Cloud Computing Practicals

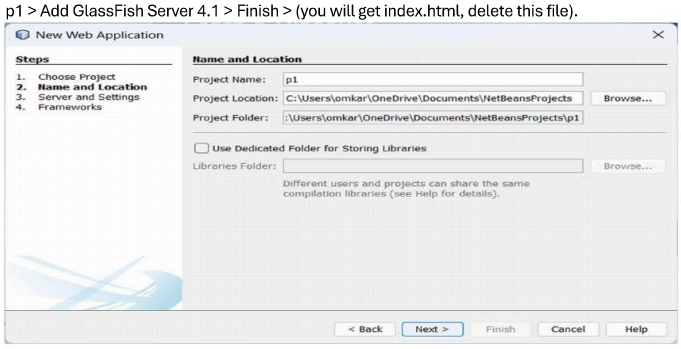
Name: Omkar Sawardekar **Roll no.:** 22109

Practical 1 - Create a Simple SOAP service

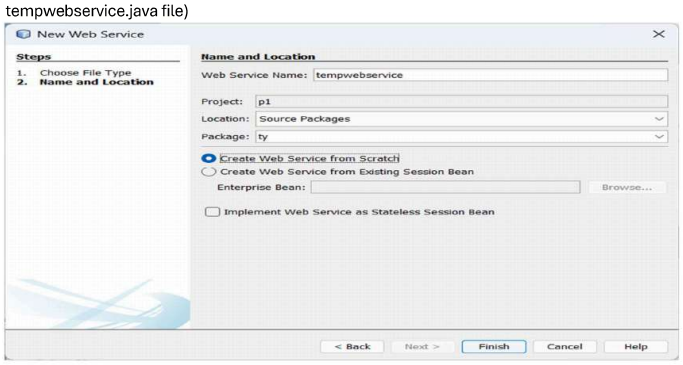
Aim: To create a Simple Web Service that converts the temperature from Celsius to Fahrenheit and vice versa.

Steps:

Step 1: Create a Web Application: File > New Project > Java Web > Web Application > Name: p1 > Add GlassFish Server 4.1 > Finish > (you will get index.html, delete this file).



Step 2: Create a Web Service in that project: Right Click on Project Name > New > Web Service... > Name: tempwebservice and Package Name: ty > OK > (you will get tempwebservice.java file)



Step 3: As we have created our WEB SERVICE (i.e. tempwebservice), WE'LL NOW ADDSOME FUNCTIONALITIES (i.e. temperature conversion methods). So, in the class tempwebservice, add 2 methods by doing:

a) Right Click > Insert Code > Add Web Service Operation...>

Name: convertFtoC >

Return Type: float >

Add a Parameter 'a' with Type: float > OK >

Now add formula in return statement((a-32) *5/9) .

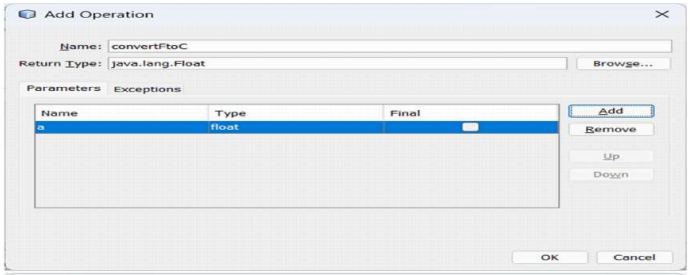
b) Right Click > Insert Code > Add Web Service Operation...>

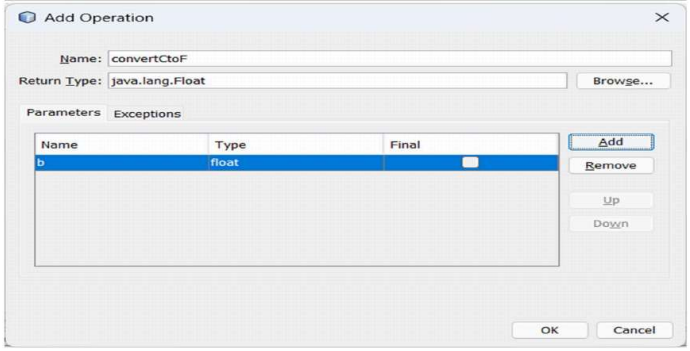
Name: convertCtoF >

Return Type: float >

Add a Parameter 'b' with Type: float > OK >

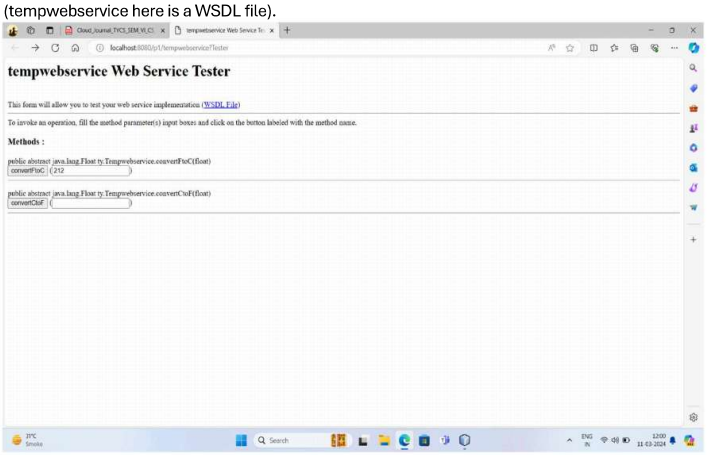
Now add formula in return statement((b*9/5) +32)

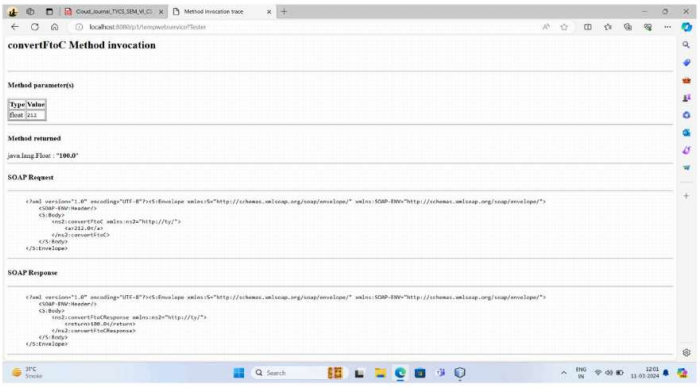




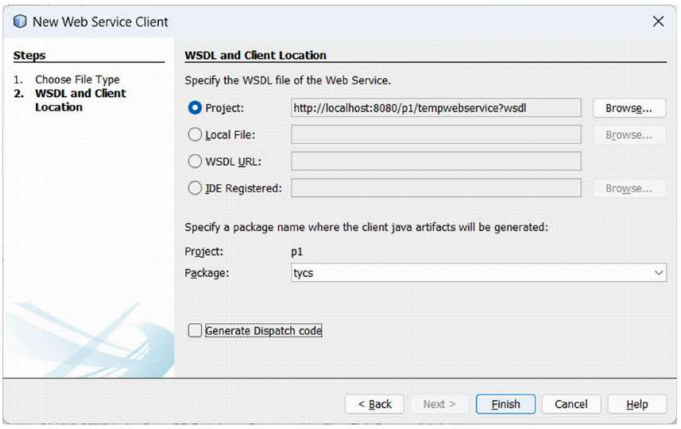
Step 4: Compile(F9) and Deploy (right click on Project Name > Deploy).

Step 5: Now we will test our web service. So, navigate to: Projects > Project Name > Web Services > right click on tempwebservice > click Test Web Service > (the browser will open). (tempwebservice here is a WSDL file).

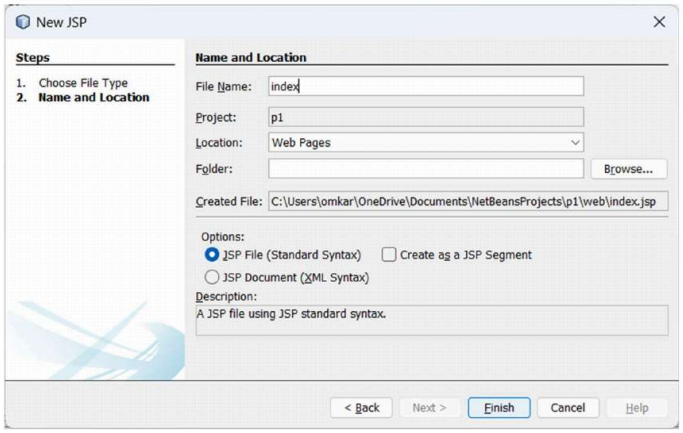




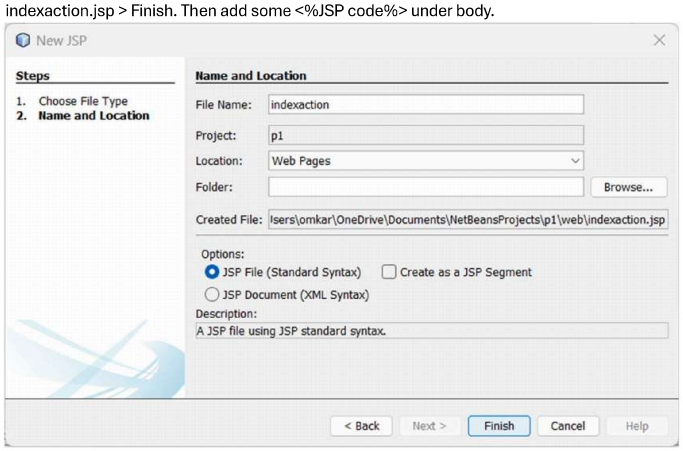
Step 6: Now we will be using the created Web Service (i.e. tempwebservice) by creating our WEB SERVICE CLIENT. So, Right Click on Project Name > New > Other > Web Services > Web Service Client... > Browse your Project Name and select WSDL file > Add Package Name: tycs > Finish.



Step 7: Now to make front page, we need to create a JSP file. So, Right Click on Project Name > New > JSP > File Name: index > Finish. Now from the palette, add HTML Form (Action: indexaction.jsp, Type: text, Method: Post) Inside Form tag, add Text Input (Name: txt) Inside Form tag, add Button (Label: Convert, Type: submit, Name: convert).



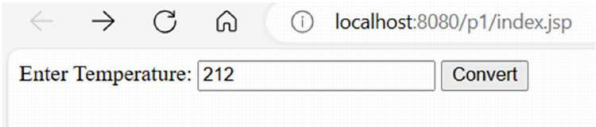
Step 8: Now to create the action behind that, we will create another JSP file, with a name that is declared in the previous JSP file. So, Right Click on Project Name > New > JSP > File Name: indexaction.jsp > Finish. Then add some <%JSP code%> under body.

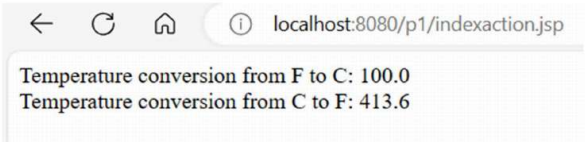


Step 9: Now RUN “index.jsp” file.

Step 10: Finish!

Outputs:



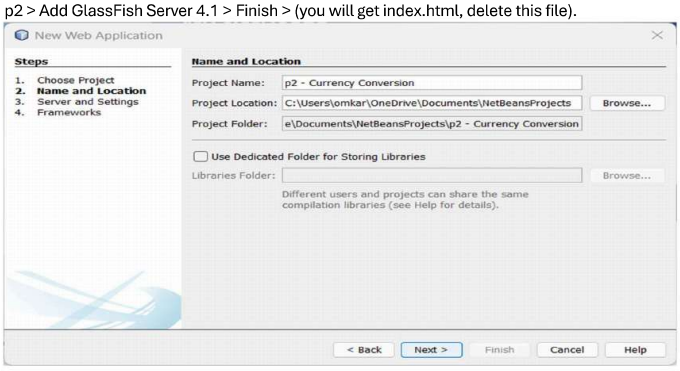


Practical 2 - Create a Simple SOAP service:

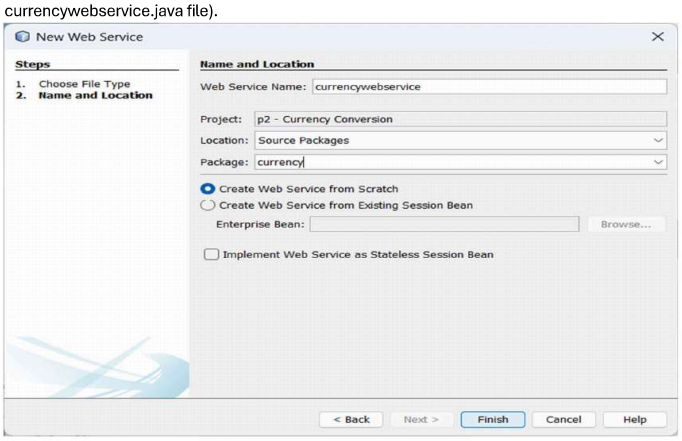
Aim: Define a simple web service like converting Rs into Dollar and call it from different platform like JAVA and .NET.

Steps:

Step 1: Create a Web Application: File > New Project > Java Web > Web Application > Name: p2 > Add GlassFish Server 4.1 > Finish > (you will get index.html, delete this file).



Step 2: Create a Web Service in that project: Right Click on Project Name > New > Web Service... > Name: currencywebservice and Package Name: currency > OK > (you will get currencywebservice.java file).



Step 3: As we have created our WEB SERVICE (i.e. currencywebservice), WE'LL NOW ADDSOME FUNCTIONALITIES (i.e. temperature conversion methods). So, in the class currrencywebservice, add 2 methods by doing:

a) Right Click > Insert Code > Add Web Service Operation...>

Name: INRtoDOLL >

Return Type: string >

Add a Parameter 'INR' with Type: double > OK >

Now add formula in return statement: (INR + " rupees in dollar are " + (INR/83.11)) .

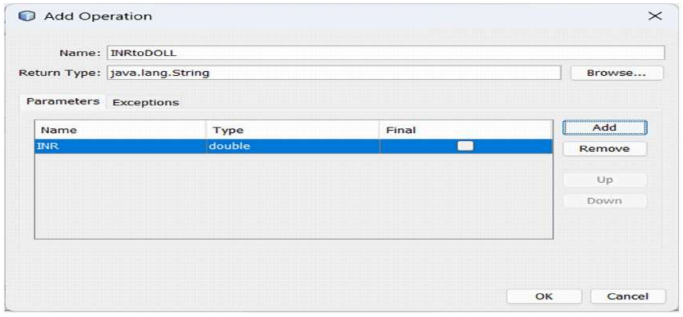
b) Right Click > Insert Code > Add Web Service Operation...>

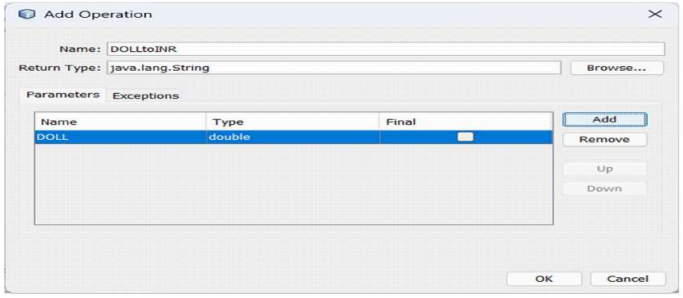
Name: DOLLtoINR >

Return Type: string >

Add a Parameter 'DOLL' with Type: double > OK >

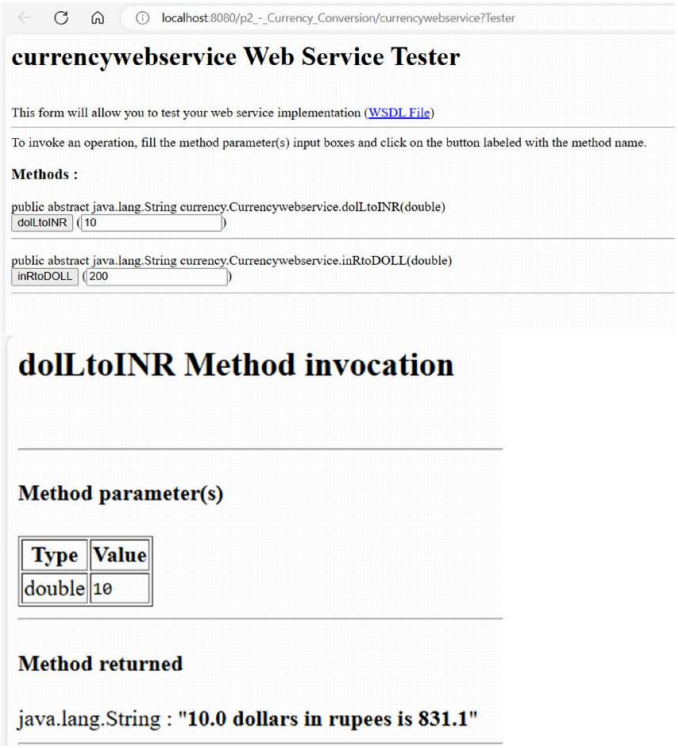
Now add formula in return statement: (DOLL + " dollars in rupees are " + (DOLL*83.11)).



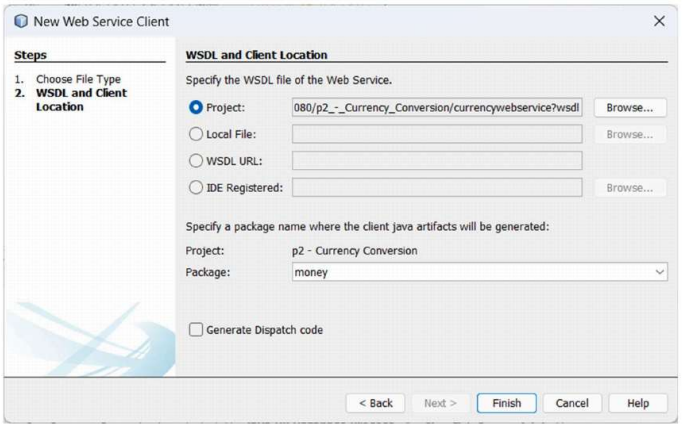


Step 4: Compile(F9) and Deploy (right click on Project Name > Deploy).

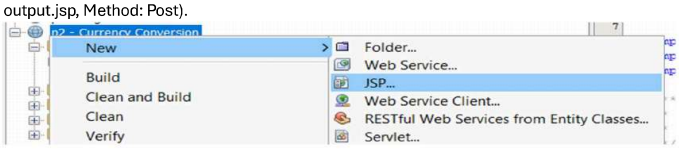
Step 5: Now we will test our web service. So, navigate to: Projects > Project Name > Web Services > right click on currencywebservice > click Test Web Service > (the browser will open). (currencywebservice here is a WSDL file).

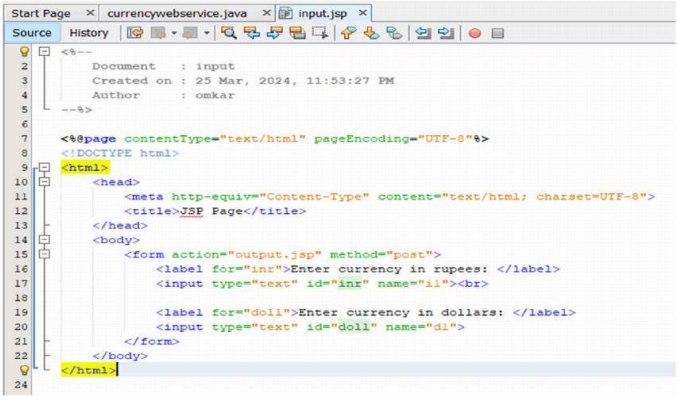


Step 6: Now we will be using the created Web Service (i.e. currencywebservice) by creating our WEB SERVICE CLIENT. So, Right Click on Project Name > New > Other > Web Services > Web Service Client... > Browse your Project Name and select WSDL file > Add Package Name: money > Finish.

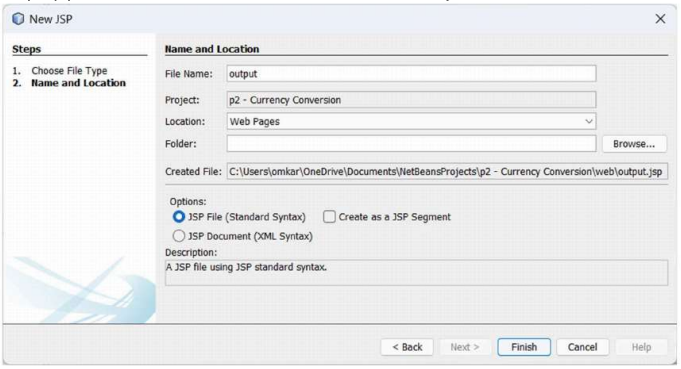


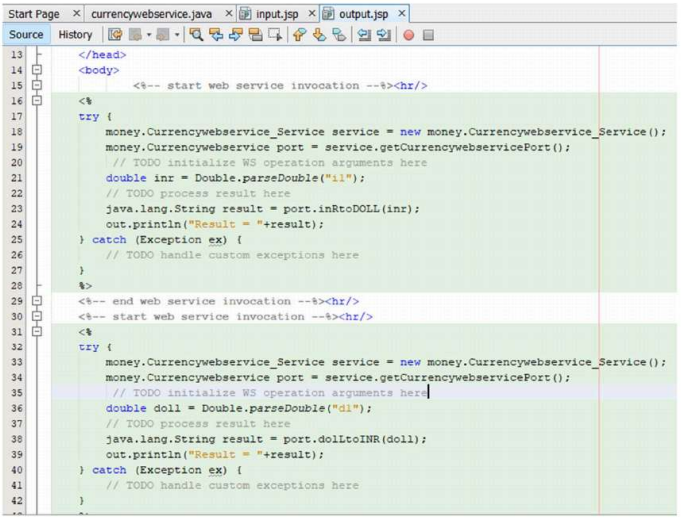
Step 7: Now to make front page, we need to create a JSP file. So, Right Click on Project Name > New > JSP > File Name: input > Finish. Now from the palette, add HTML Form (Action: output.jsp, Method: Post).





Step 8: Now to create the action behind that, we will create another JSP file, with a name that is declared in the previous JSP file. So, Right Click on Project Name > New > JSP > File Name: output.jsp > Finish. Then add some <%JSP code%> under body.

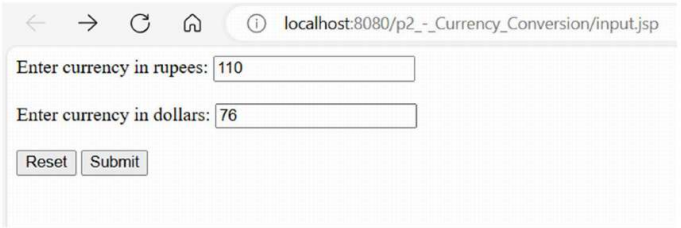




Step 9: Now RUN “index.jsp” file.

Step 10: Finish!

Outputs:

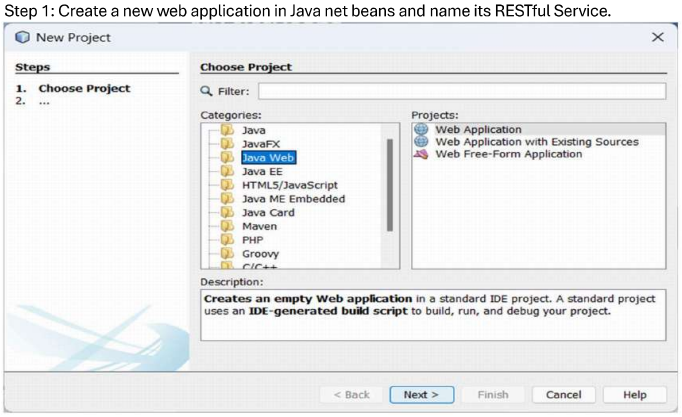


Practical 3 - Create RESTful Web service:

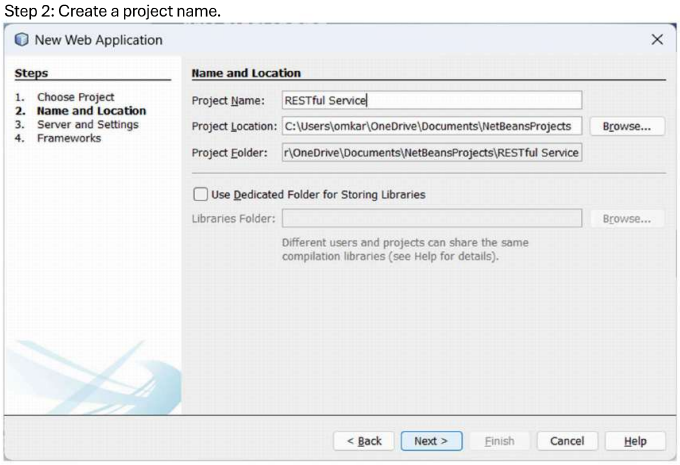
Aim: Define a RESTful web service that accepts the details to be stored in a database and performs CRUD operation.

Steps:

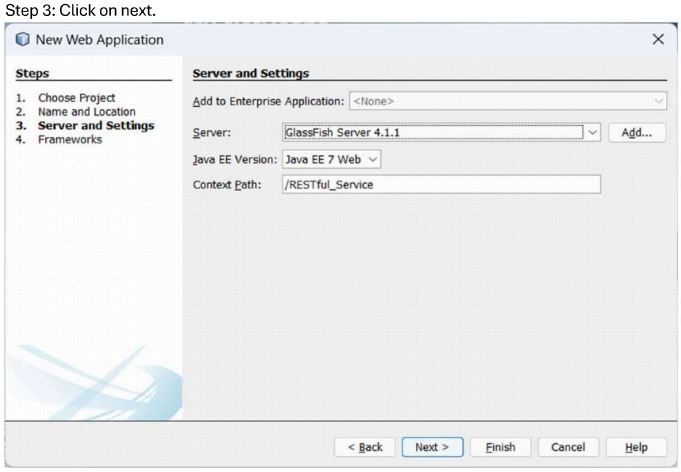
Step 1: Create a new web application in Java net beans and name its RESTful Service.



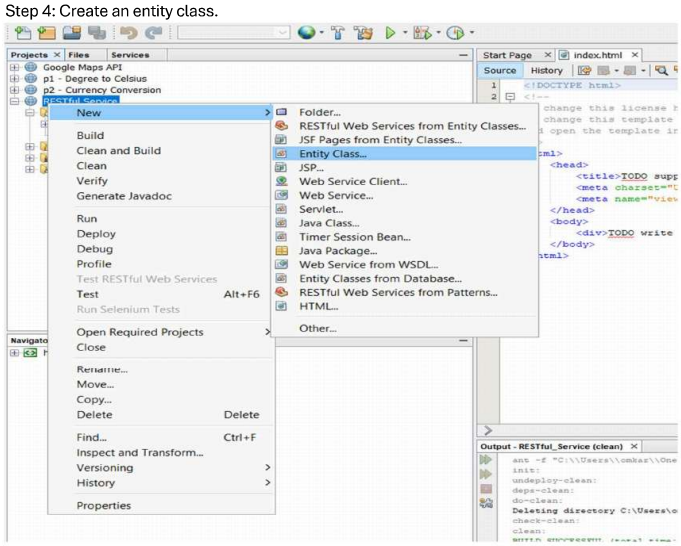
Step 2: Create a project name.

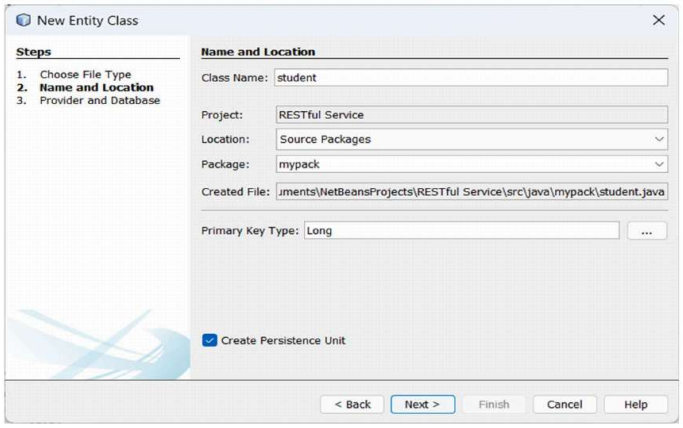


Step 3: Click on next.

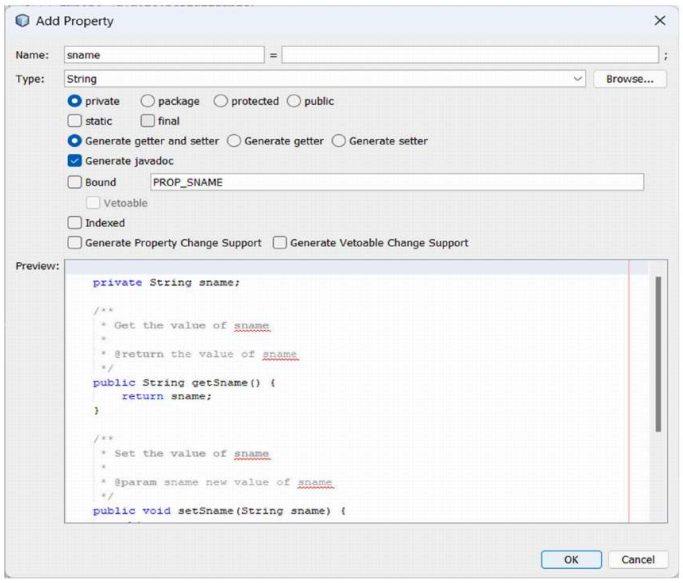


Step 4: Create an entity class.





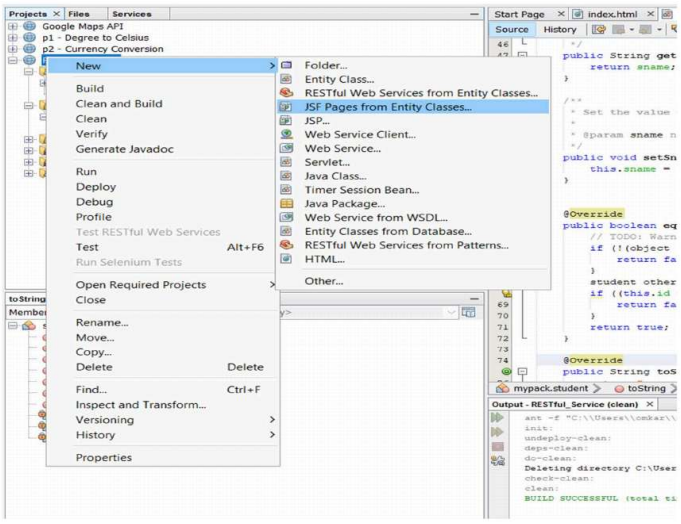
Step 5: Insert code after set id property.

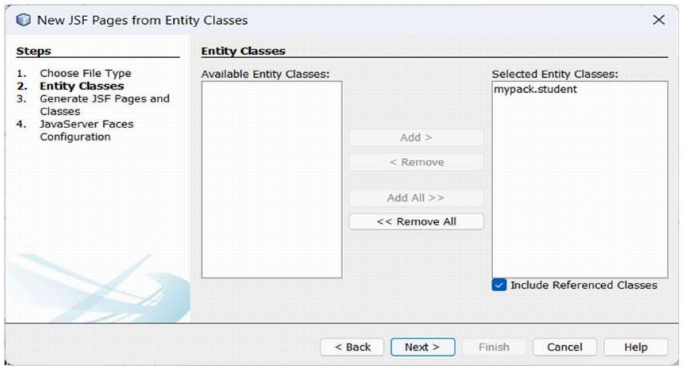


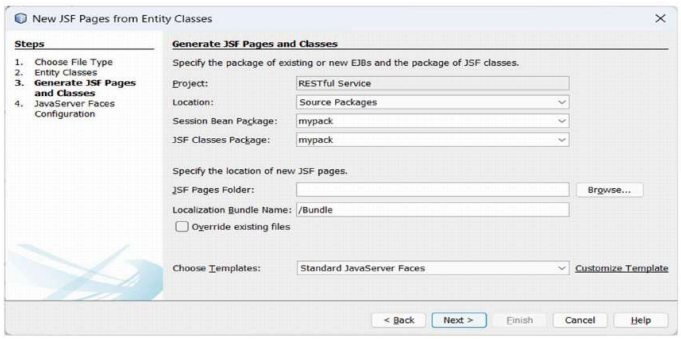
Step 6: Highlighted text below in the image will be generated.

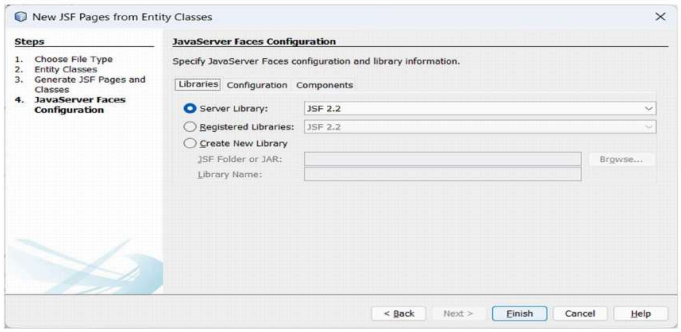
```
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }
    private String sname;

    /**
     * Get the value of sname
     *
     * @return the value of sname
     */
    public String getSname() {
        return sname;
    }

    /**
     * Set the value of sname
     *
     * @param sname new value of sname
     */
    public void setSname(String sname) {
        this.sname = sname;
    }
```

Step 7: Add JSF pages from the entity class.

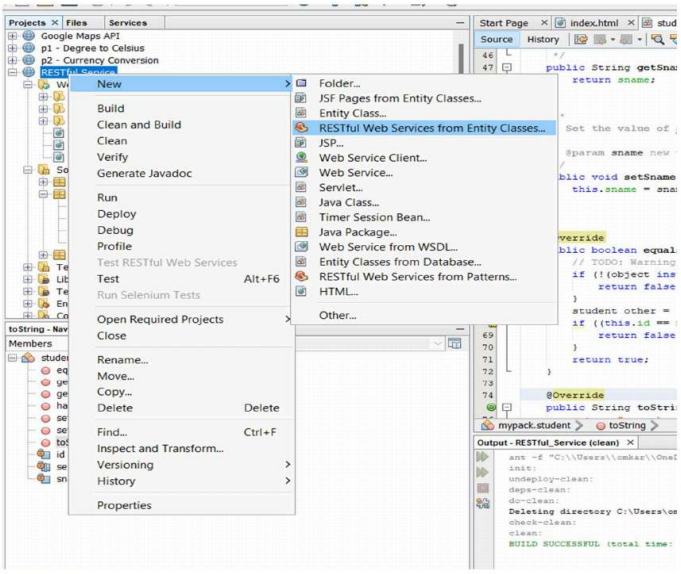


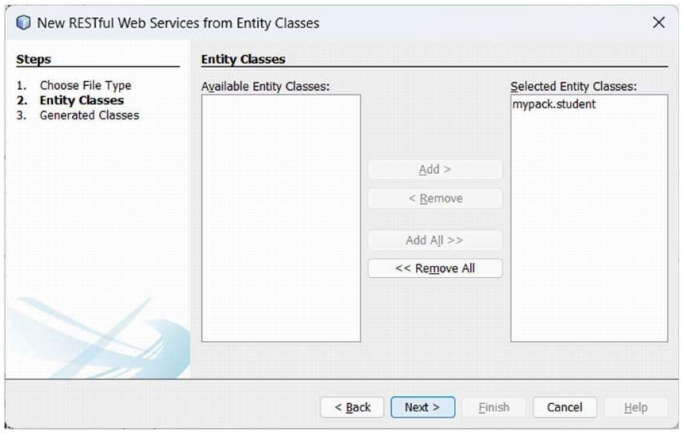

New JSF Pages from Entity Classes
Steps
1. Choose File Type
2. Entity Classes
3. Generate JSF Pages and Classes
4. JavaServer Faces Configuration
Entity Classes
Available Entity Classes:
Selected Entity Classes:
mypack.student
Add >
< Remove
Add All >>
<< Remove All
Include Referenced Classes
< Back
Next >
Finish
Cancel
Help

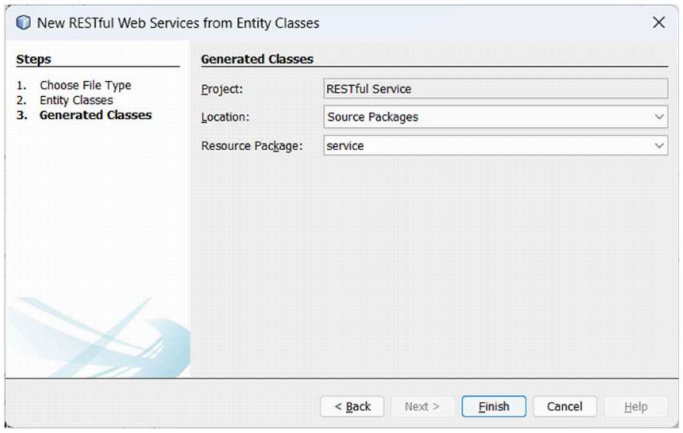

New JSF Pages from Entity Classes
Steps
1. Choose File Type
2. Entity Classes
3. Generate JSF Pages and Classes
4. JavaServer Faces Configuration
Generate JSF Pages and Classes
Specify the package of existing or new EJBs and the package of JSF classes.
Project:
RESTful Service
Location:
Source Packages
Session Bean Package:
mypack
JSF Classes Package:
mypack
Specify the location of new JSF pages.
JSF Pages Folder:
Browse...
Localization Bundle Name:
/Bundle
Override existing files
Choose Templates:
Standard JavaServer Faces
Customize Template
< Back
Next >
Finish
Cancel
Help


New JSF Pages from Entity Classes
Steps
1. Choose File Type
2. Entity Classes
3. Generate JSF Pages and Classes
4. JavaServer Faces Configuration
JavaServer Faces Configuration
Specify JavaServer Faces configuration and library information.
Libraries
Configuration
Components
Server Library:
JSF 2.2
Registered Libraries:
JSF 2.2
Create New Library
JSF Folder or JAR:
Browse...
Library Name:
< Back
Next >
Finish
Cancel
Help

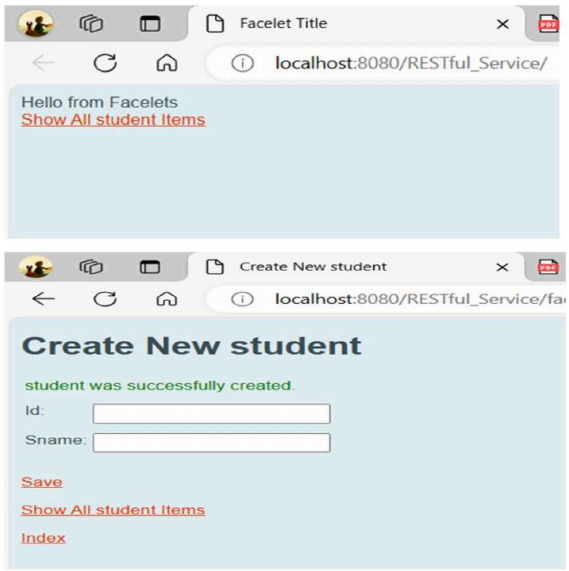
Step 8: Now add RESTful web services from entity class as shown below in the image.

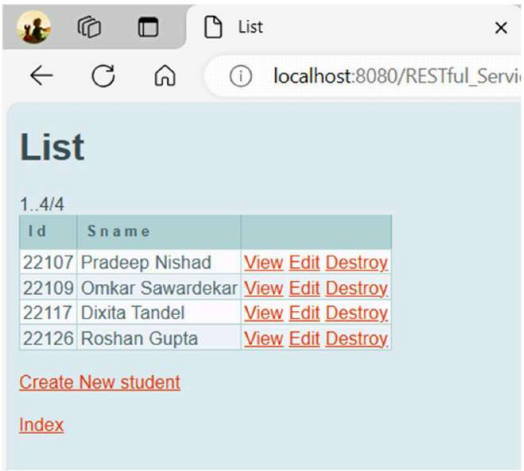






Step 9: Now deploy and run the project (make sure Oracle is shut on port 8080.





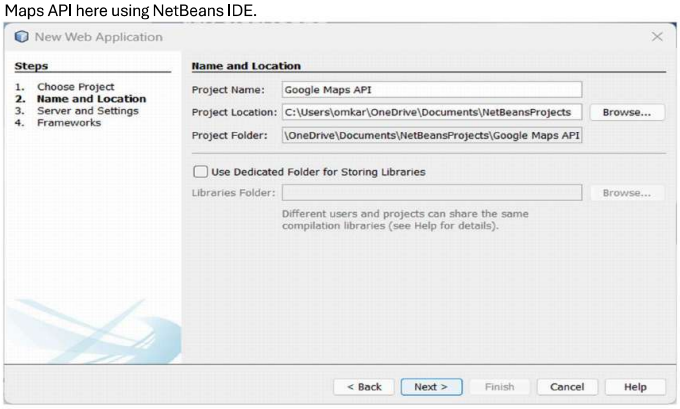
Step 10: Conclusion: we have successfully defined a RESTful web service that accepts the details to be stored in a database and performs CRUD operations. We are not storing data in DBMS we are just accepting in the list as we have to accept details.

Practical 4 - Create a Simple RESTful Web service:

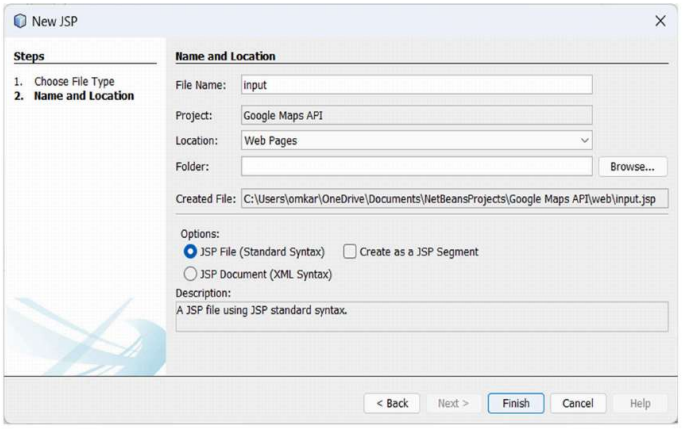
Aim: Develop application to consume Google's search / Google's Map RESTful Web service.

Steps:

Step 1: First of all, we need to create a Java Web Application with any name, let it be Google Maps API here using NetBeans IDE.



Step 2: Create input.jsp.



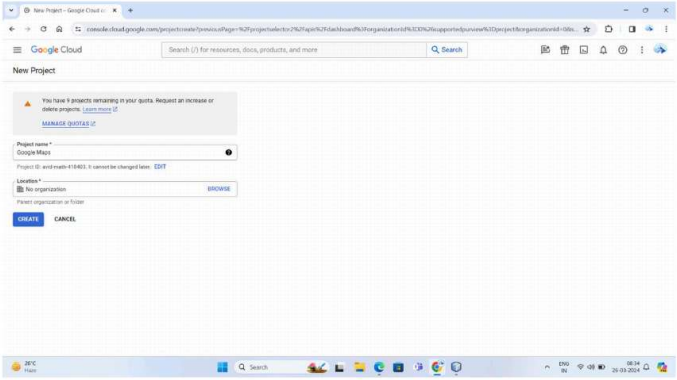
Step 3: The code inside the input.jsp will be:

Input.jsp

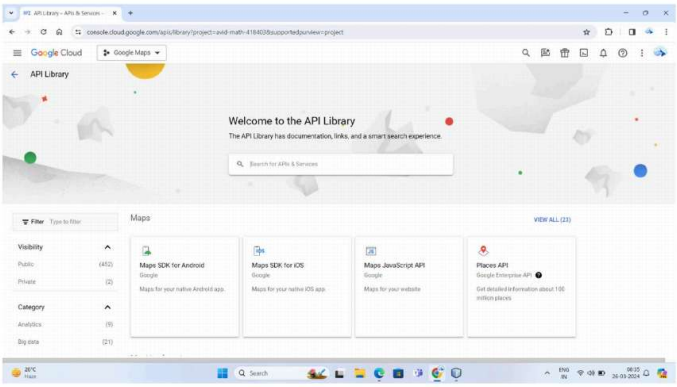
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <form action="index.jsp">
            <pre>
                Enter latitude:<input type="text" name="t1" />
                Enter longitude:<input type="text" name="t2" />
                <input type="submit" value="Show" />
            </pre>
        </form>
    </body>
</html>
```

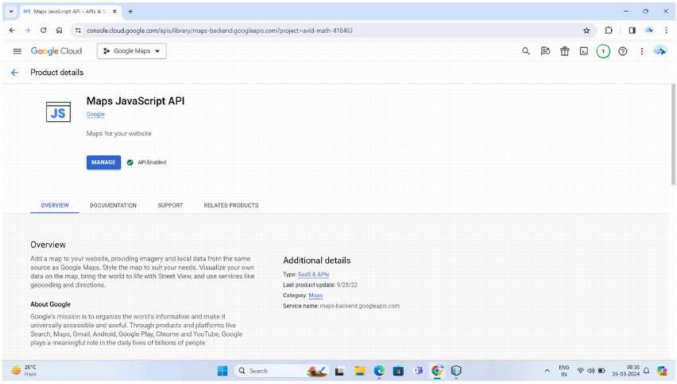
Step 4: Before running the application, we need the Google API key. The steps are shown here:
- Visit Google APIs Console (https://console.developers.google.com, you have to login with your Google account).

a) Create a new API Project.

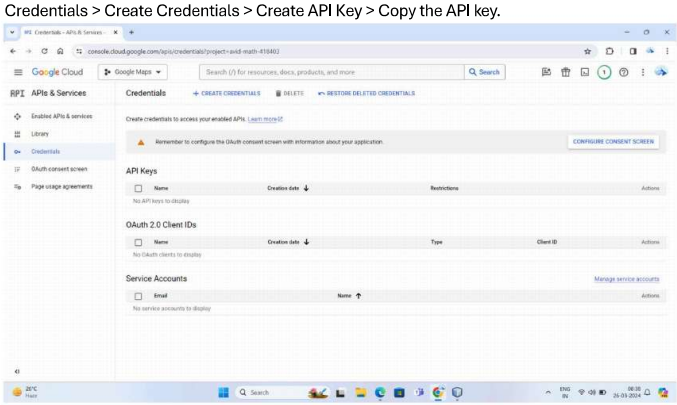


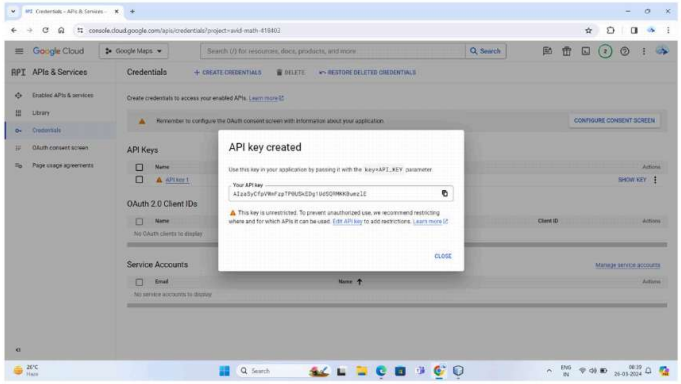
b) Enable the Google Maps API V3.



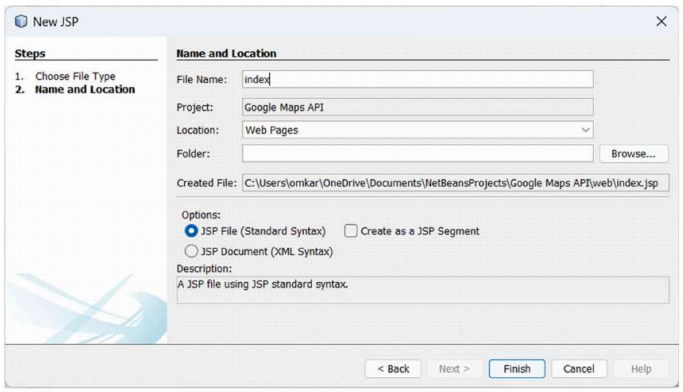


Step 5: Now, to create an API key, go to Google Cloud > Enabled API and Services > Credentials > Create Credentials > Create API Key > Copy the API key.





Step 6: Create another file index.jsp.



Index.jsp

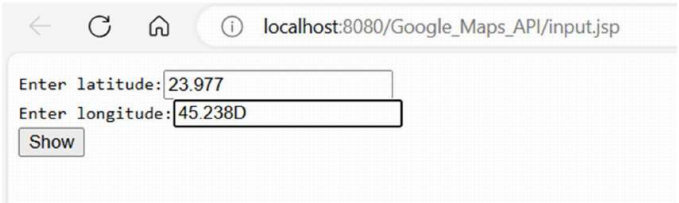
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
```

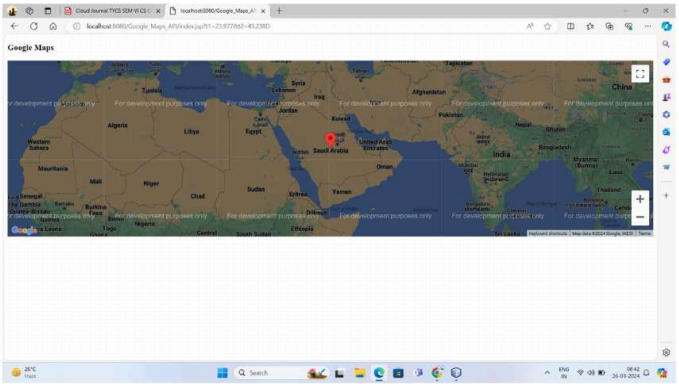
```
        <style>
        #map
        {
        height: 400px;
        width: 100%;
        }
        </style>
</head>
<body>
        <%
                double lati=Double.parseDouble(request.getParameter("t1"));
                double longi=Double.parseDouble(request.getParameter("t2"));
        %>
        <h3> Google Maps </h3>
        <div id="map"></div>
        <script lang="javascript">
        function initMap()
        {
                var info={lat: <%=lati%>, lng: <%=longi%>};
                var map = new google.maps.Map(document.getElementById('map'),
        {
                zoom: 4, center: info
        });
                var marker = new google.maps.Marker
        ({
                position: info, map: map
        });
        }
        </script>
```

```
            <script async defer
            src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCfpVWmFzpTP0USkEDg1UdSQRMKK0umzlE&callback=initMap">
            </script>
        </body>
</html>
```

Step 7: Run “input.jsp” file.

Outputs:





Practical 5 – KVM:

Aim: To install and configure virtualization using KVM.

Steps:

Step 1: Check Hardware Virtualization Support:

Ensure that your CPU supports hardware virtualization (Intel-VT or AMD-V) and that is enabled in the BIOS settings

Step 2: Install KVM Packages:

1) Update your package repository: sudo apt update
2) Install the KVM and QEMU packages: sudo apt install qemu -kvm libvirt-daemon-system libvirt-clients bridge-utils

Step 3: Start and Enable Libvirt Service:

1) Start the libvirt service: sudo systemct1 start libvirtd
2) Enable libvirt to start on boot: sudo systemct1 enable libvirtd

Step 4: Configure Networking:

1) Create the network bridge interface: sudo nano/etc/network/interfaces
2) Add the following lines: auto br0 iface br0 inet dhcp bridge_ports enp0s3
3) Restart the networking service: sudo systemct1 restart networking

Step 5: Create a Virtual Machine:

You can create and manage virtual machines using tools like virt-manager or virsh.

Step 6: Install Virt-Manager:

If you prefer GUI tool for managing virtual machine, you can install virt-manager: sudo apt install virt-manager

Step 7: Launch Virt-Manager:

You can launch Virt-Manager from the application menu or using the command virt-manager in the terminal.

Step 8: Create Virtual Machine:

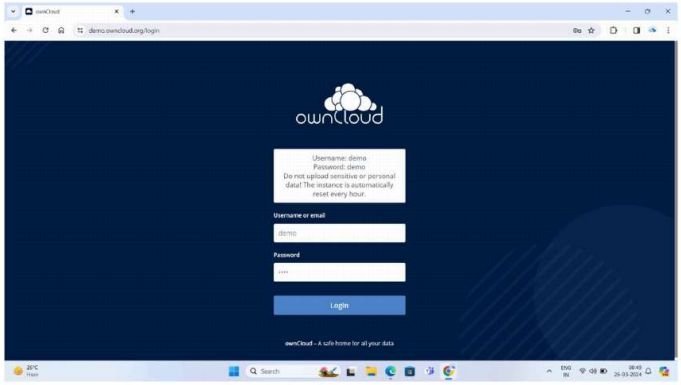
Using Virt-Manager, you can create, configure and manage virtual machines on your KVM hypervisor

Practical 6 – OwnCloud:

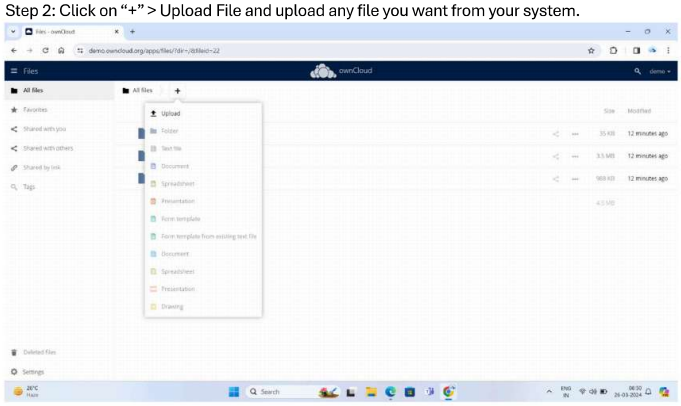
Aim: To upload file on OwnCloud and share it across internet.

Steps:

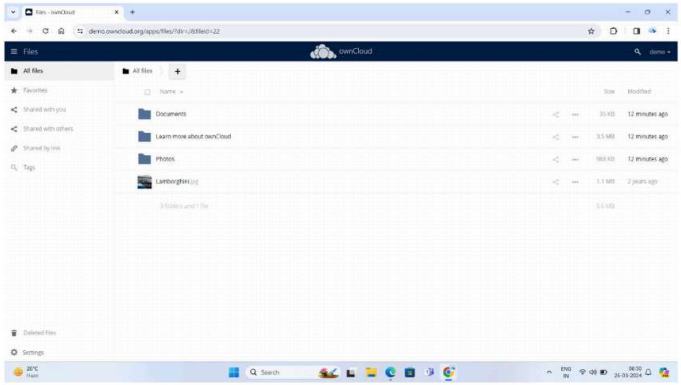
Step 1: Go to Google Chrome, search “OwnCloud demo”. Then open the website, fill up the username and password as “demo”.



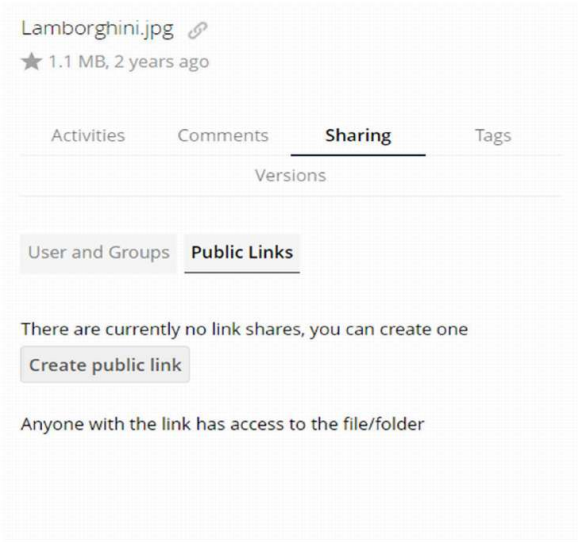
Step 2: Click on “+” > Upload File and upload any file you want from your system.



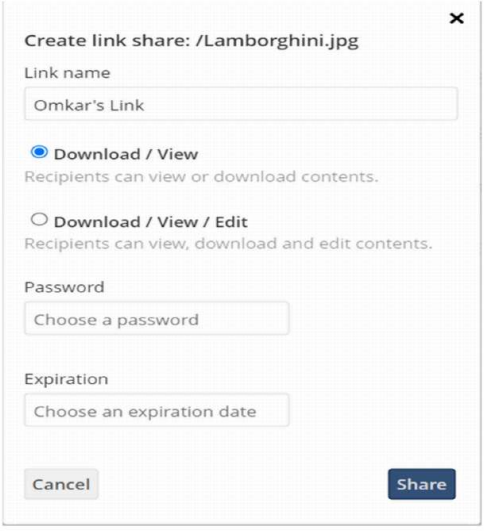
Step 3: It will take few seconds to upload the file. Once uploaded, you will see it as added at bottom.



Step 4: Click on the file. A menu will open on right hand side of the web page. Go to Sharing window.



Step 5: Click on Create public link. A window will open, give the name of your link.



Step 6: Now a link will be created at right hand side menu. You can now share it across internet via email or any other social media platform.