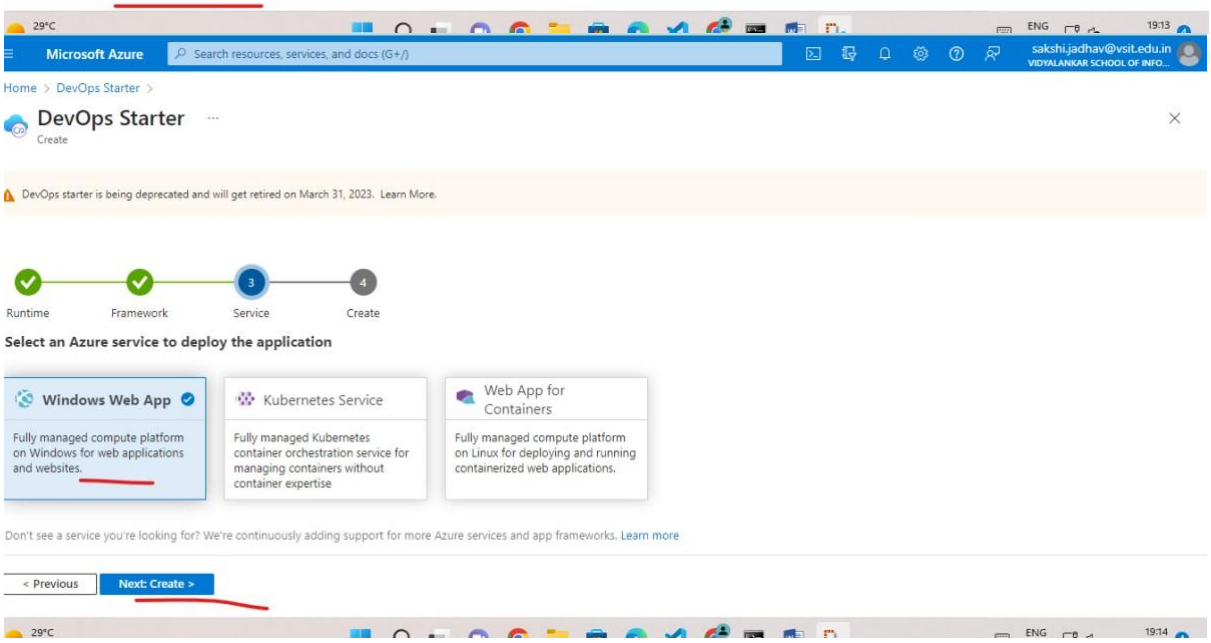
• Use DevOps Starter to deploy a Node.js app
DevOps Starter creates a workflow in GitHub. You can use an existing GitHub organization. DevOps Starter also creates Azure resources such as Web App in the Azure subscription of your choice.
Sign in to the Azure portal.
In the search box, type DevOps Starter, and then select. Click on Add to create a new one.



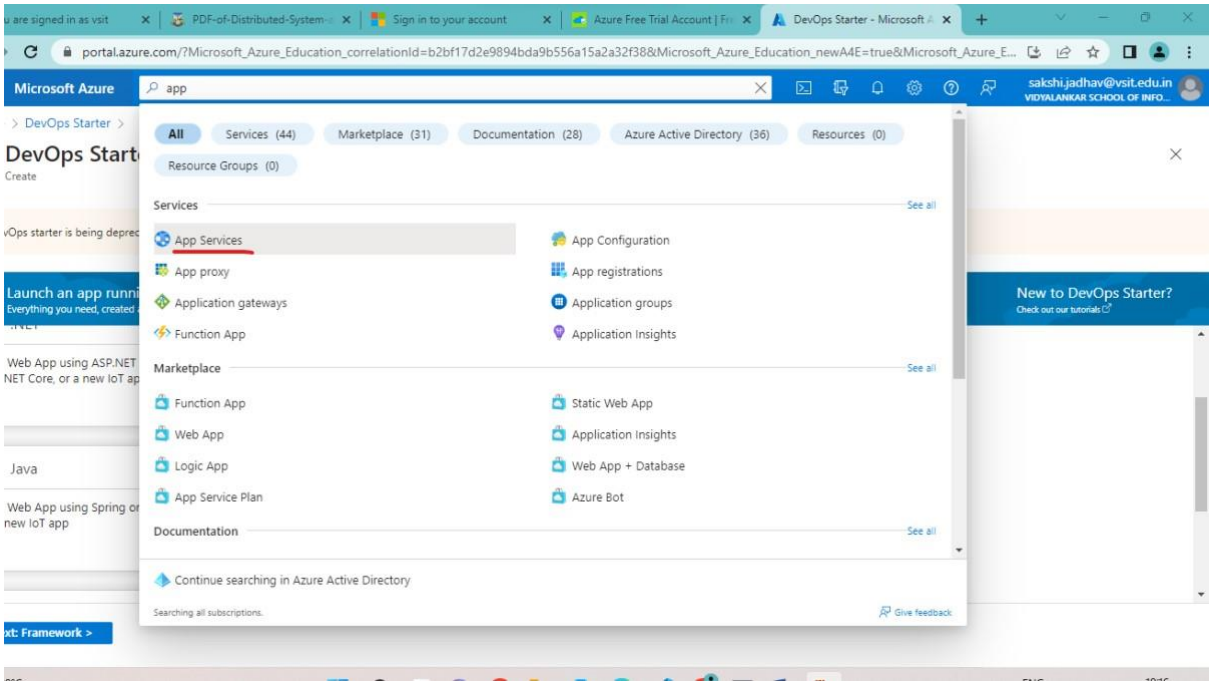
Select Node.js, and then select Next.
Under Choose an application Framework, select Express.js, and then select Next. The application framework, which you chose in a previous step, dictates the type of Azure service deployment target that's available here.



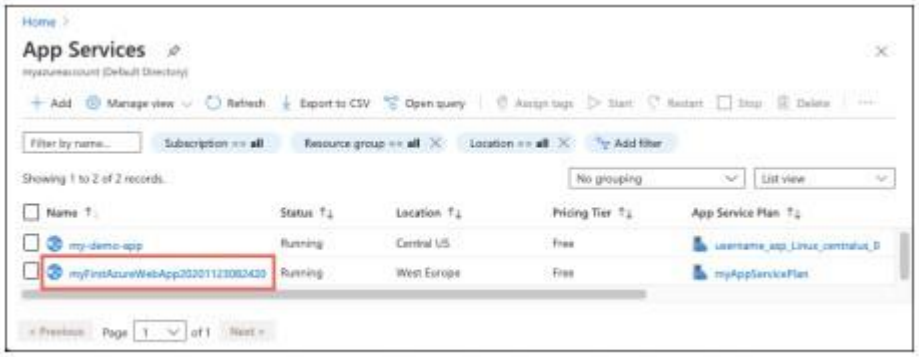




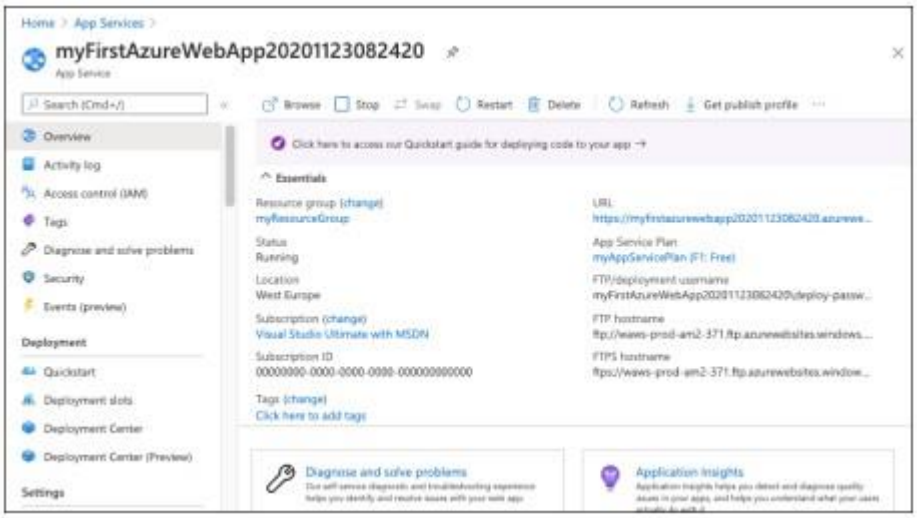
<u>Software as a service (SaaS):</u>
Software as a service (SaaS) allows users to connect to and use cloud based apps over the Internet. Common examples are email, calendaring and office tools (such as Microsoft Office 365).
Manage the Azure app:
To manage your web app, go to the Azure portal, and search for and select App Services.



On the App Services page, select the name of your web app.



The Overview page for your web app, contains options for basic management like browse, stop, start, restart, and delete. The left menu provides further pages for configuring your app.



Practical 8

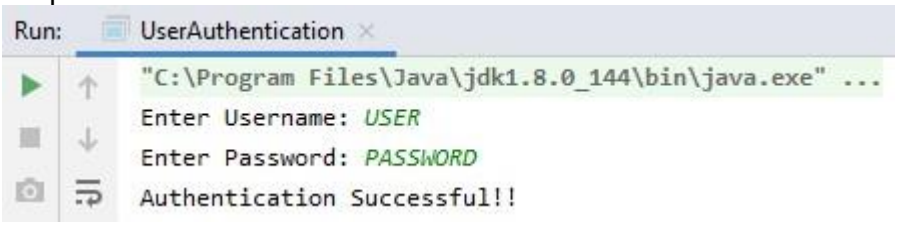
Aim: Java program to illustrate how user authentication is done.

Code:

• UserAuthentication.java

```
import java.util.Scanner;

public class UserAuthentication {     public static void main(String[] args) {      String
userName, password;
    Scanner sc = new Scanner(System.in);        System.out.print("Enter Username: ");
userName = sc.nextLine();
        System.out.print("Enter Password: ");
        password = sc.nextLine();
        if (userName.equals("USER") && password.equals("PASSWORD"))
            System.out.println("Authentication Successful!!");        else
            System.out.println("Authentication Failed!!");
    }
}
```

Output:



Practical 9

Aim: Java program to check the authentication of the user.

Code:

```
AuthenticationOfUser.java public class AuthenticationOfUser {    public static void
main(String[] args) {        String userName = "Uday Bhai";        String password = "Aalu
Kanda";
        if(userName.equals("Uday Bhai") && password.equals("Aalu Kanda"))
            System.out.println("Authentication Successful!!");        else
            System.out.println("Username/Password not matching");
    }
}
```

Output:

```
Run:    AuthenticationOfUser
        "C:\Program Files\Java\jdk1.8.0_144\bin\java.exe" ...
        Authentication Successful!!
```

```
AuthenticationOfUser.java public class AuthenticationOfUser {    public static void
main(String[] args) {        String userName = "Uday Bhai";        String password = "Alu
Kanda";
        if(userName.equals("Uday Bhai") && password.equals("Aalu Kanda"))
            System.out.println("Authentication Successful!!");        else
            System.out.println("Username/Password not matching");
    }
}
```

Output:

```
Run:    AuthenticationOfUser
        "C:\Program Files\Java\jdk1.8.0_144\bin\java.exe" ...
        Username/Password not matching
```

PRACTICAL 10

Aim: Create a login form in java.

Code:

LoginForm.java

```
import javax.swing.*; import java.awt.*; import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class LoginForm extends JFrame implements ActionListener {

   JButton b1;
   JPanel newPanel;
   JLabel userLabel, passLabel;     final JTextField textField1, textField2;

   LoginForm()
   {
      userLabel = new JLabel();        userLabel.setText("Username");        textField1 = new
JTextField(15);        passLabel = new JLabel();        passLabel.setText("Password");
textField2 = new JPasswordField(15);        b1 = new JButton("SUBMIT");        newPanel =
new JPanel(new GridLayout(3, 1));        newPanel.add(userLabel);
newPanel.add(textField1);        newPanel.add(passLabel);        newPanel.add(textField2);
newPanel.add(b1);
      add(newPanel, BorderLayout.CENTER);
      b1.addActionListener(this);
      setTitle("LOGIN FORM");
   }
   public void actionPerformed(ActionEvent ae)
   {
      String userValue = textField1.getText();        String passValue = textField2.getText();
      if (userValue.equals("test1@gmail.com") && passValue.equals("test"))
      {
         NewPage page = new NewPage();
         page.setVisible(true);
         JLabel wel_label = new JLabel("Welcome: "+userValue);
page.getContentPane().add(wel_label);
      }        else{
      System.out.println("Please enter valid username and password");
      }
   }
}
class LoginFormDemo
{
   public static void main(String arg[])
   {        try
      {
         LoginForm form = new LoginForm();
         form.setSize(300,100);
         form.setVisible(true);
      }
      catch(Exception e)
      {
         JOptionPane.showMessageDialog(null, e.getMessage());
      }
   }
}
```

• NewPage.java

```
import javax.swing.*;

public class NewPage extends JFrame {     NewPage() {
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
setTitle("Welcome");
        setSize(400, 200);
    } }
```

Output: