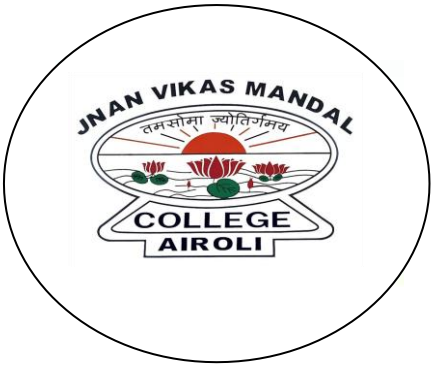
Cloud Computing And Web Services

T.Y.B.Sc Computer Science

(Semester-VI)

For Academic Year (2023-24)



JNAN VIKAS MANDAL'S

PADMASHREE DR. R.T.DOSHI DEGREE COLLEGE OF COMPUTER SCIENCE

MOHANLAL RAICHAND MEHTA COLLEGE OF COMMERCE

DIWALIMAA DEGREE COLLEGE OF SCIENCE

AMRATLAL RAICHAND MEHTA COLLEGE OF ARTS

JVM'S DEGREE COLLEGE OF INFORMATION TECHNOLOGY

AIROLI, NAVI MUMBAI – 400708

CERTIFICATE

This is to certify that the Mr./Miss. _____ having Roll No _____ of B.Sc.CS Semester VI has completed the practical work in the subject of **Cloud Computing and Web Services** during the Academic year 2023-24 being the partial requirement for the fulfillment of the curriculum of Degree of Bachelor of Science in Computer Science, University of Mumbai.

Place: **Date:**

Sign of Subject In-Charge

External Examiner

CS-IT Coordinator

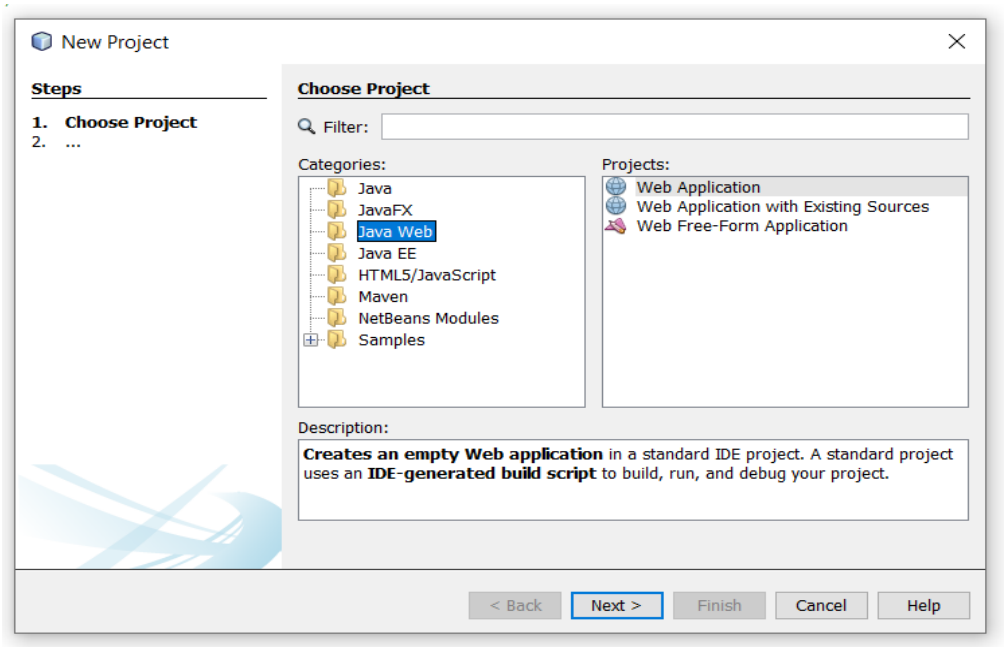
INDEX

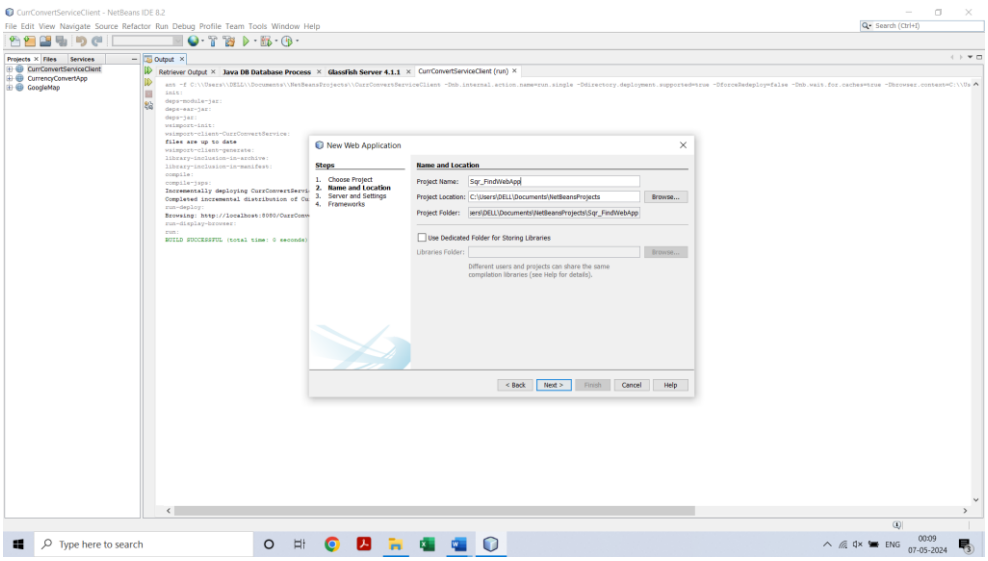
Sr.No.	Practical's	Date	Sign
1	Define a simple services like Converting Rs into Dollar and Call it from different platform like JAVA and .NET		
2	Create a Simple SOAP service to find the Square of a Number		
3	Create a Simple REST Service to demonstrate CRUD operations with "Student" database		
4	Develop application to consume Google's search / Google's Map RESTful Web service.		
5	Installation and Configuration of virtualization using KVM.		
6	Develop application to download image/video from server or upload image/video to server using MTOM techniques		
7	Implement FOSS-Cloud Functionality VSI (Virtual Server Infrastructure) Infrastructure as a Service (IaaS), Storage		
8	Implement FOSS-Cloud Functionality - VSI Platform as a Service (PaaS),		
9	Using AWS Flow Framework develop application that includes a simple workflow. Workflow calls an activity to print hello world to the console. It must define the basic usage of AWS Flow Framework, including defining contracts, implementation of activities and workflow coordination logic and worker programs to host them		
10	Implementation of Openstack with user and private network creation.		

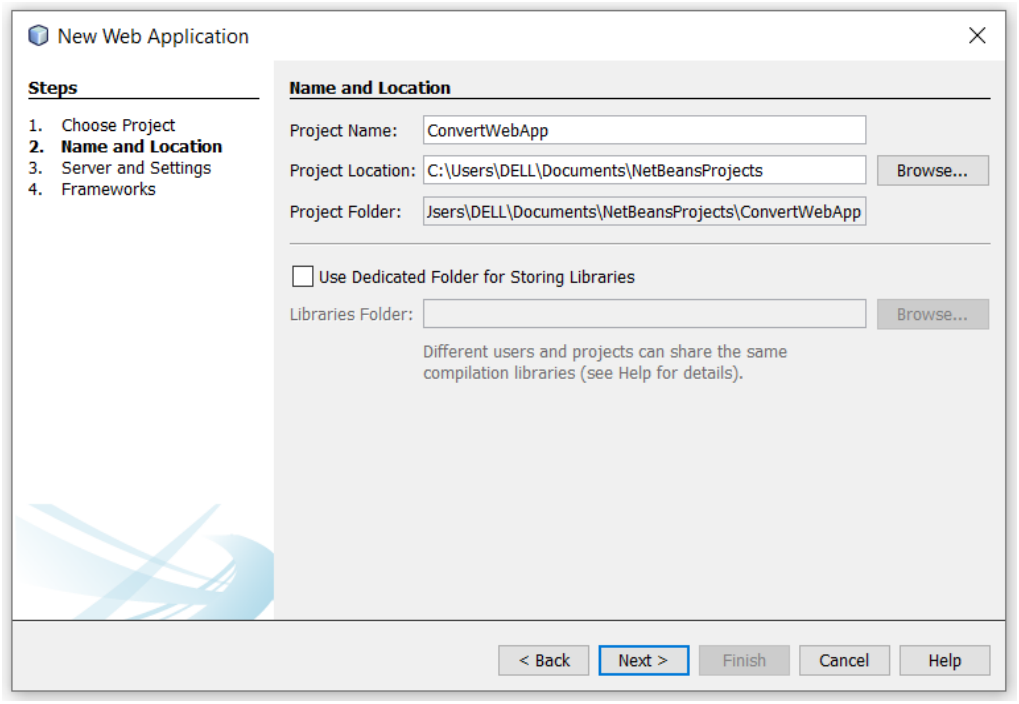
Practical-1(A)

Define a simple services like Converting Rupees into Dollar and Call it from different platform like JAVA.

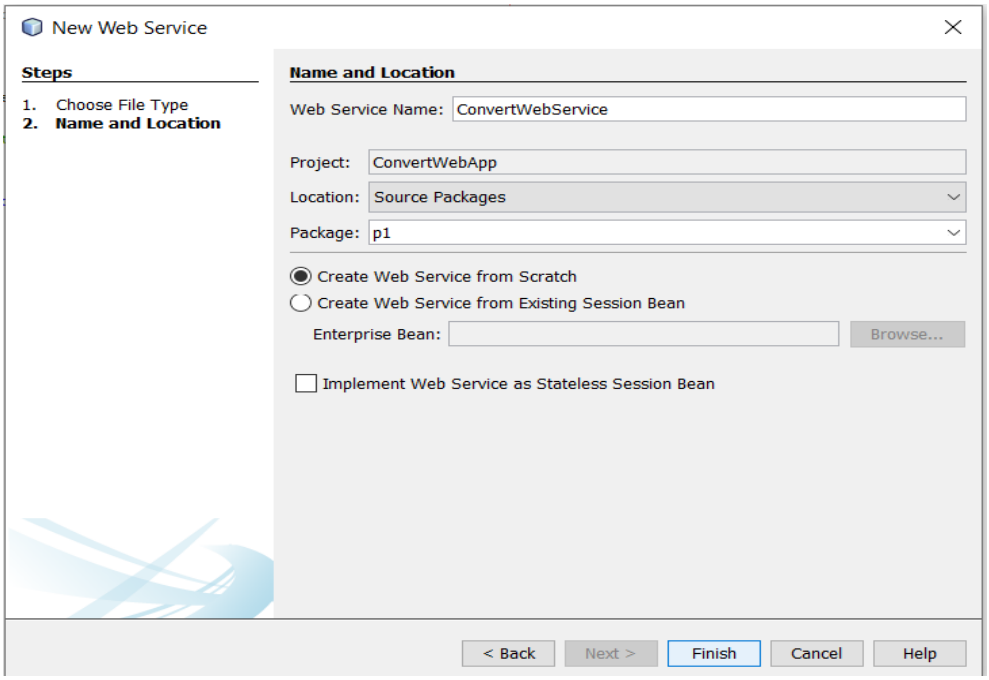
Step1 : To create Web Service through Net Beans



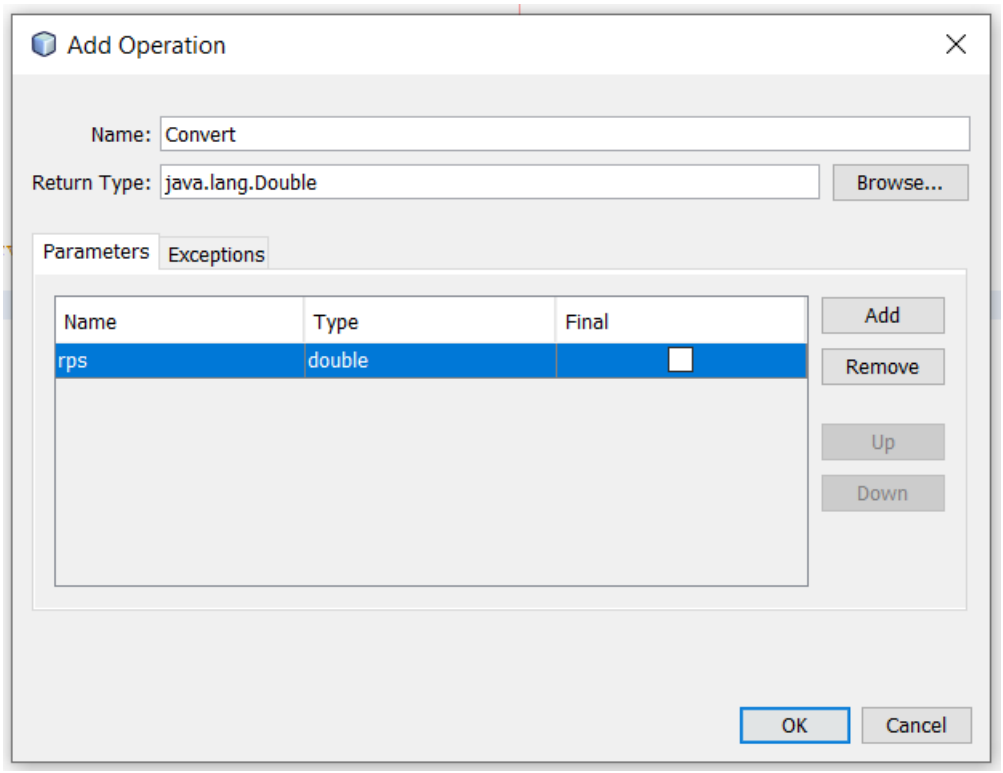




Step 2: To create Web Service " ConvertWebService"

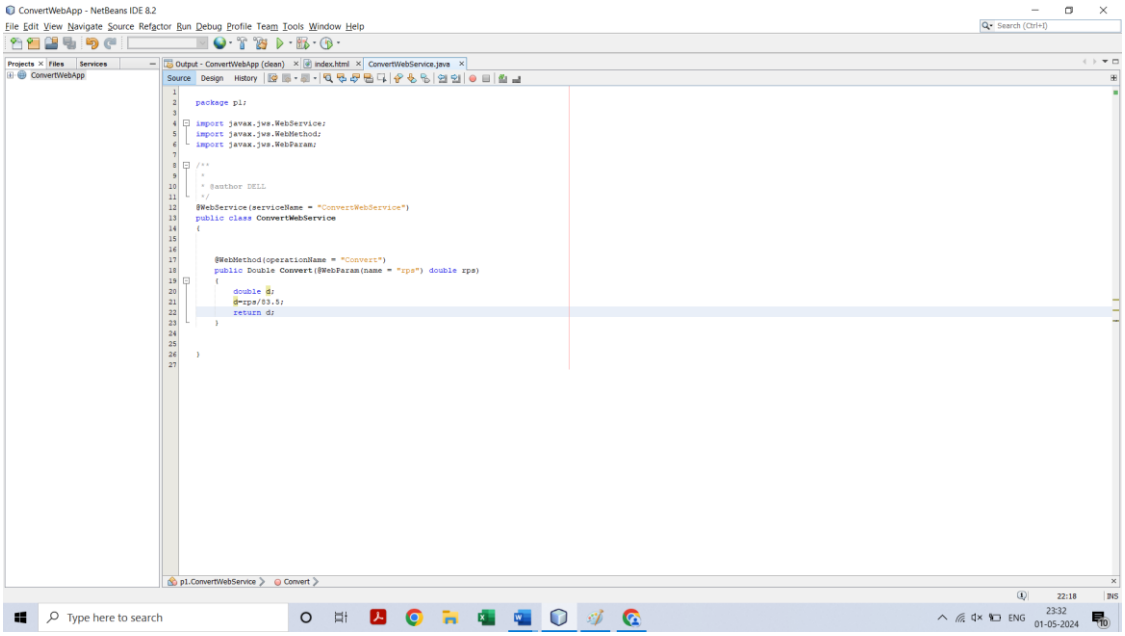


Step 3 : to Add Operation "Convert" to Web Service



Here Covert is a method having one parameter as rps

Step 4 : After adding code of Operation to web Service

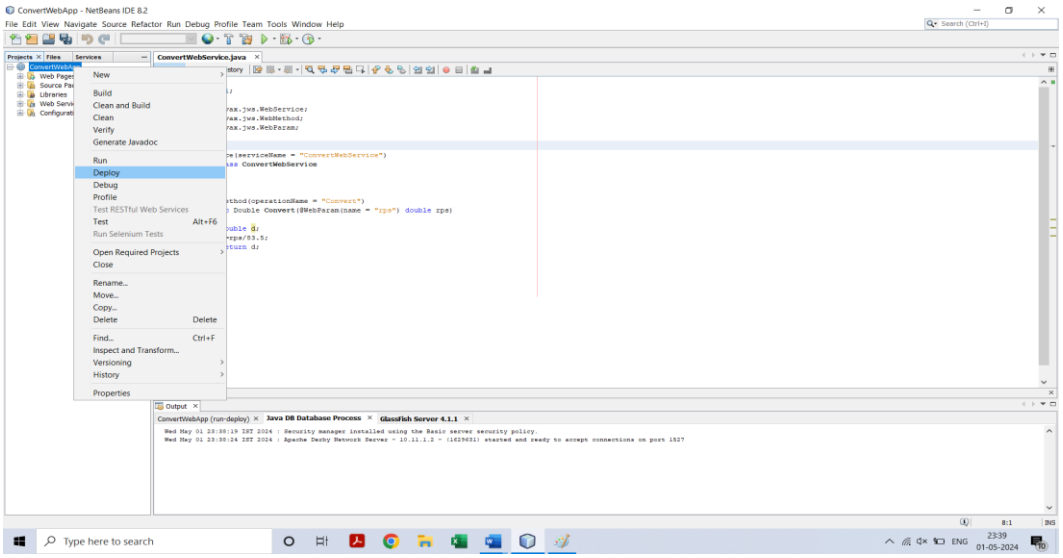


Code of Operation :

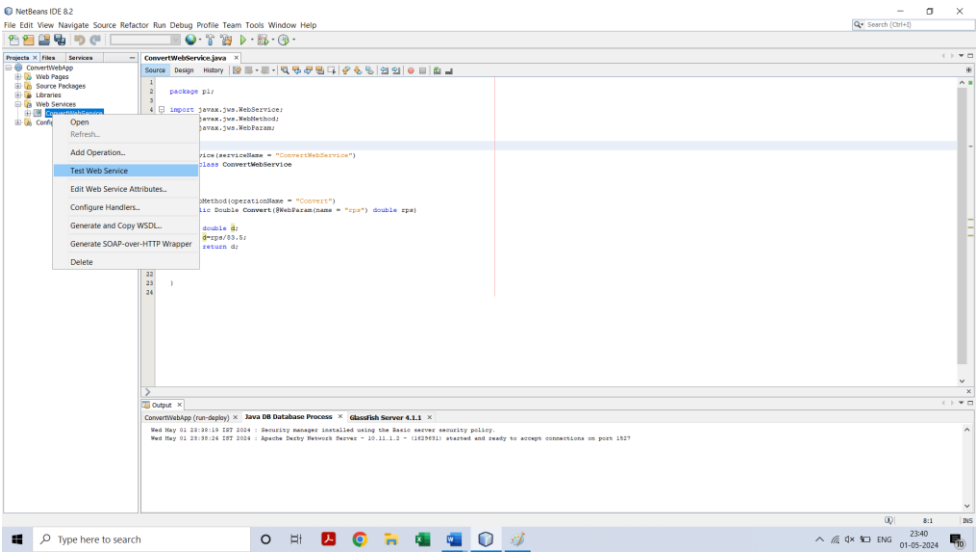
```
@WebMethod(operationName = "Convert")
   public Double Convert(@WebParam(name = "rps") double rps)
   {
      double d;
      d=rps/83.5;
      return d;
   }
```

Step 5 : Now to deploy the web service and Test

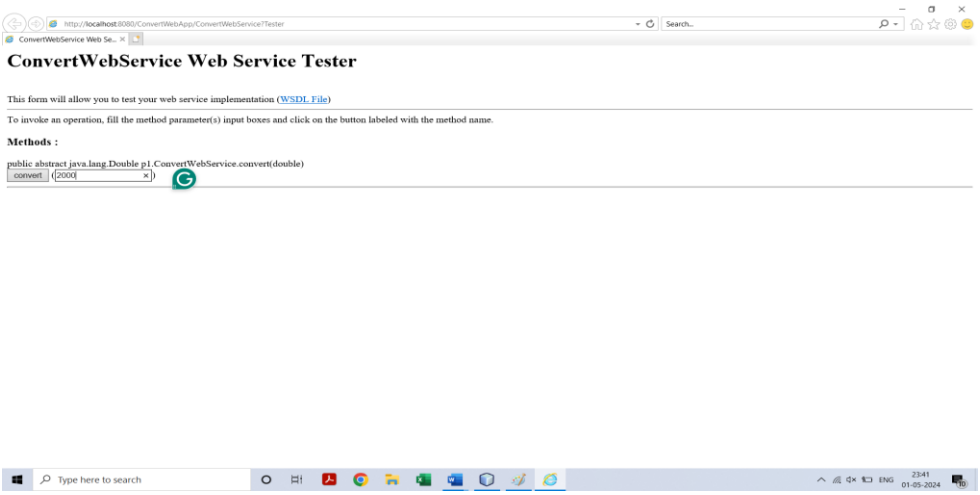
i) Deploy



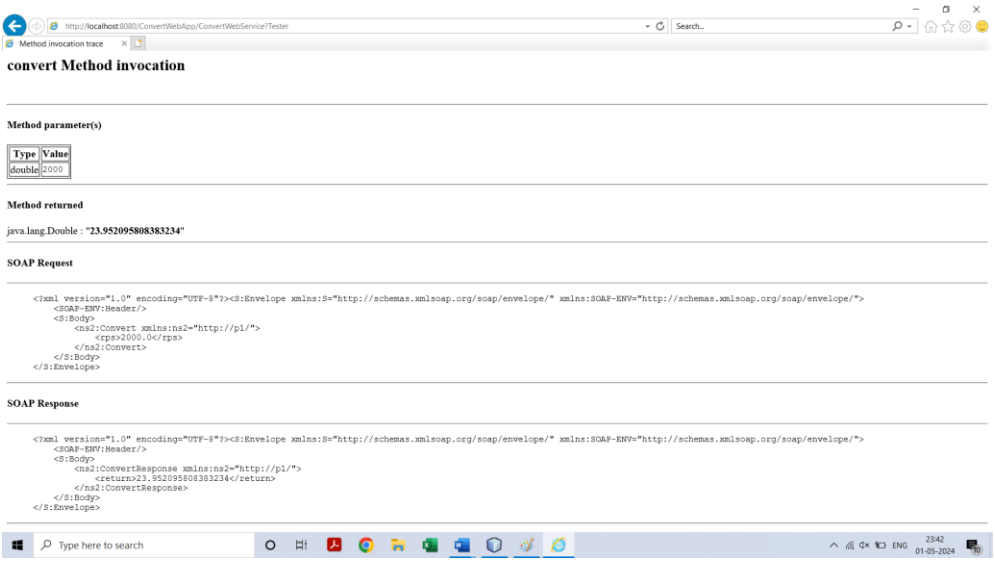
ii) Test Web service



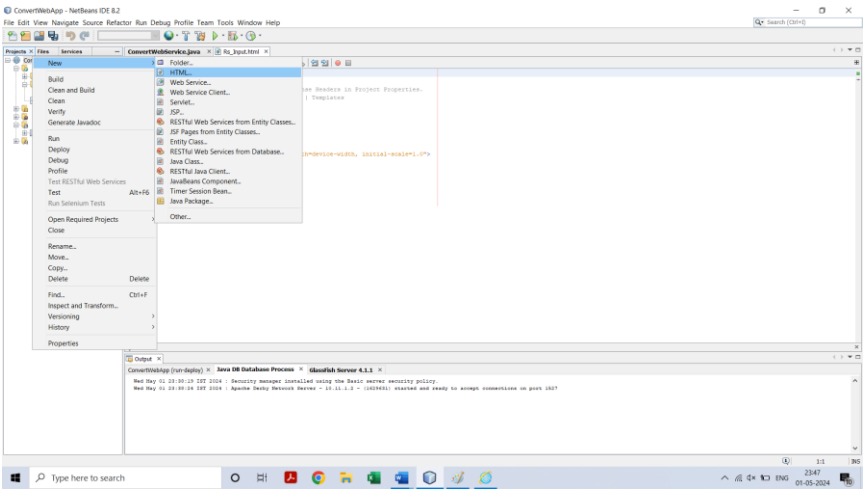
After testing

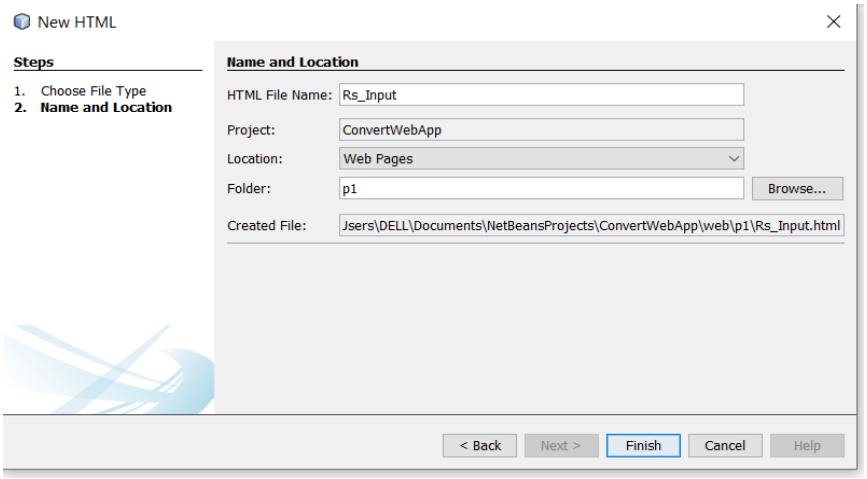


Successfully Deployed and Tested



Step 6: Now to take user input Create input.html page



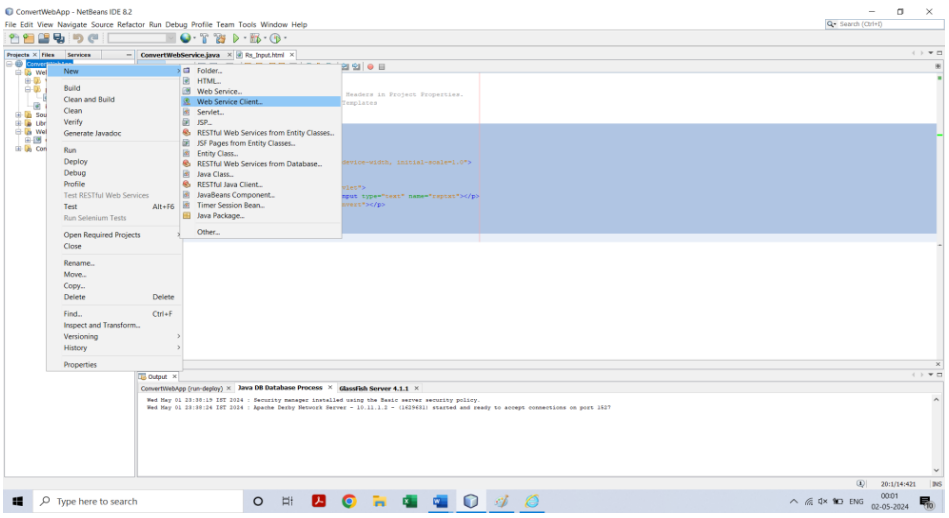


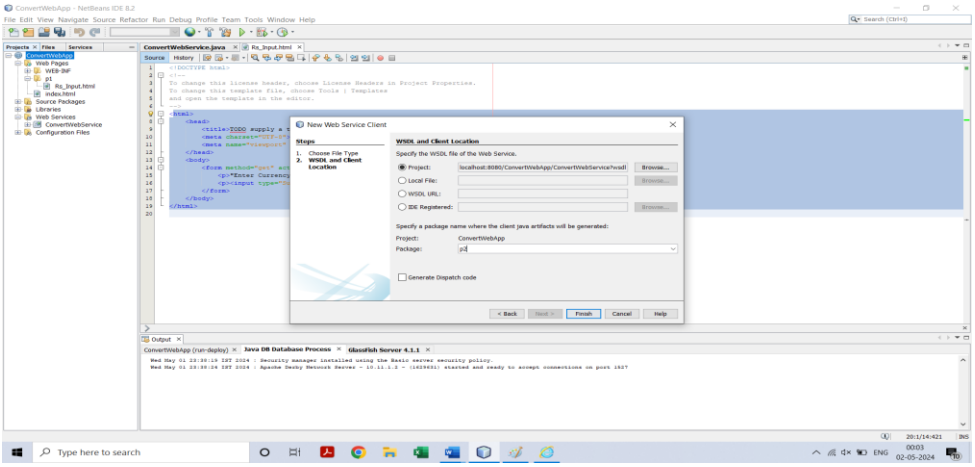
Now insert the code to create html page :

```
< html>
            <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
        <body>
    <form method="get" action="Output.jsp">
      <p>"Enter Currency in Ruppes " <input type="text" name="rsptxt"></p>
      <p><input type="Submit" value="Convert"></p>
    </form>
                </body>
</html>
```

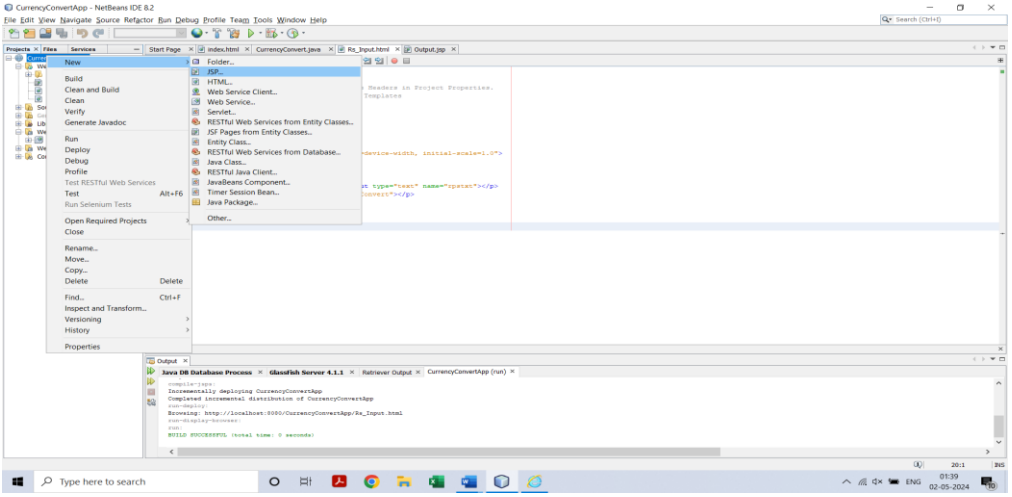
Step 7 :

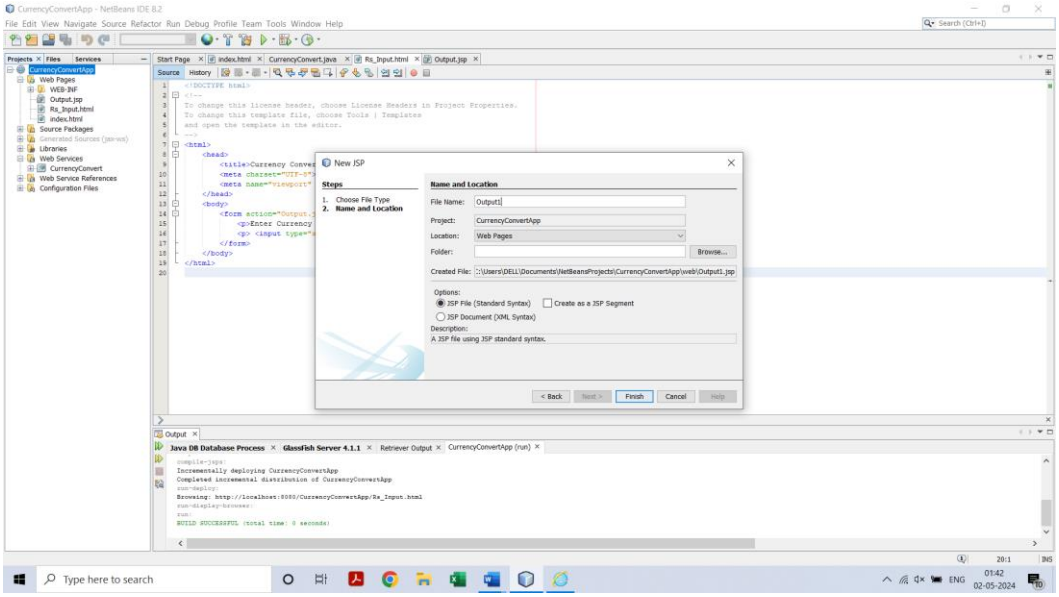
Now to create web service client, for same right click on project and select new then select web service client you will get the following screen:



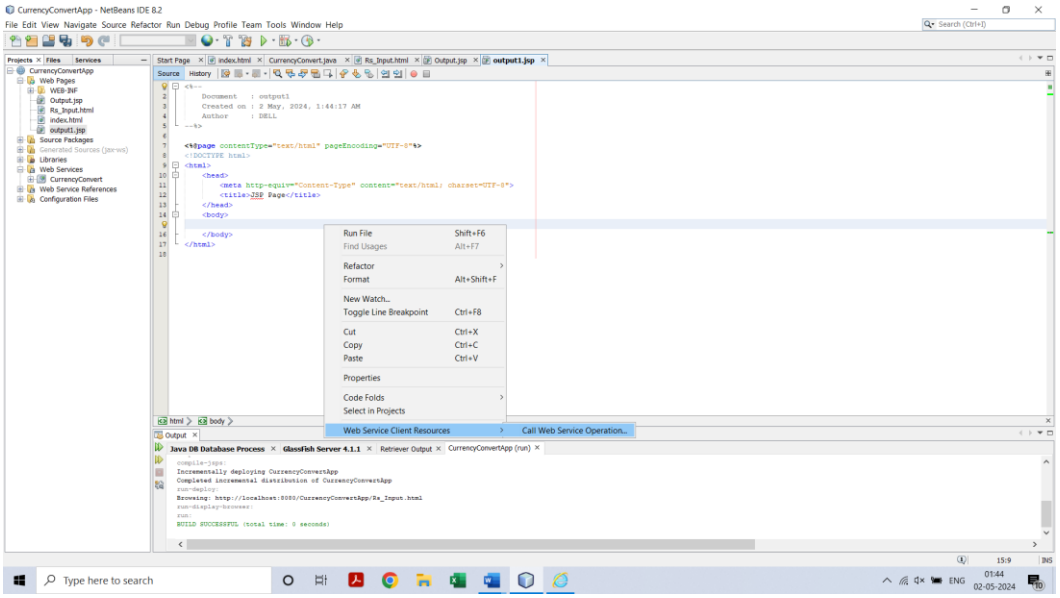


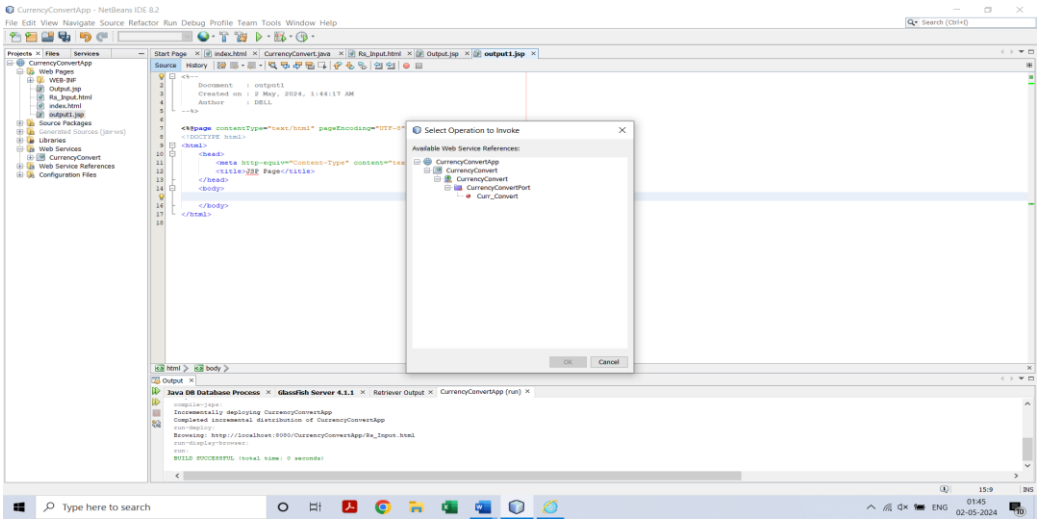
Step 8: Now finally to create Output.jsp page for output





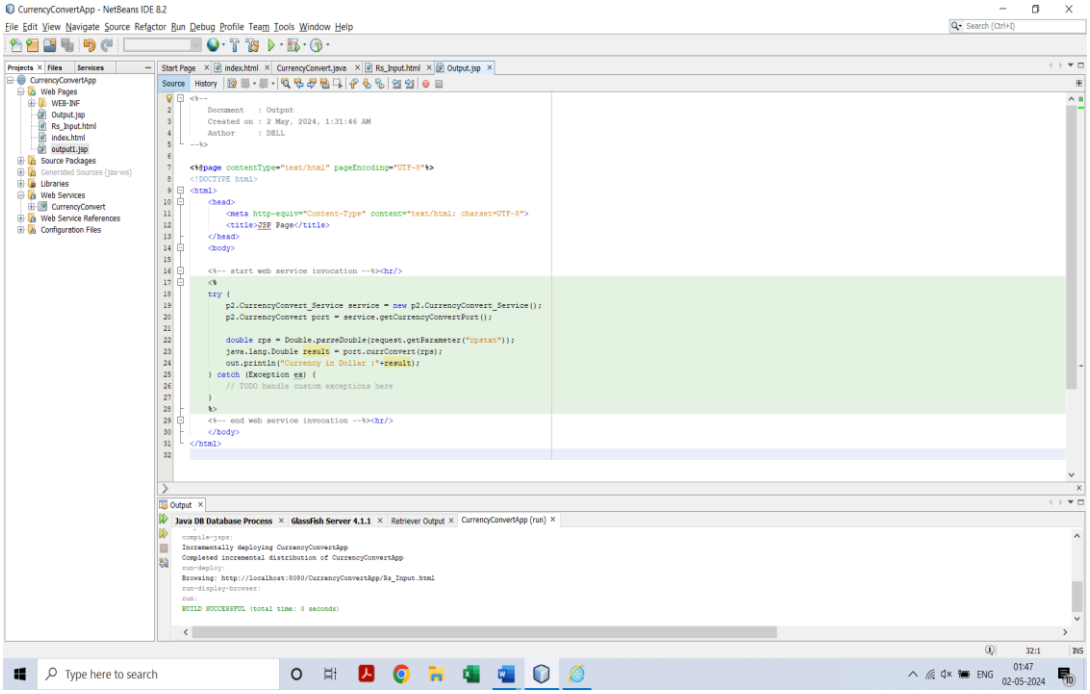
Step 9: After creating the page code is to be added





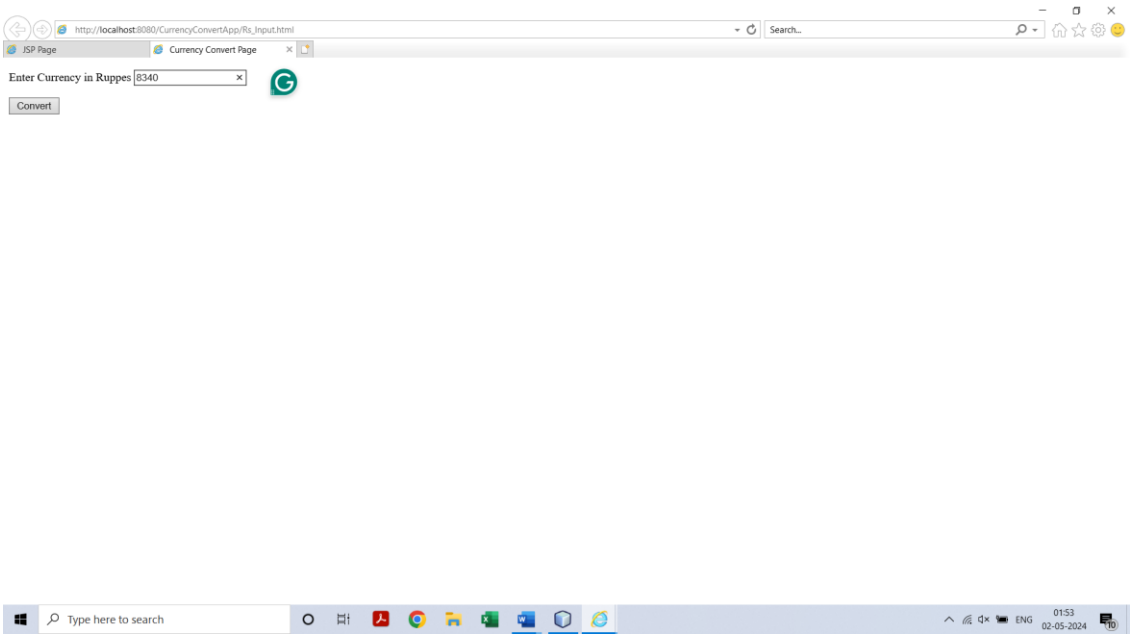
Code to Output.jsp page

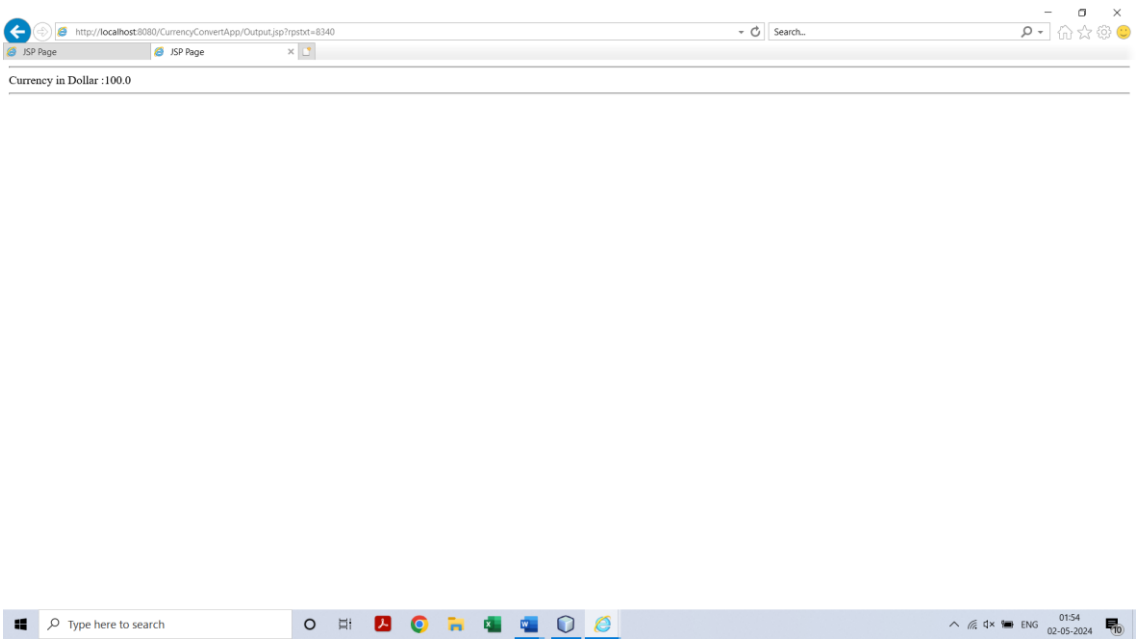
```
<%
   try {
  p2.CurrencyConvert_Service service = new p2.CurrencyConvert_Service();
  p2.CurrencyConvert port = service.getCurrencyConvertPort();

  double rps = Double.parseDouble(request.getParameter("rpstxt"));
      java.lang.Double result = port.currConvert(rps);
  out.println("Currency in Dollar :"+result);
   } catch (Exception ex) {
  // TODO handle custom exceptions here
   }
   %>
```



Step 10 : Run the application. The following output will be genereted

Output :

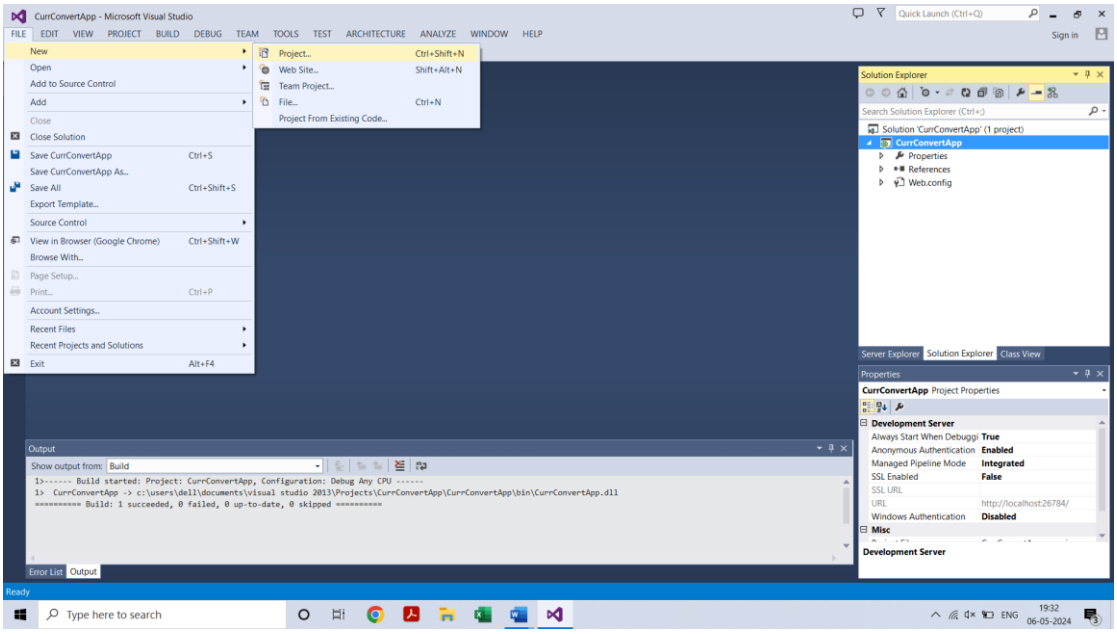


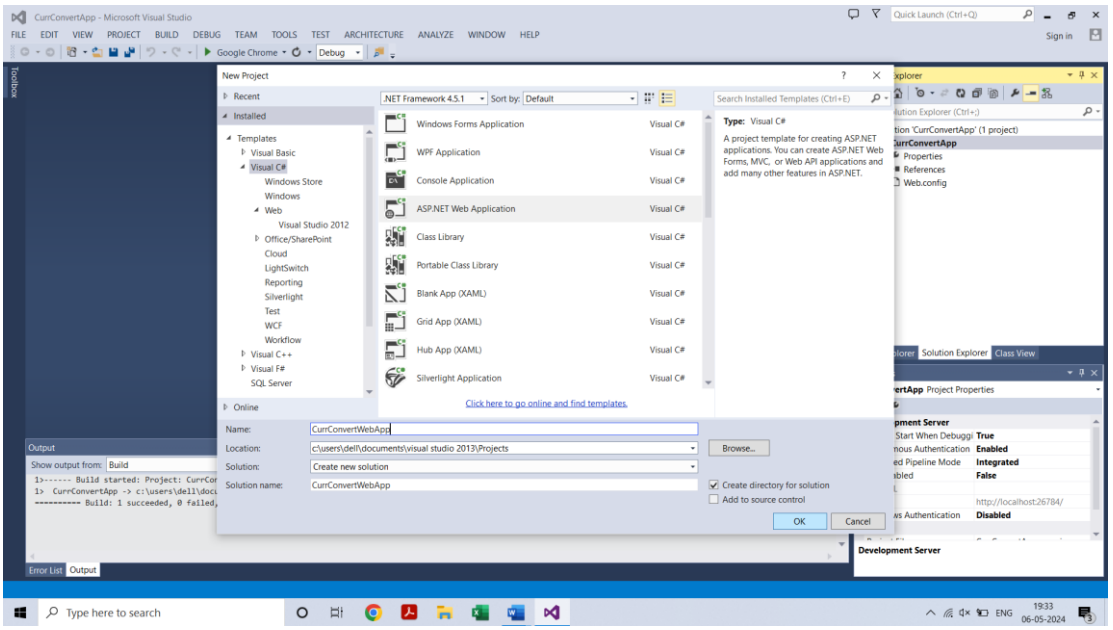

http://localhost:8080/CurrencyConvertApp/Output.jsp?rpstxt=8340
Search...
JSP Page
JSP Page
Currency in Dollar :100.0
Type here to search
ENG
01:54
02-05-2024

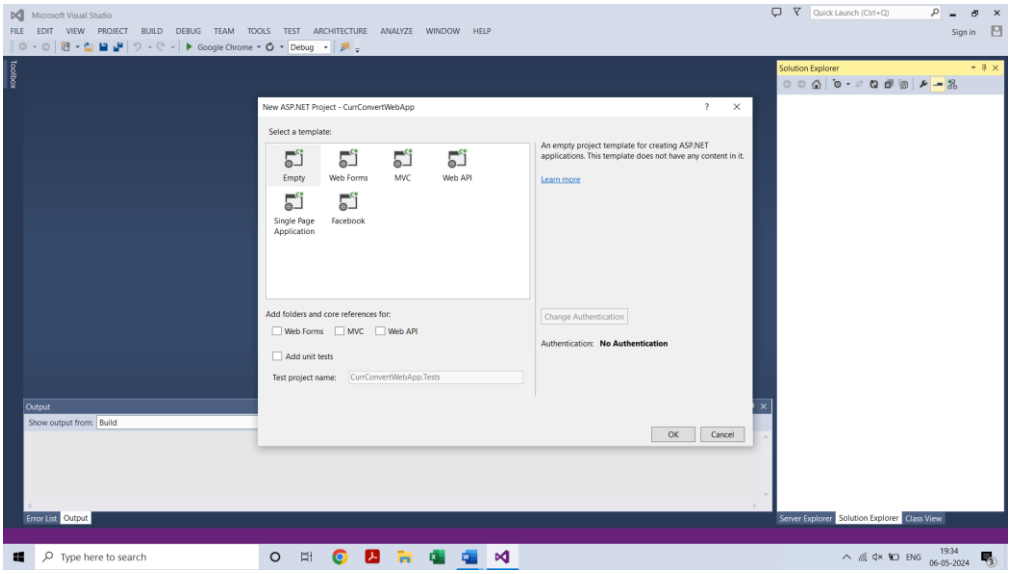
Practical-1(B)

Define a simple services like Converting Rupees into Dollar and Call it from different platform like .NET

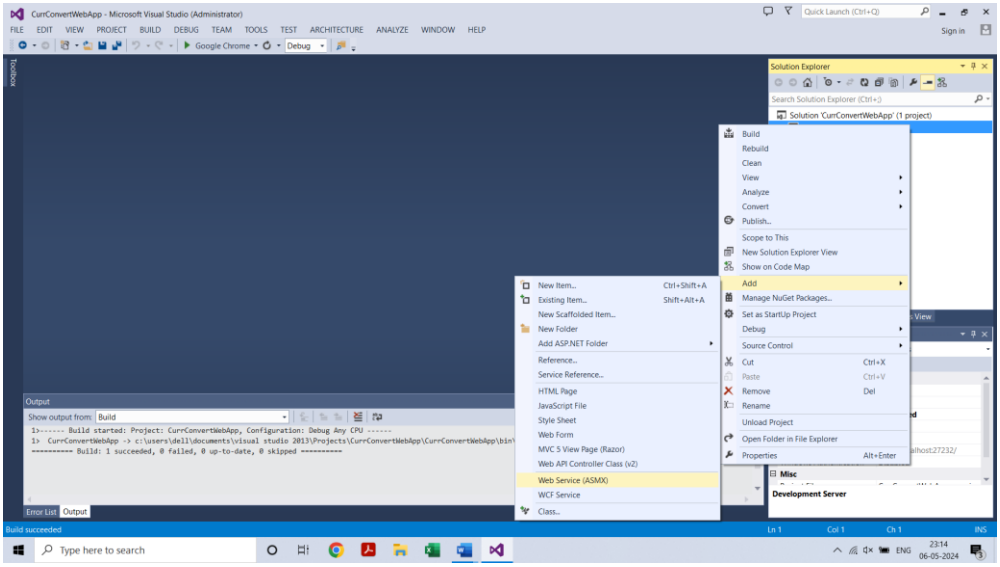
Step1 : To create Web Service through Visual Studio

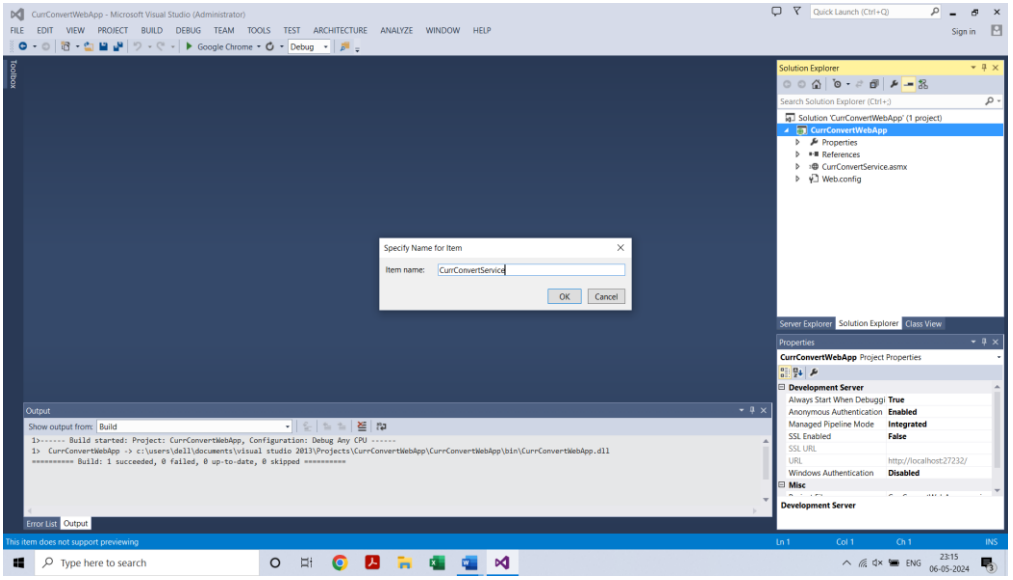






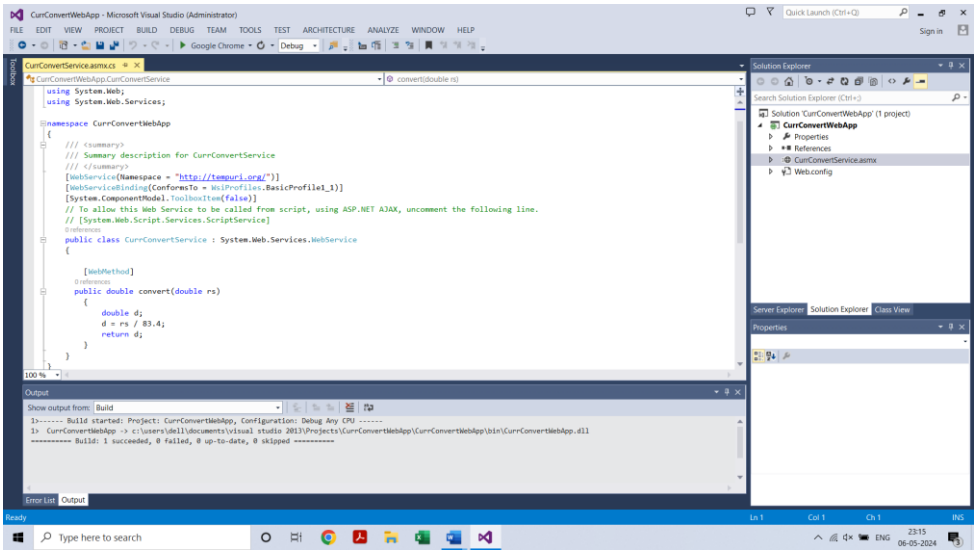
Now to Create a web Service :



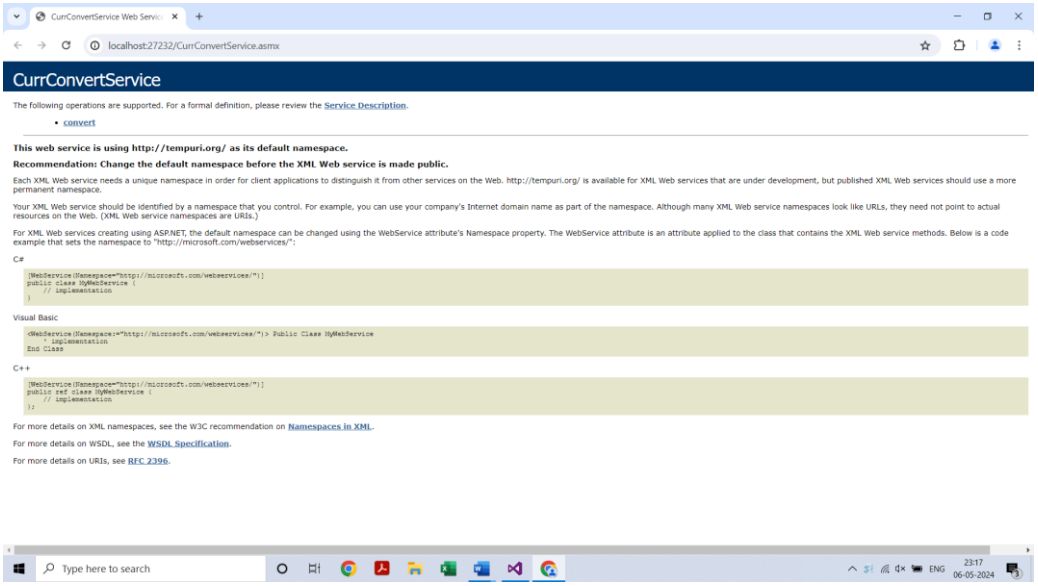


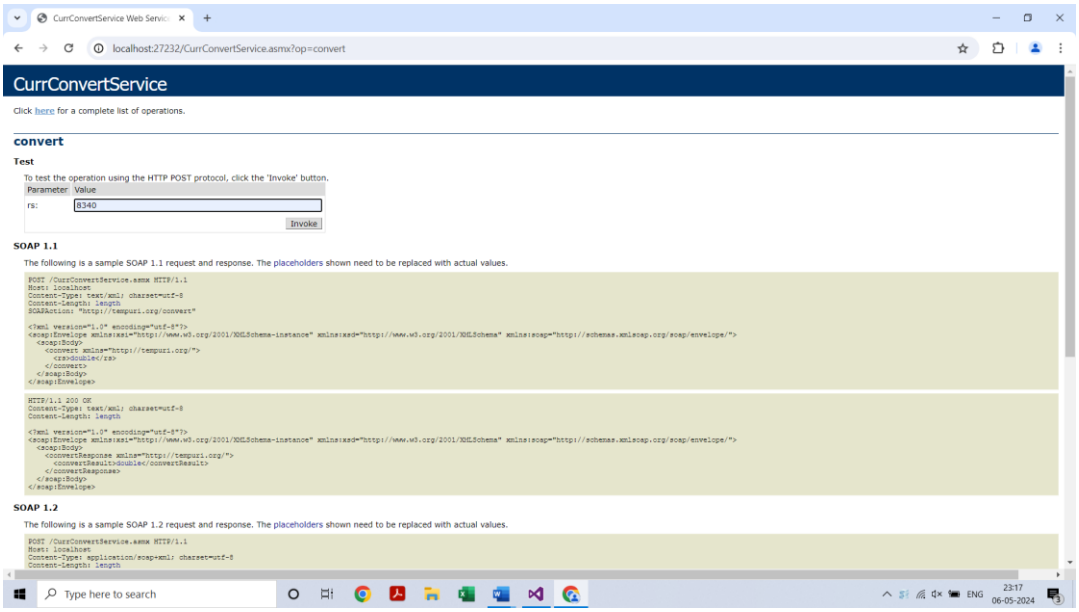
After click on Add button, CurrConvertService.asmx.cs file will be automatically opened. Add the following code into Class CurrConvertService :

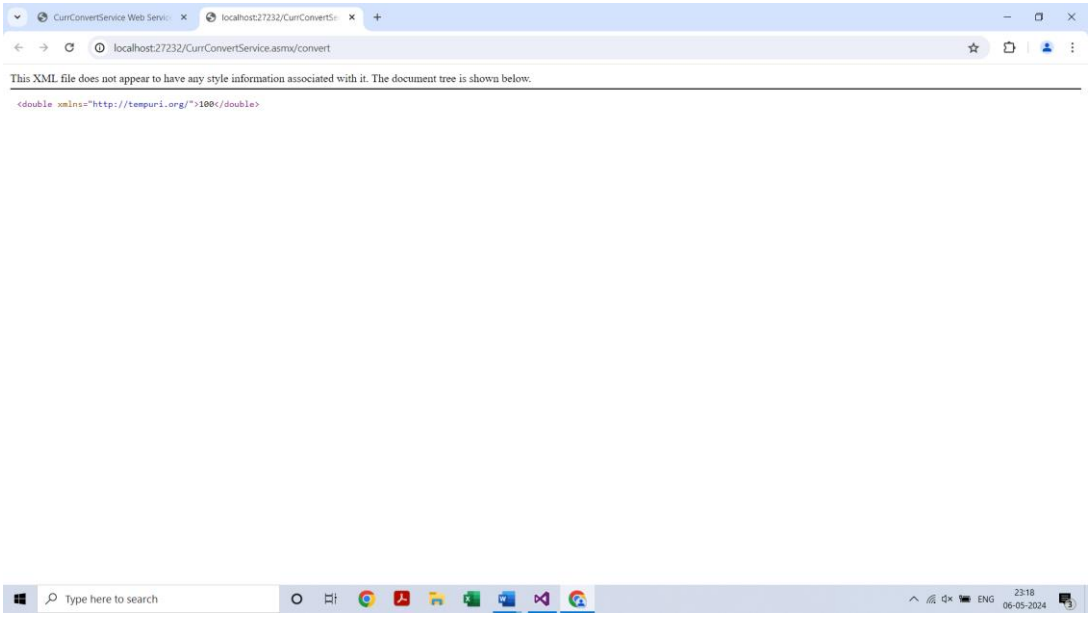
```
public double convert(double rs)
    {
        double d;
        d = rs / 83.4;
        return d;
    }
```



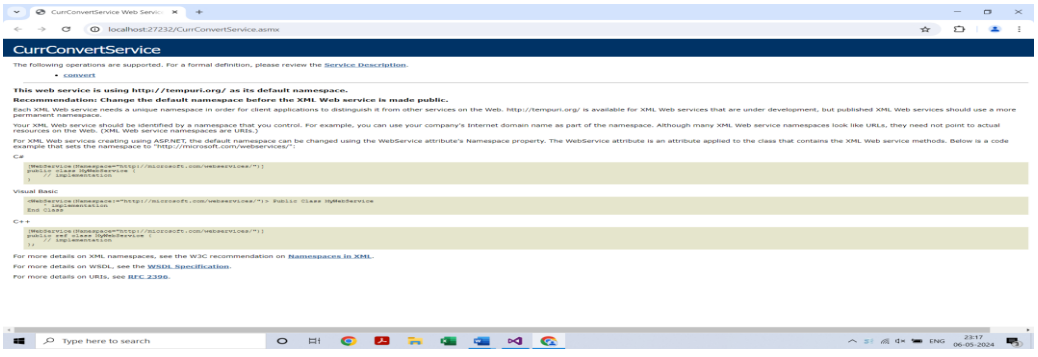
Save the changes. And run the project. A window will open in browser and that is the web service

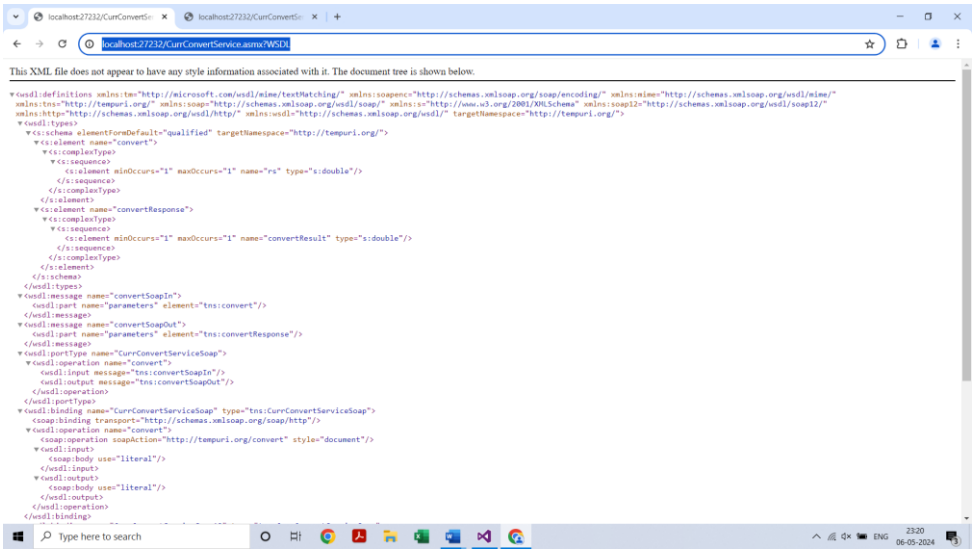






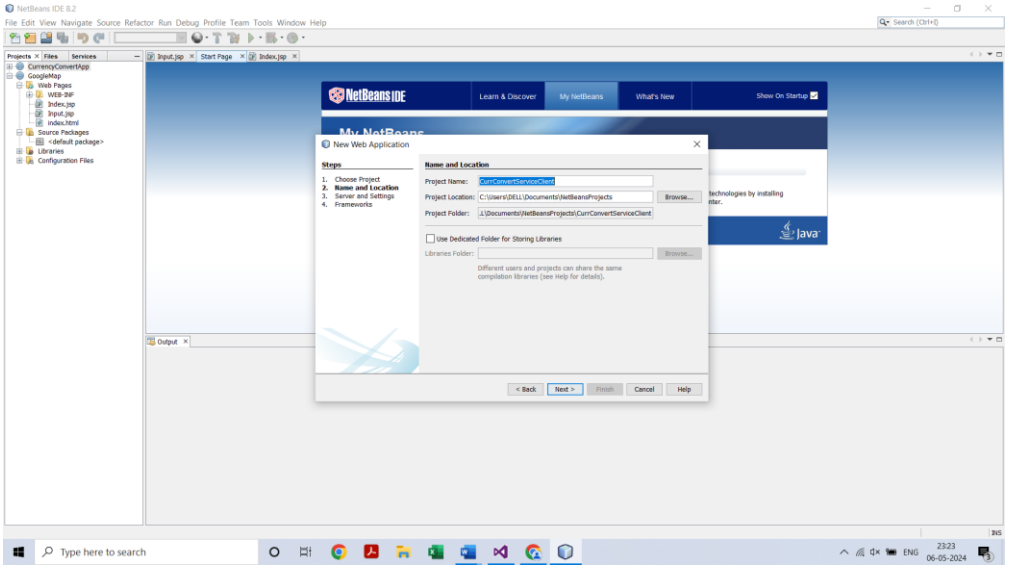
Now click on Service Description option. Now a new window will be opened. The link is selected and copied. We will use this link in NetBeans to consume this service





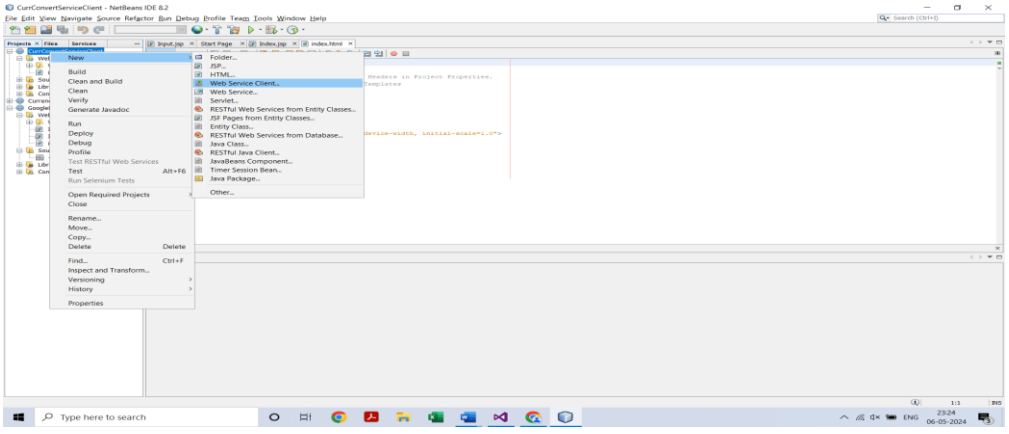
Now NetBeans is opened.

To Create a Web Application with name CurrConvertServiceClient. Next -> Finish

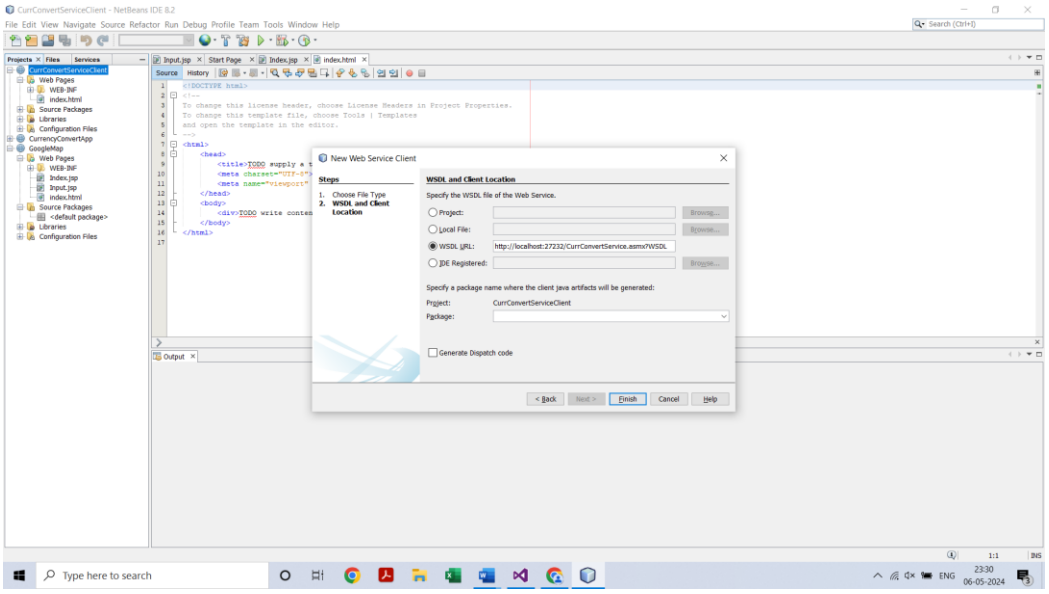


Now a Web Service Client is created.

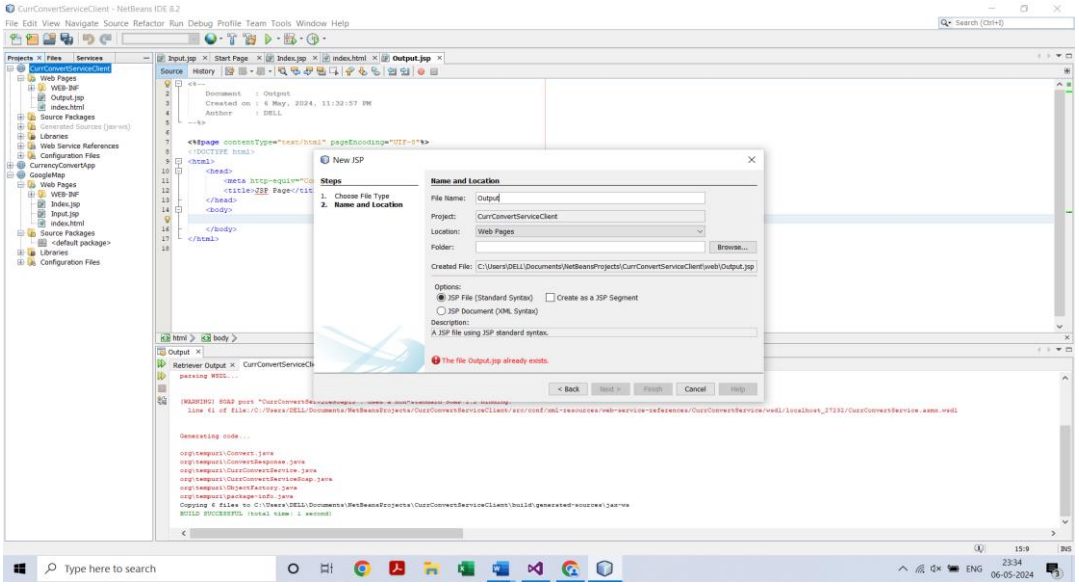
Right click on CurrConvertServiceClient -> New -> Web Service Client.



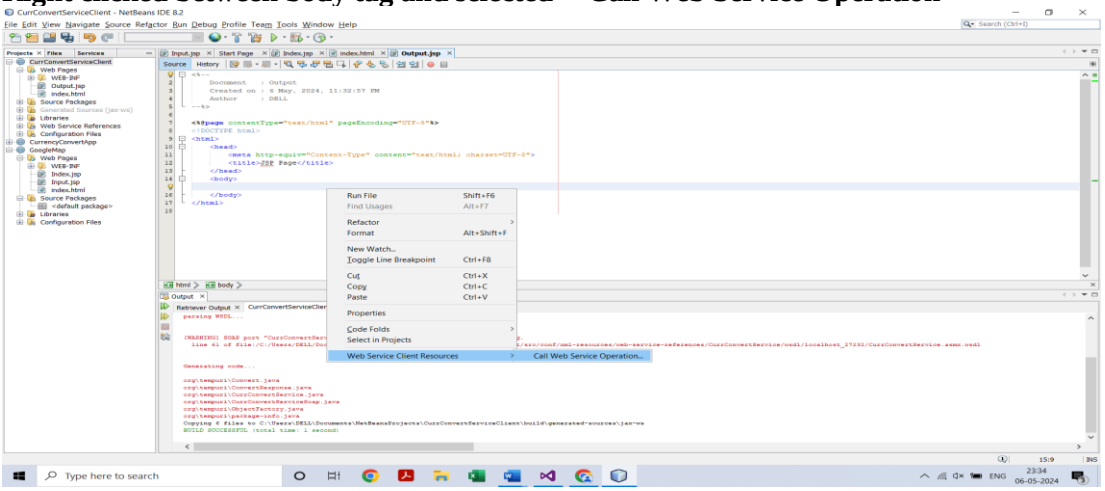
WSDL URL is selected and pasted the link that is copied from browser on run of Visual Studio. After that click on Finish button.

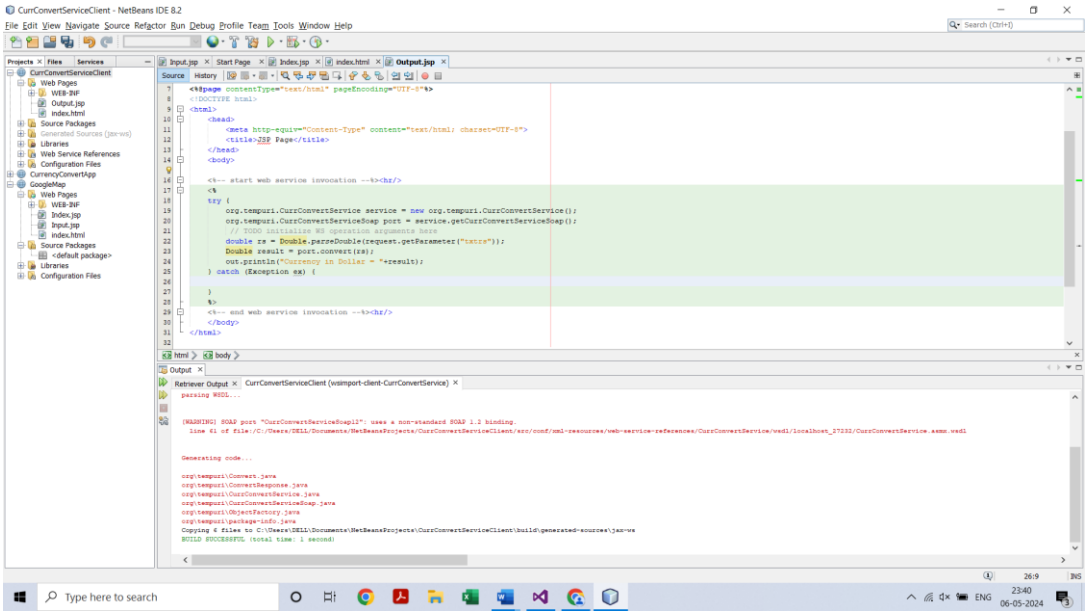


Now a JSP page is created. Right click on CurrConvertServiceClient -> New -> JSP



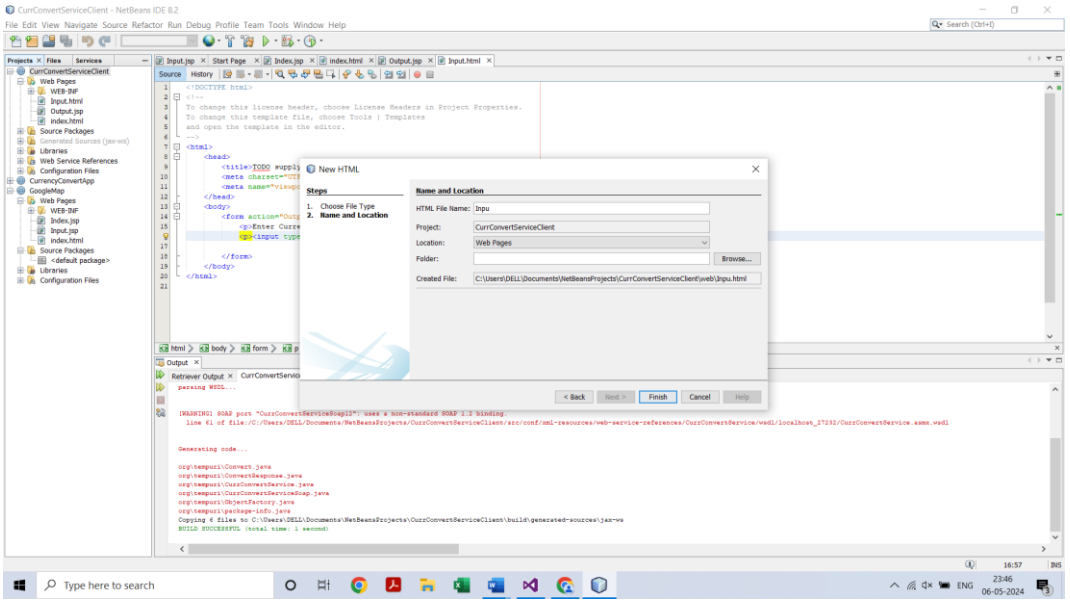
Right clicked between body tag and selected-> Call Web Service Operation

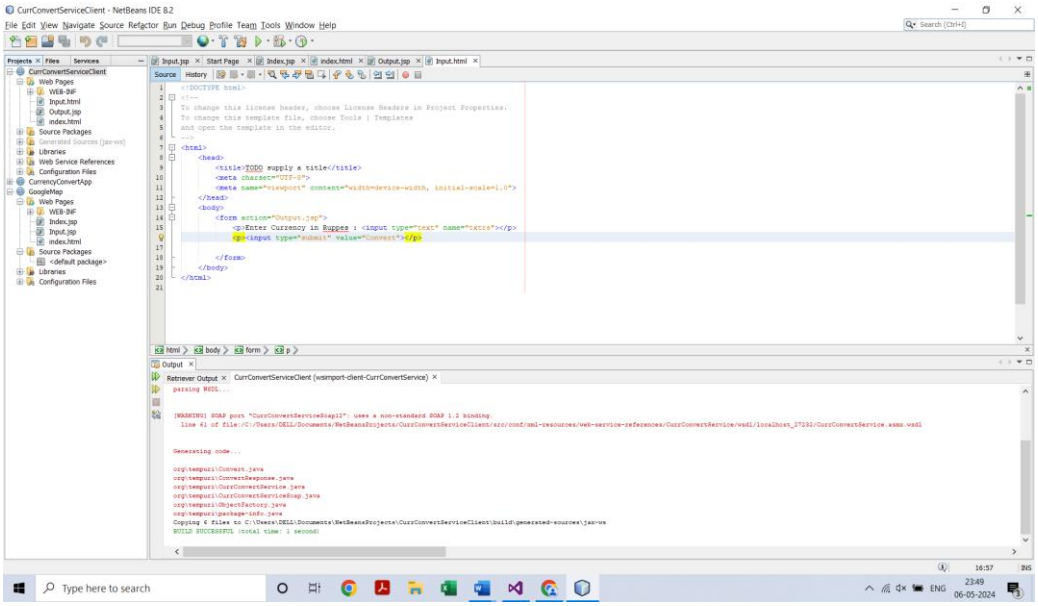




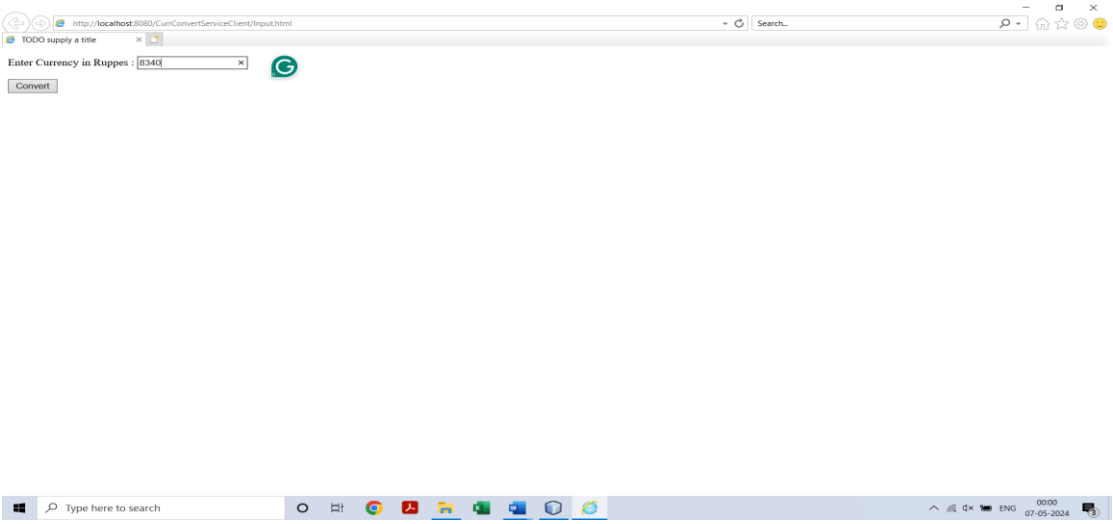
Now Input.html file of CurrConvertServiceClient project is created and opened and the contents of body tag is replaced with following code. And then saved it.

```
<form action="Output.jsp">
        <p>Enter Currency in Ruppes :  <input type="text" name="txtrs"></p>
        <p><input type="submit" value="Convert"></p>
</form>
```





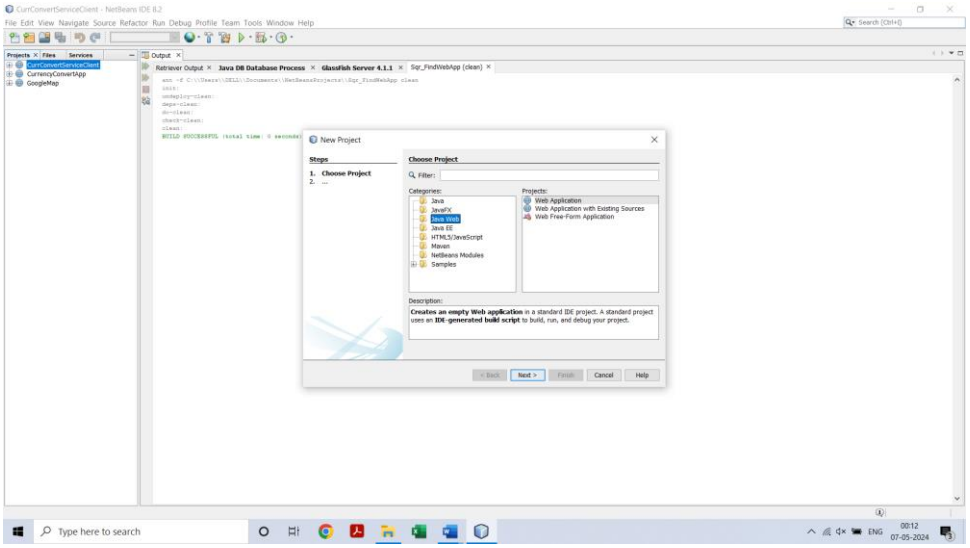
Now the web application is run.

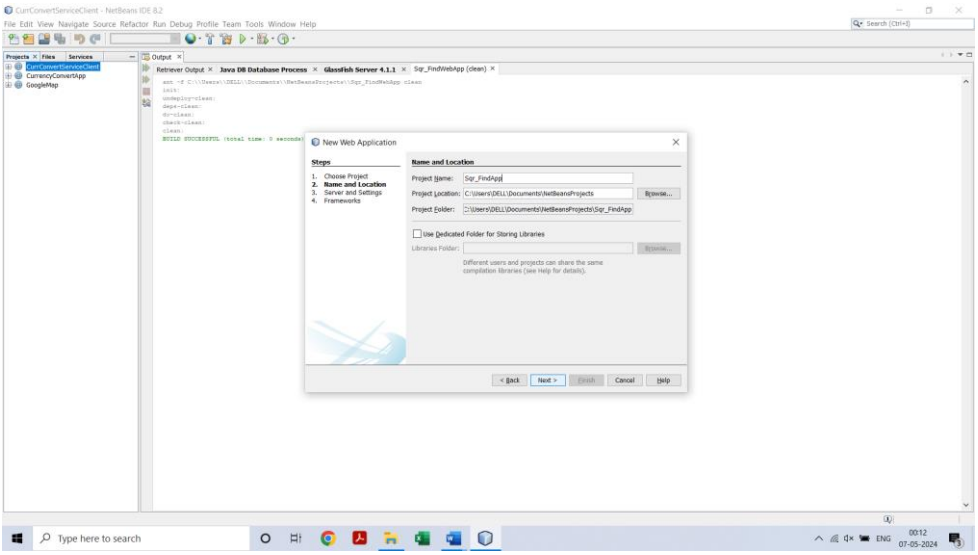


Practical-2

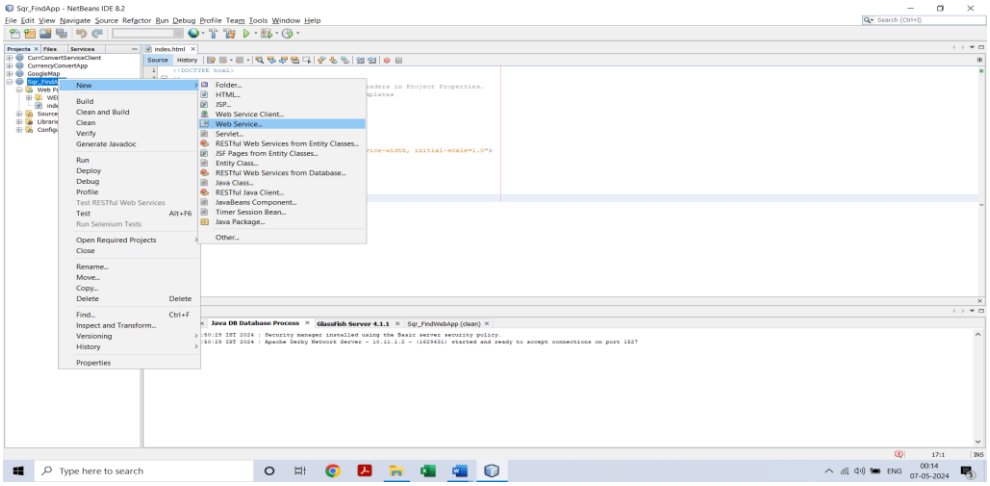
Create a Simple SOAP service to find Square of a Number

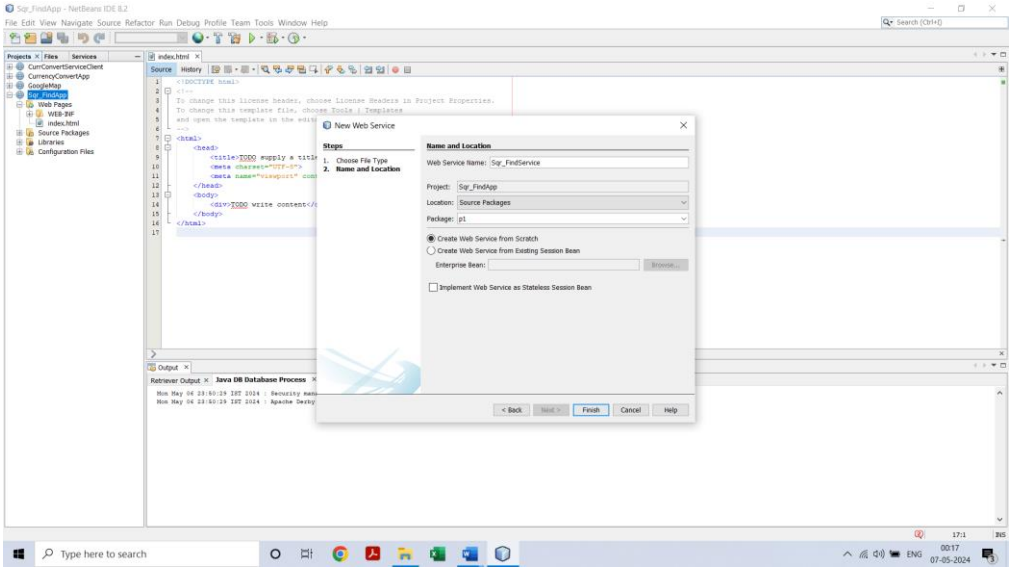
Step1 : To create Web Service through Net Beans



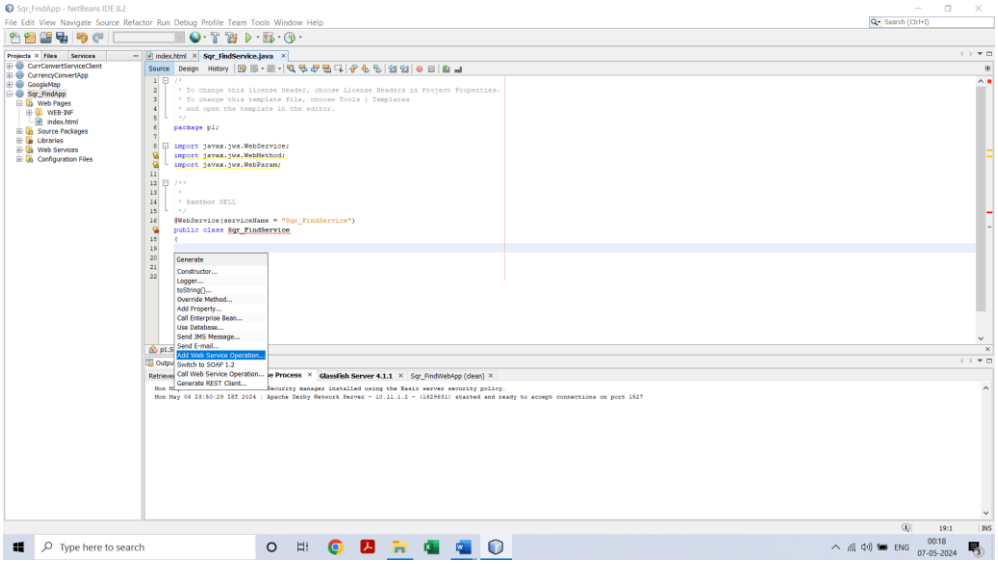


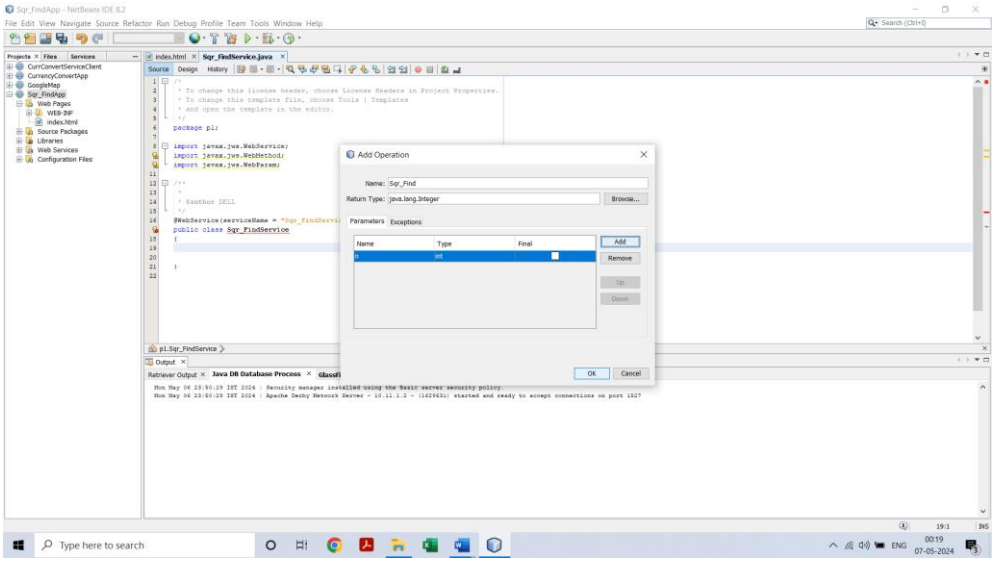
Step 2: To create Web Service " Sqr_FindService"





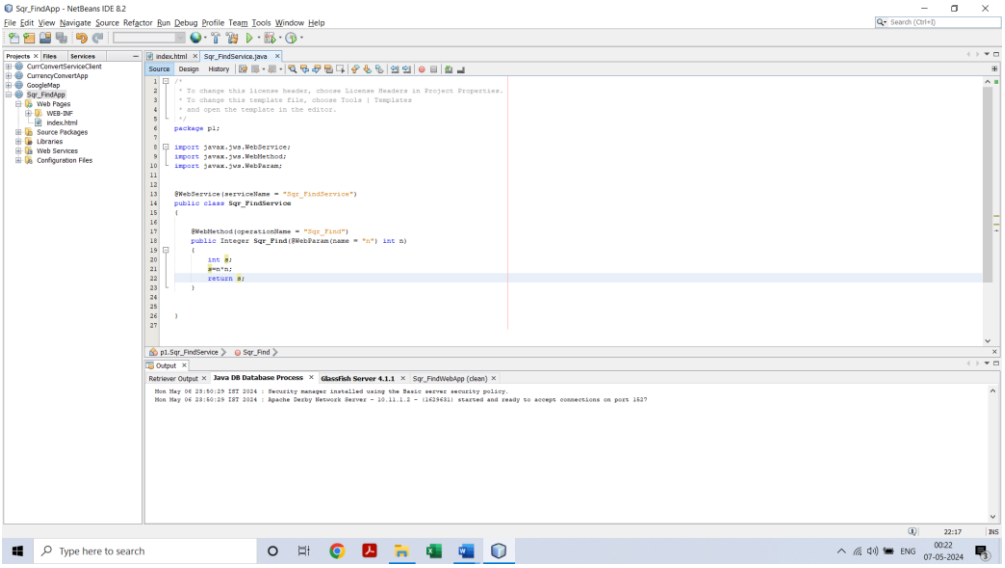
Step 3 : to Add Operation “Sqr_Find” to Web Service





Here Sqr_Find is a method having one parameter as n

Step 4 : After adding code of Operation to web Service

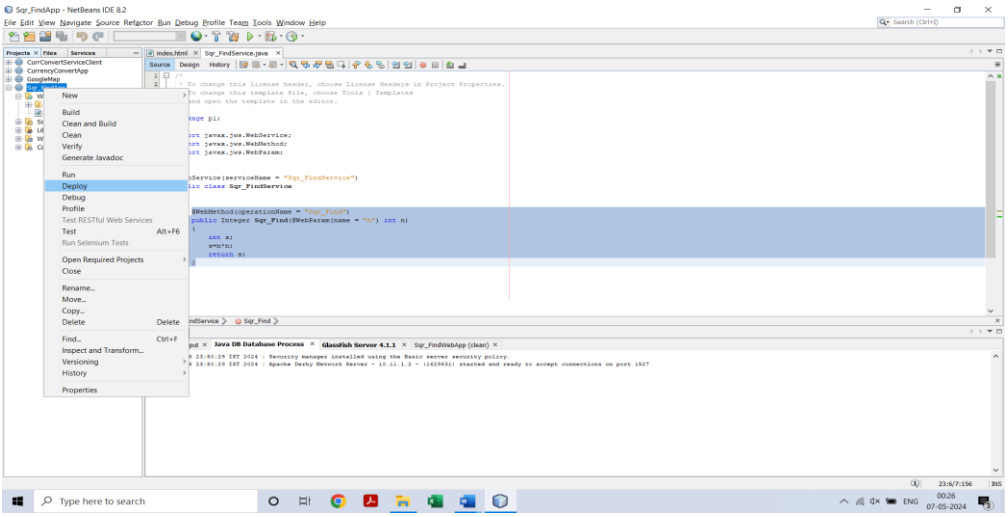


Code of Operation :

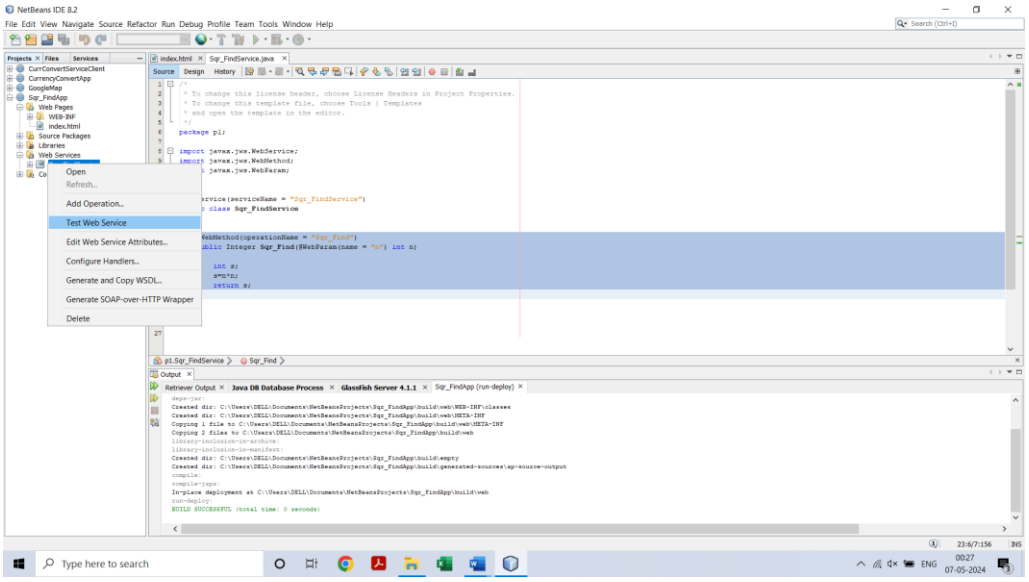
```
@WebMethod(operationName = "Sqr_Find")
    public Integer Sqr_Find(@WebParam(name = "n") int n)
    {
        int s;
        s=n*n;
        return s;
    }
```

Step 5 : Now to deploy the web service and Test

iii) Deploy

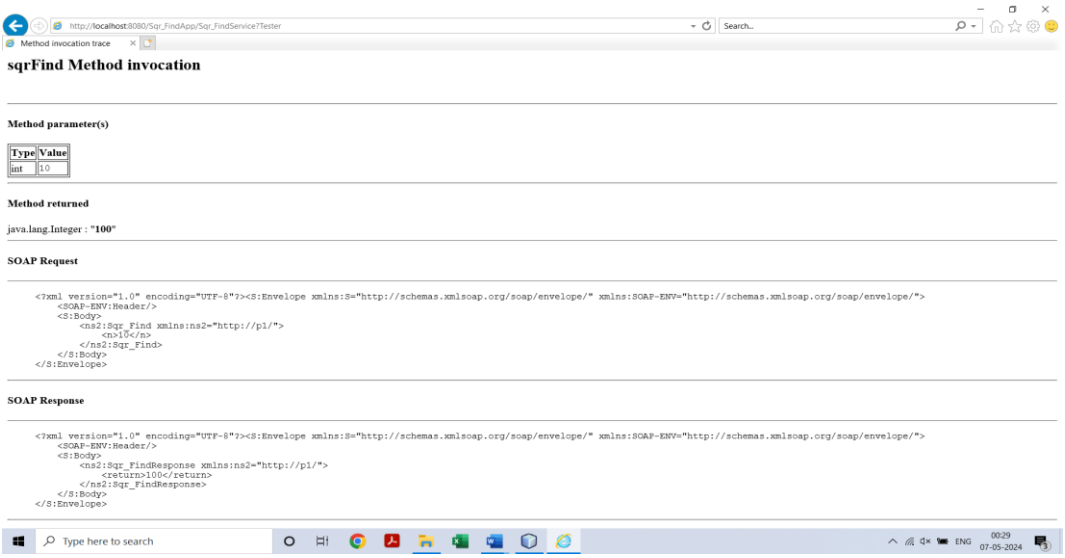


iv) Test Web Service

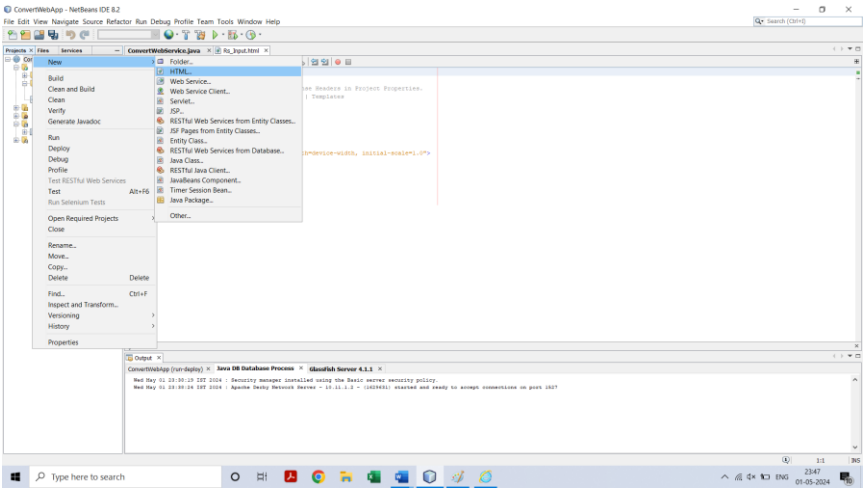


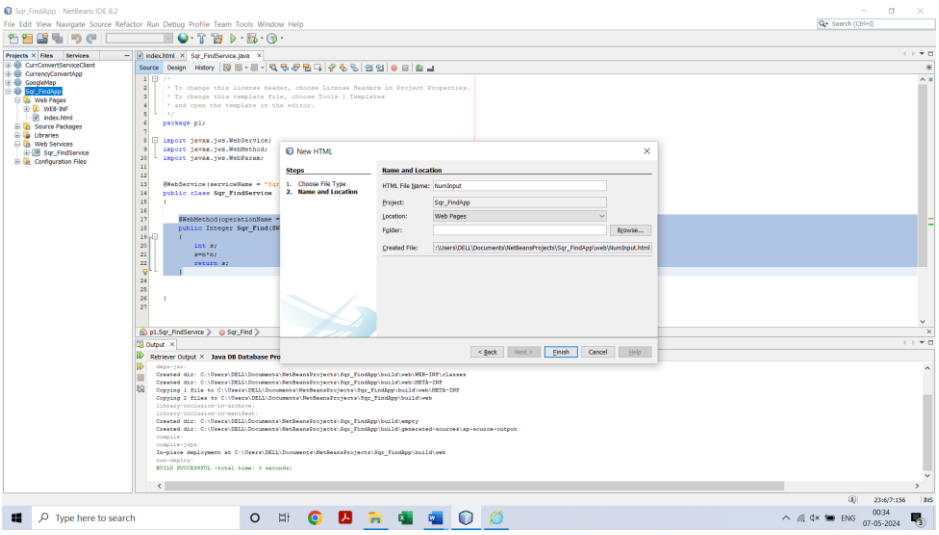
After testing

Successfully Deployed and Tested



Step 6: Now to take user input Create NumInput.html page



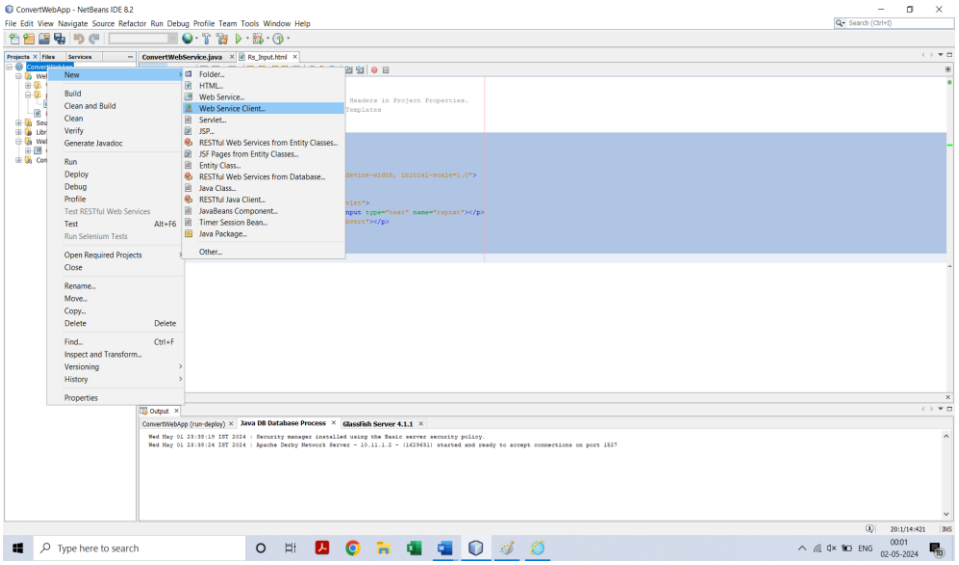


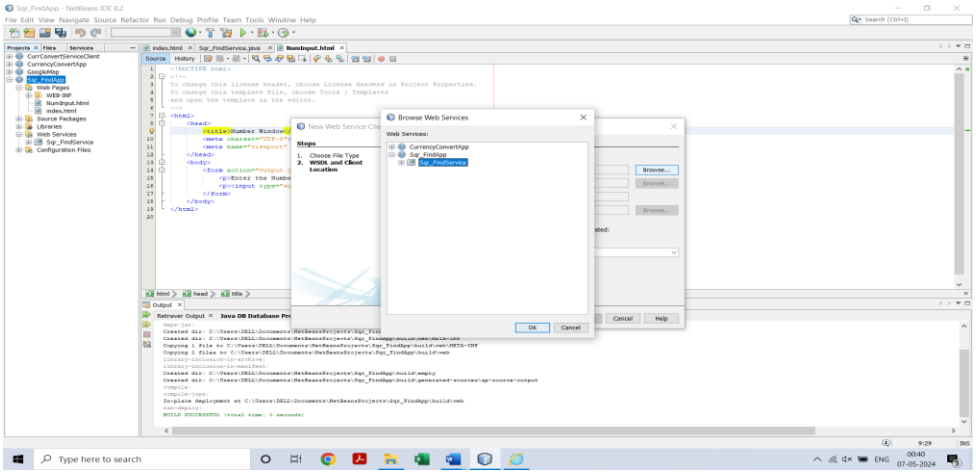
Now insert the code to create html page :

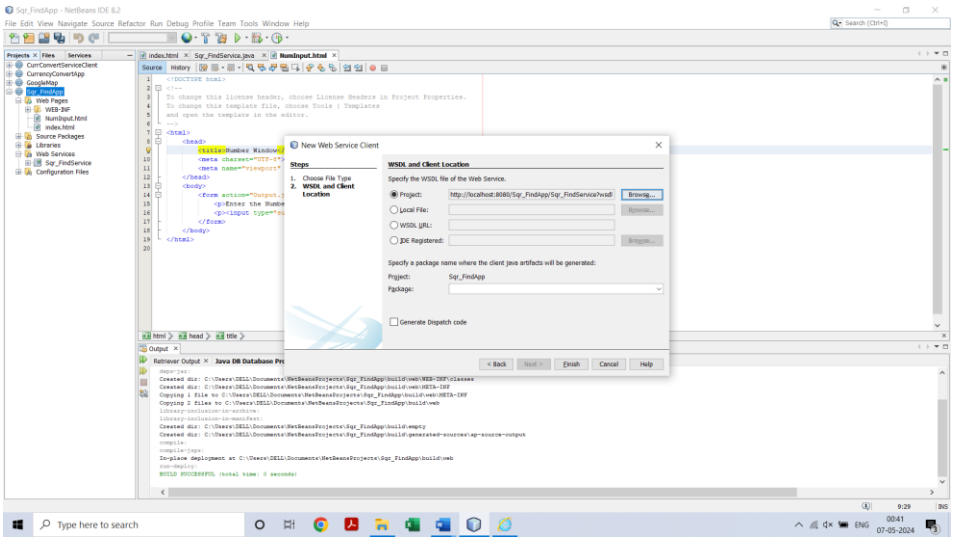
```
<html>
  <head>
    <title>Number Window</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <form action="Output.jsp">
      <p>Enter the Number : <input type="text" name="txtnum"></p>
      <p><input type="submit" value="Find_SQR"></p>
    </form>
  </body>
   </html>
```

Step 7 :

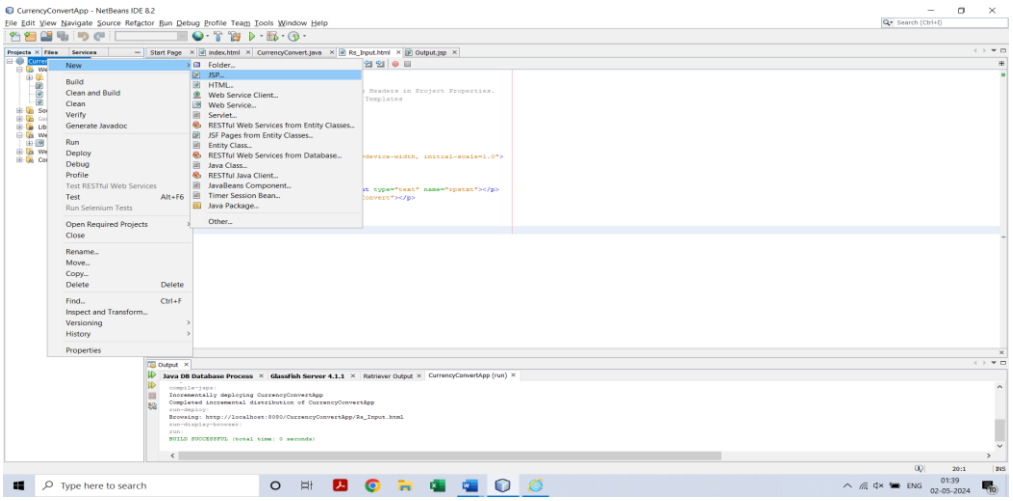
Now to create web service client, for same right click on project and select new then select web service client you will get the following screen:

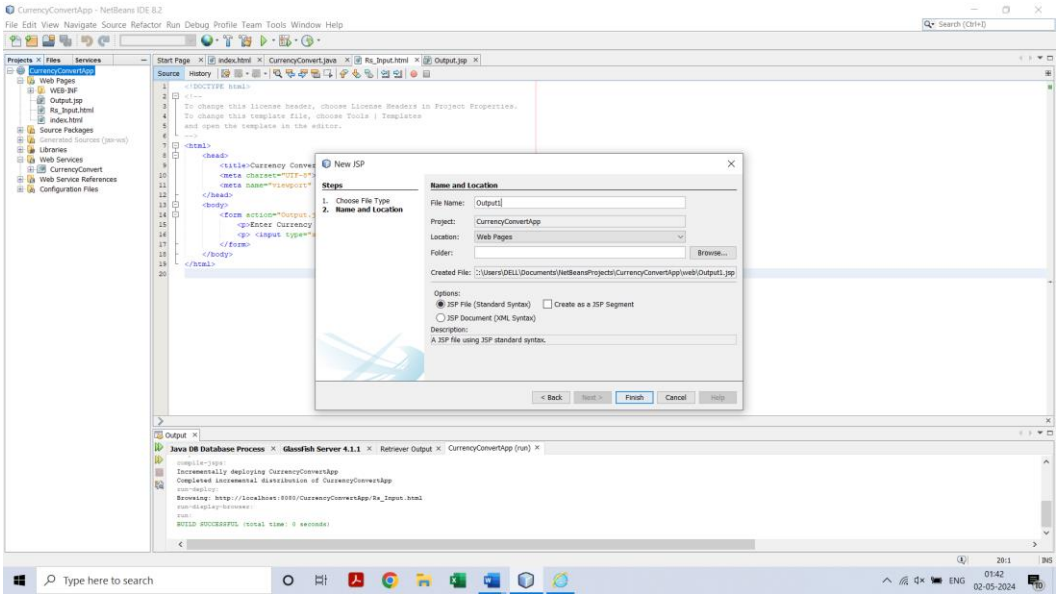




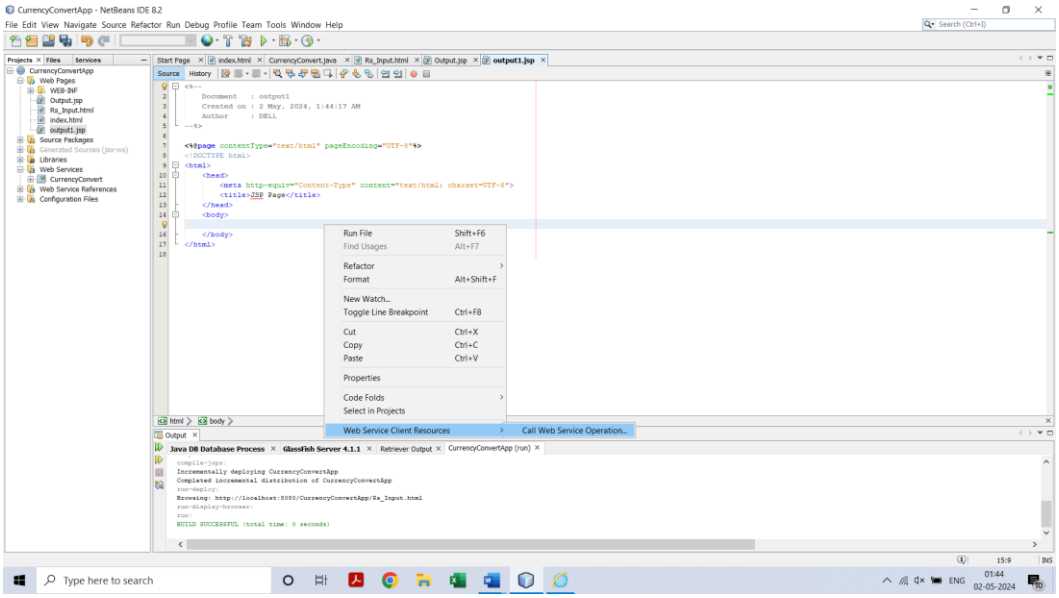


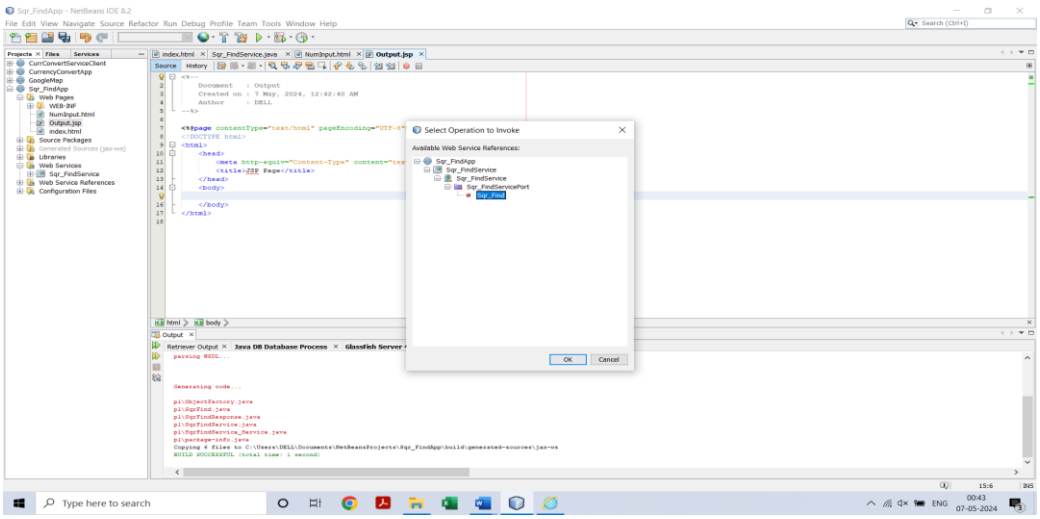
Step 8: Now finally to create Output.jsp page for output





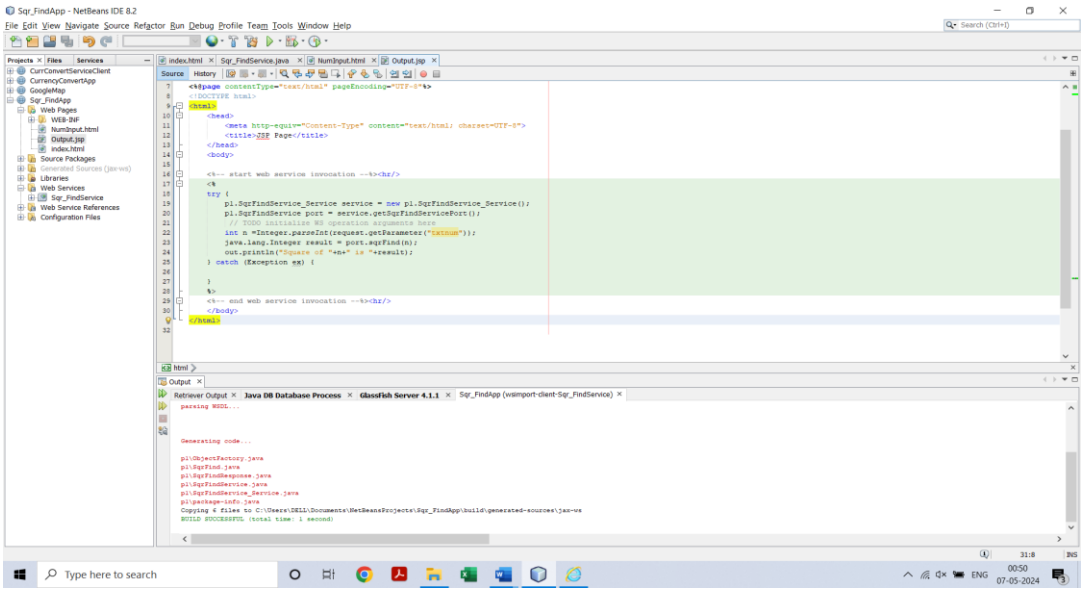
Step 9: After creating the page code is to be added



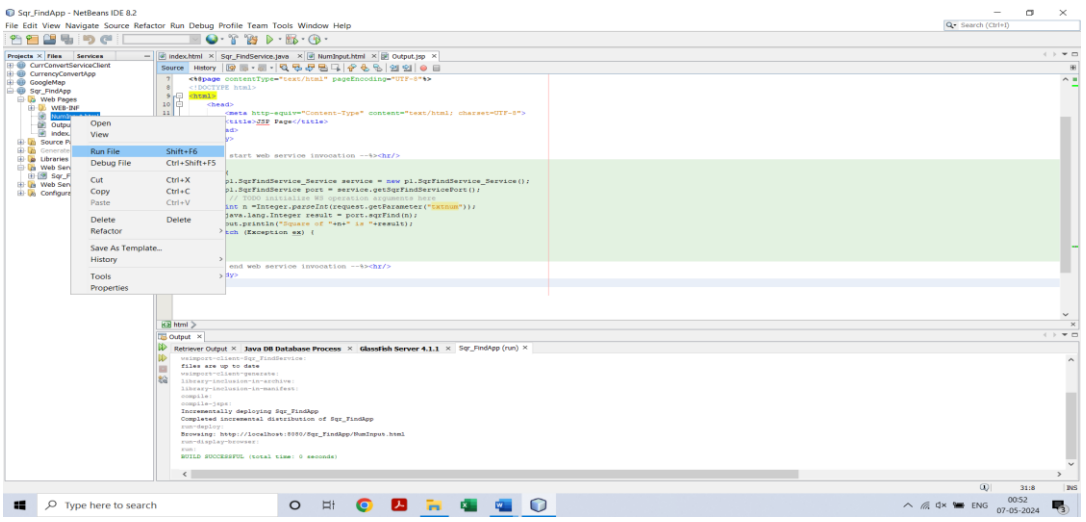


Code to Output.jsp page

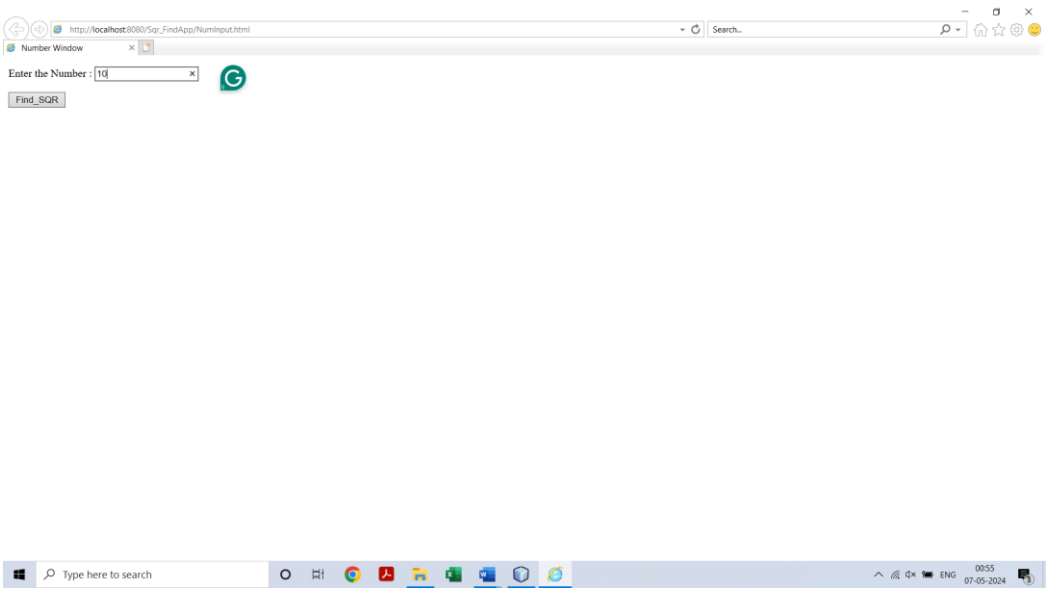
```
<%
   try {
         p1.SqrFindService_Service service = new p1.SqrFindService_Service();
         p1.SqrFindService port = service.getSqrFindServicePort();

         int n =Integer.parseInt(request.getParameter("txtnum"));
         java.lang.Integer result = port.sqrFind(n);
         out.println("Square of "+n+" is "+result);
   }
catch (Exception ex)
 {

   }
    %>
```

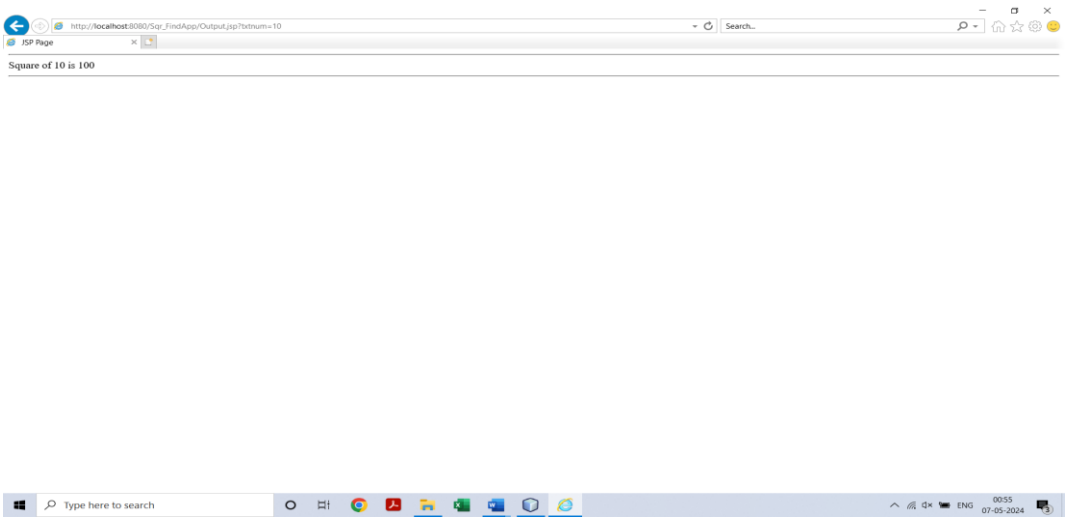


Step 10 : Run the application. The following output will be generated



Output :

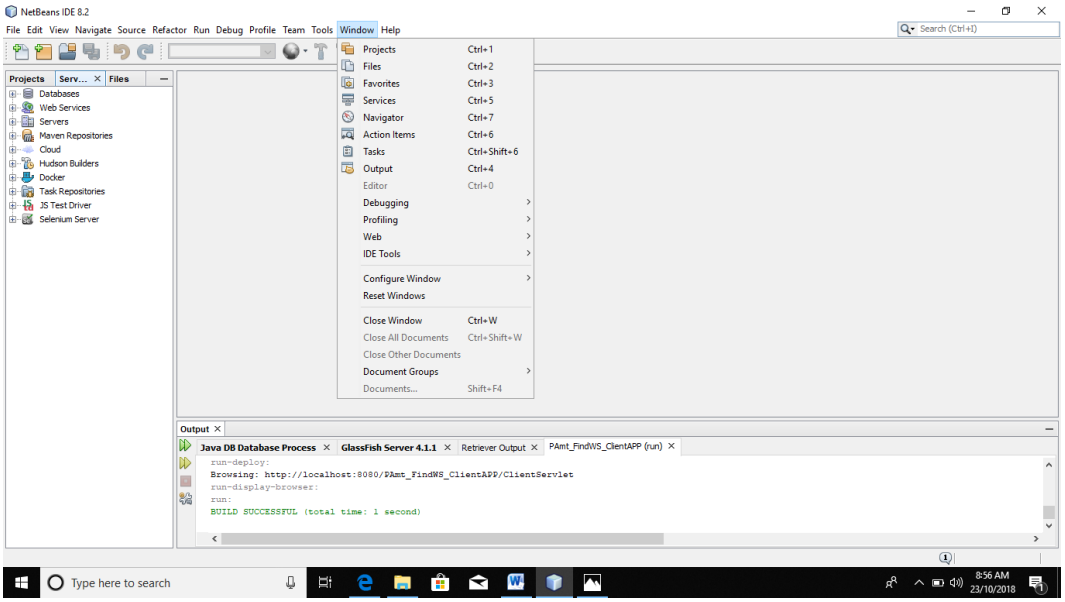

http://localhost:8080/Sqr_FindApp/NumInput.html
Search...
Number Window
Enter the Number : 10
Find_SQR
Type here to search
ENG
00:55
07-05-2024


http://localhost:8080/Sqr_FindApp/Output.jsp?txtnum=10
Search...
JSP Page
Square of 10 is 100
Type here to search
ENG
00:55
07-05-2024

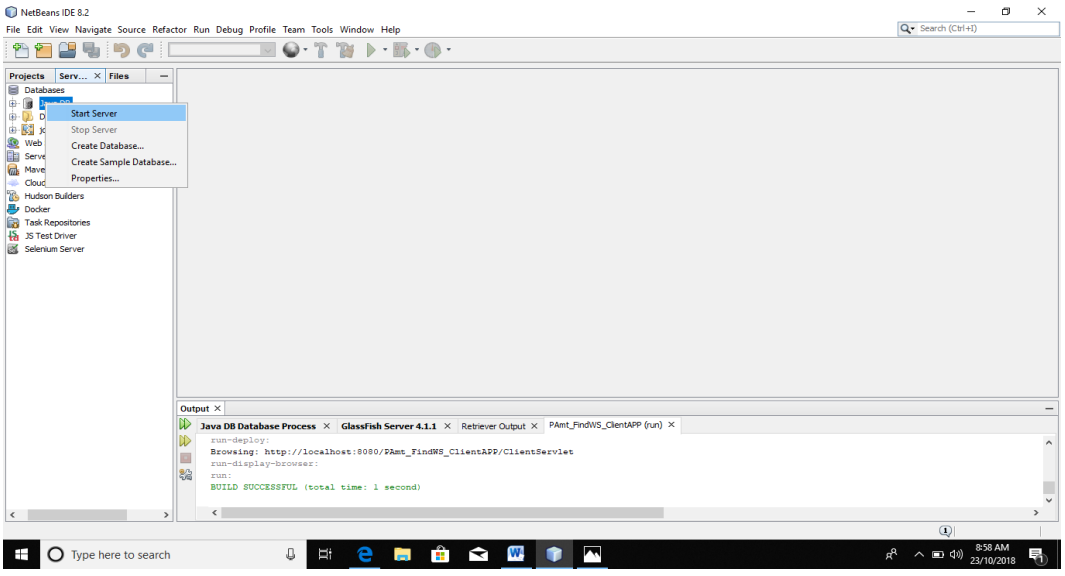
Practical-3

Create a Simple REST Service to demonstrate CRUD operations with "Student" database

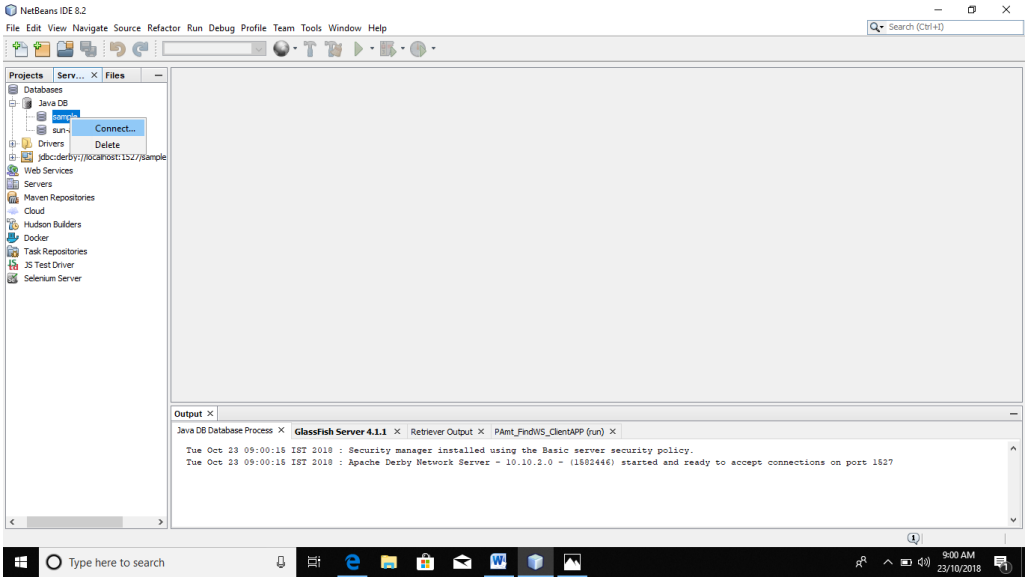
1. **Click on Window menu and click on Projects, Files & Services to open it.**



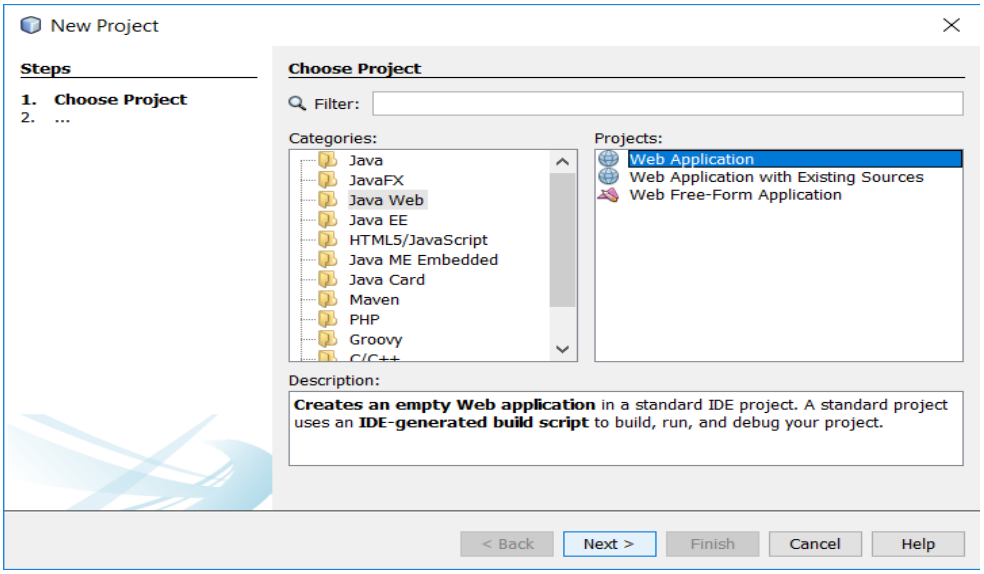
2. **Right click on Java DB and then click on Start Server to start the server .**

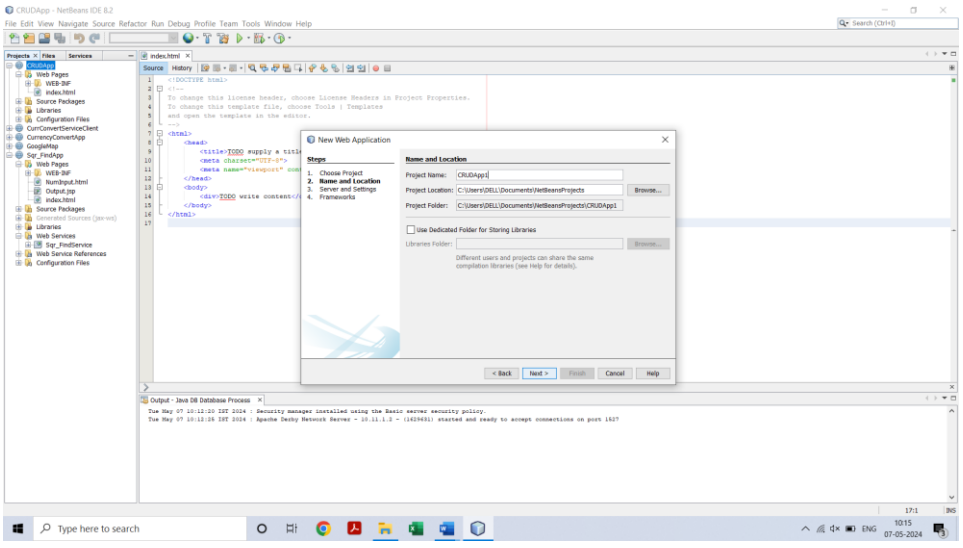


3. **Now expand Java DB and right click on sample and then click on connect to connect the sample database with server.**

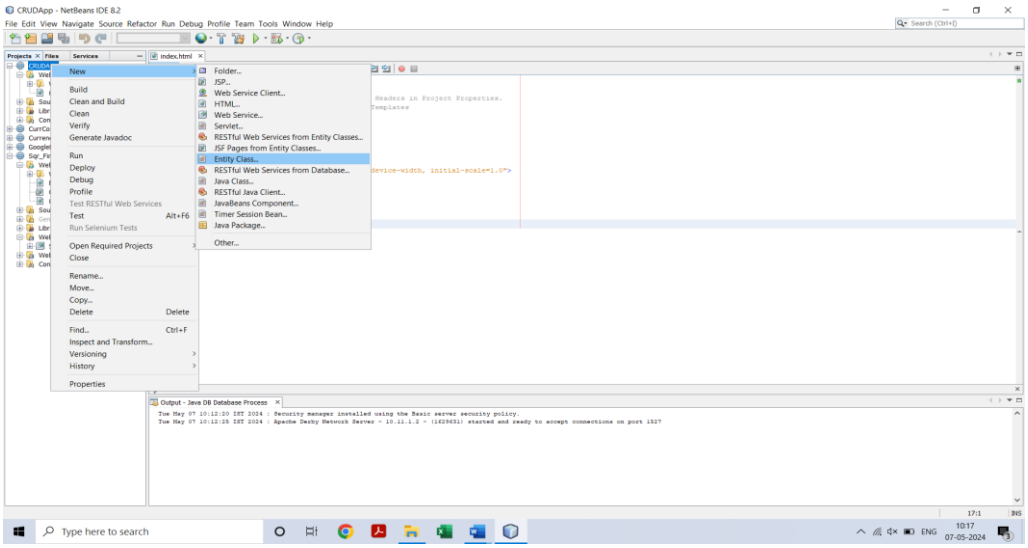


4. Now create a web application with the name CRUD_Operation. A window will open like following pic.

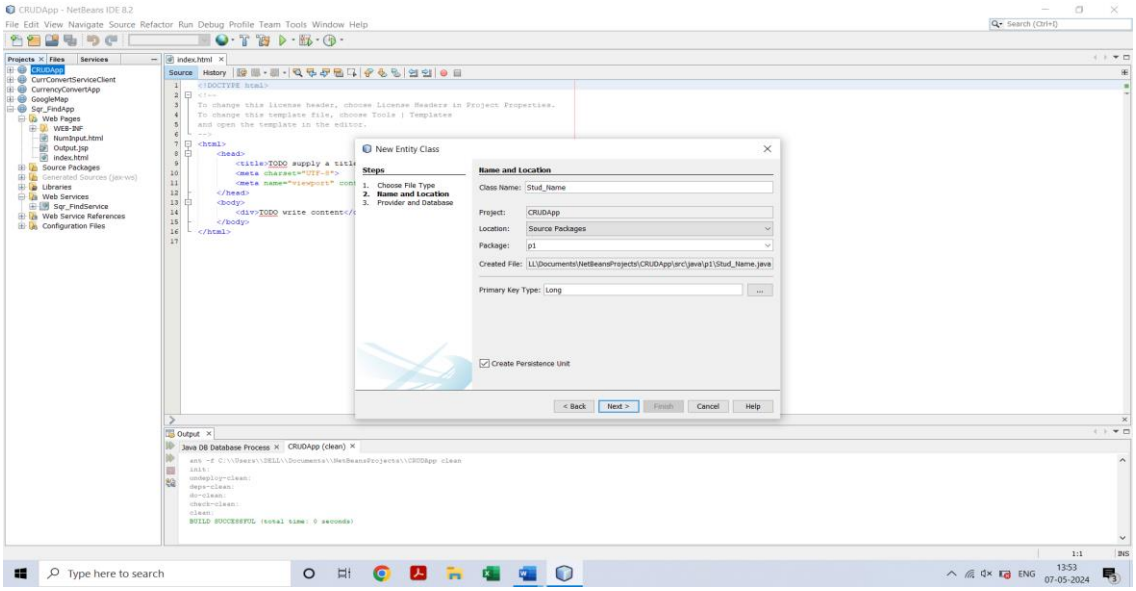


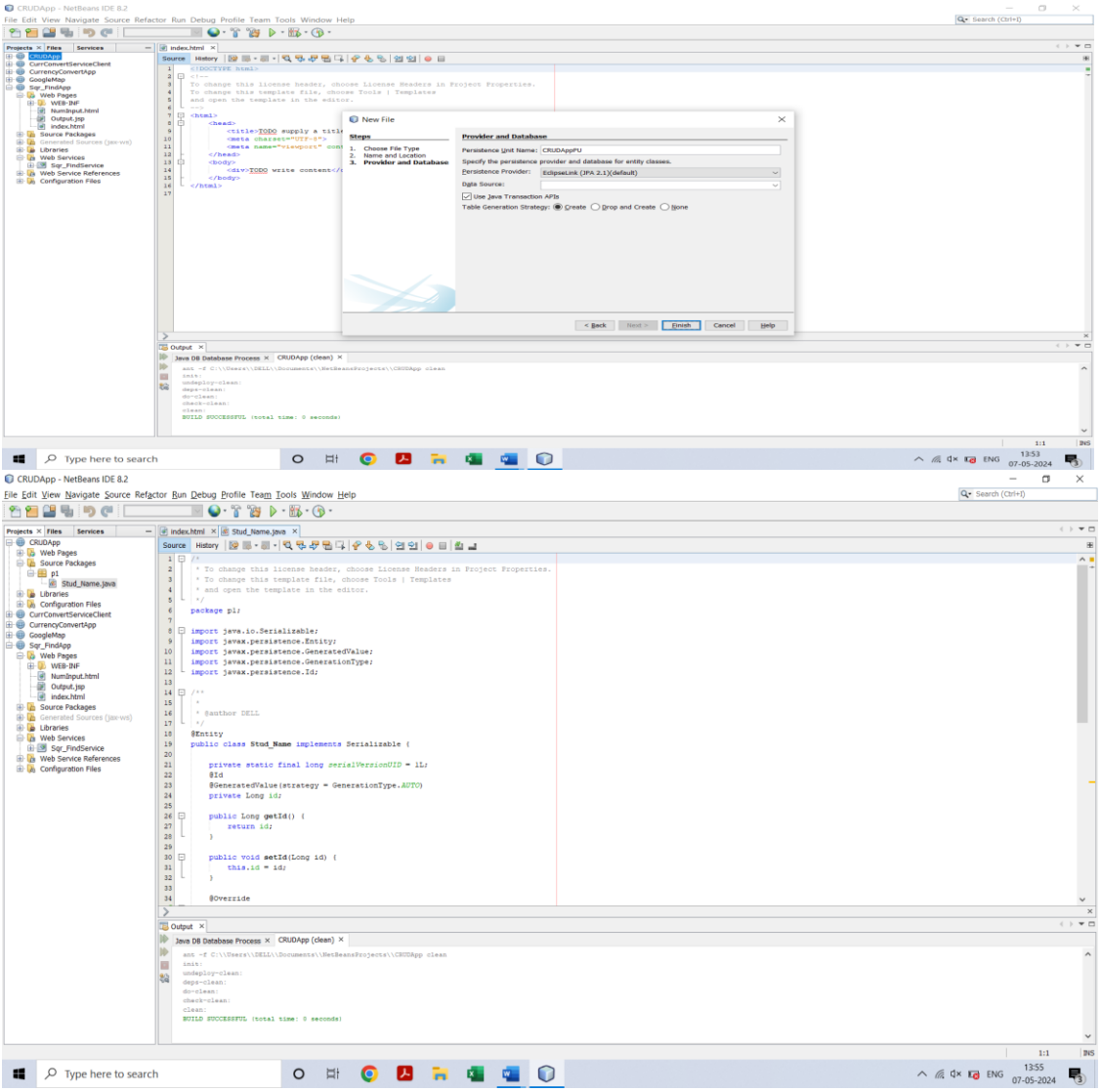


5. **Create an entity class. Right click on project name -> New -> Entity Class.**



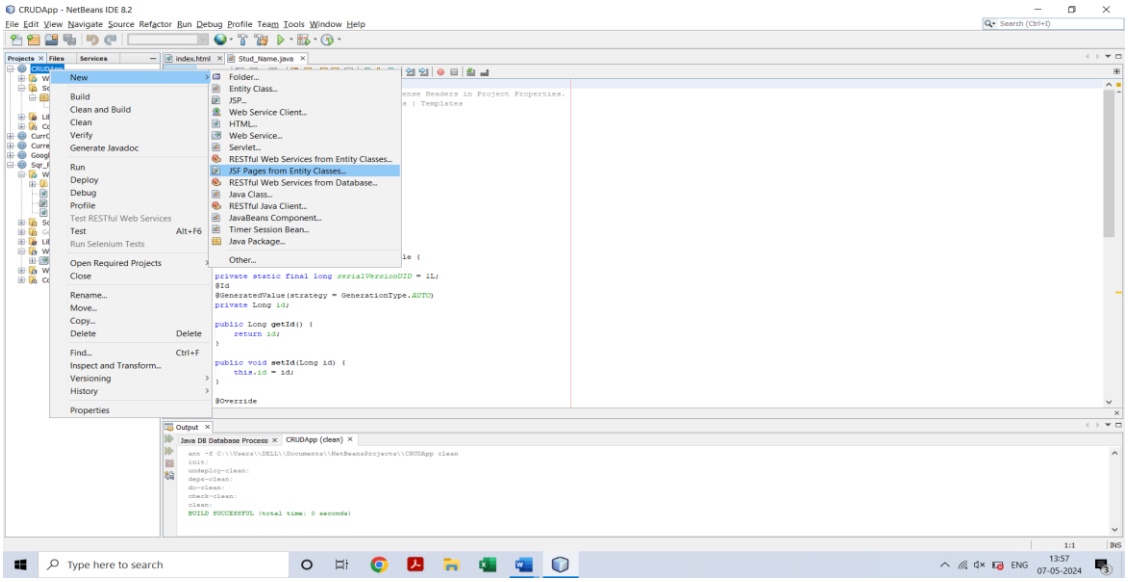
6. **A window will appear like bellow pic. Enter following data and click on Next ….**
   Class Name ->Stud_Name Package Name -> p1
7. **Click on finish.**

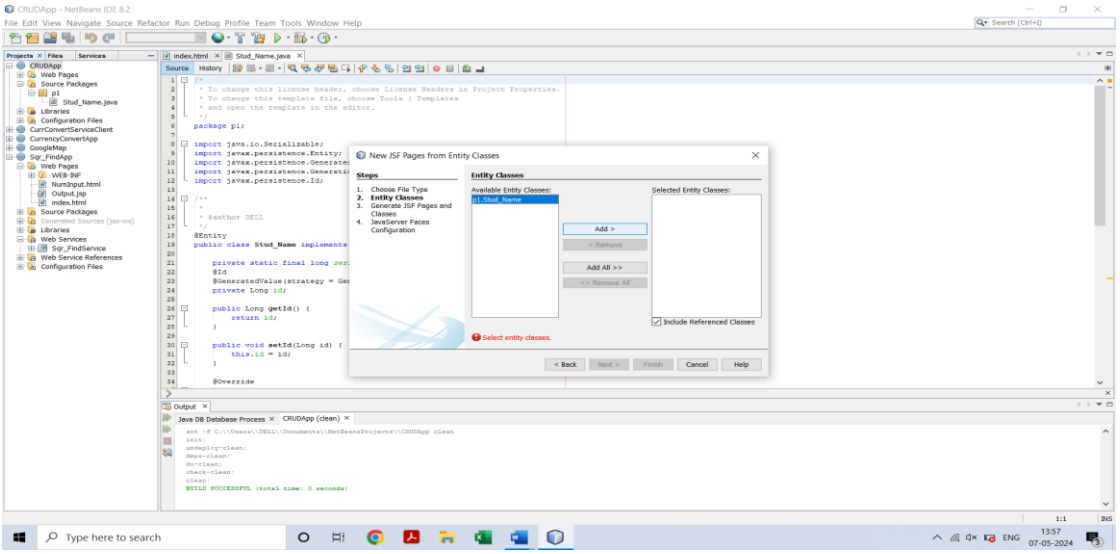


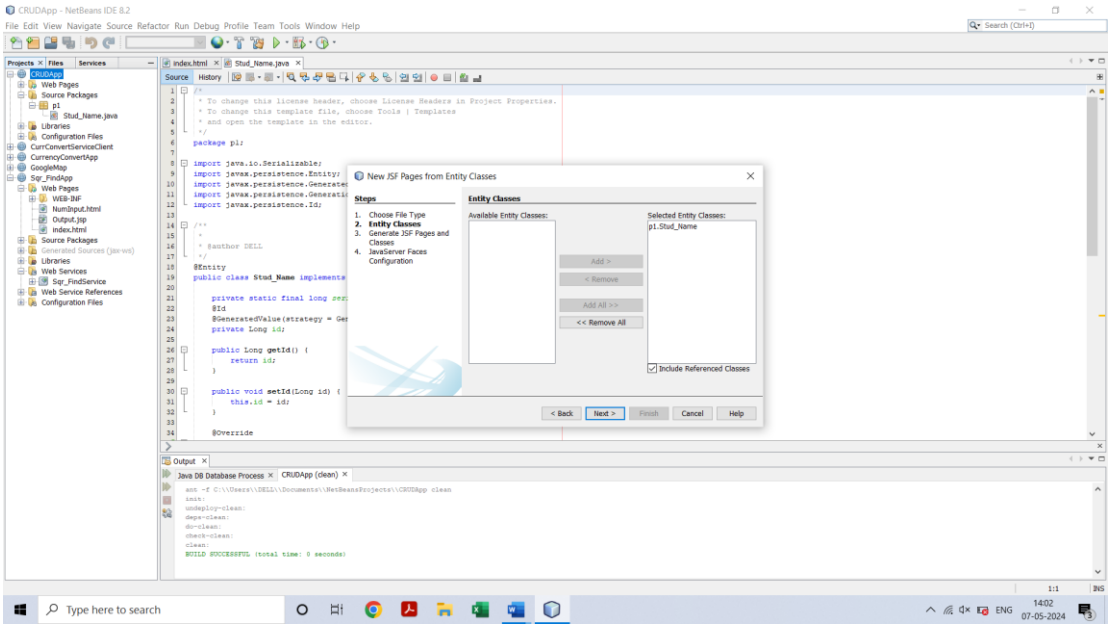


8. Right click on project name and create JSF Pages from Entity Classes.

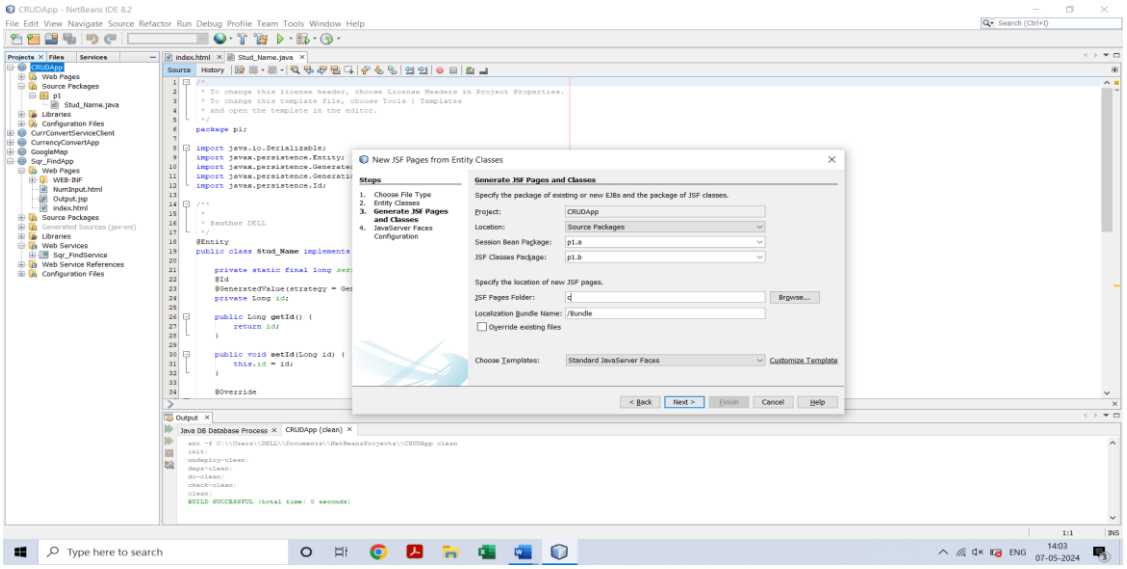
9. Select and click on Add button and then Next button on below



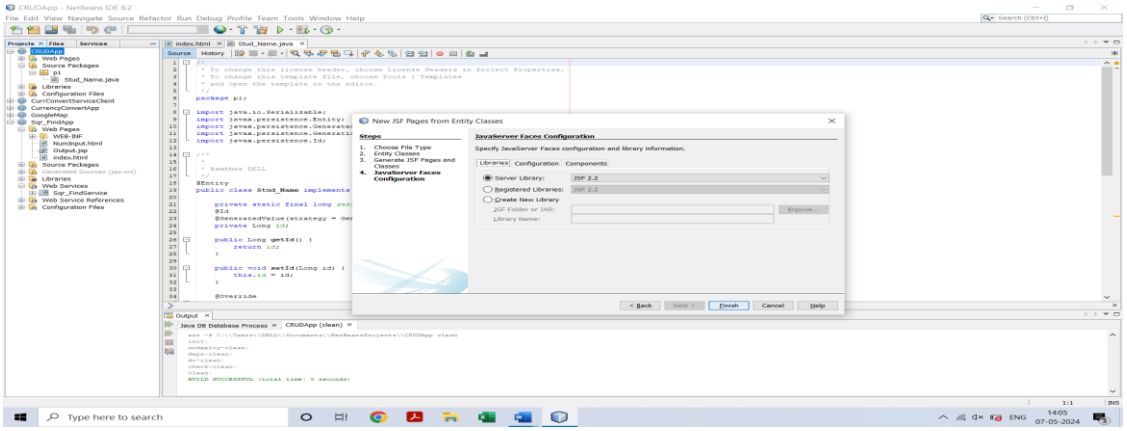




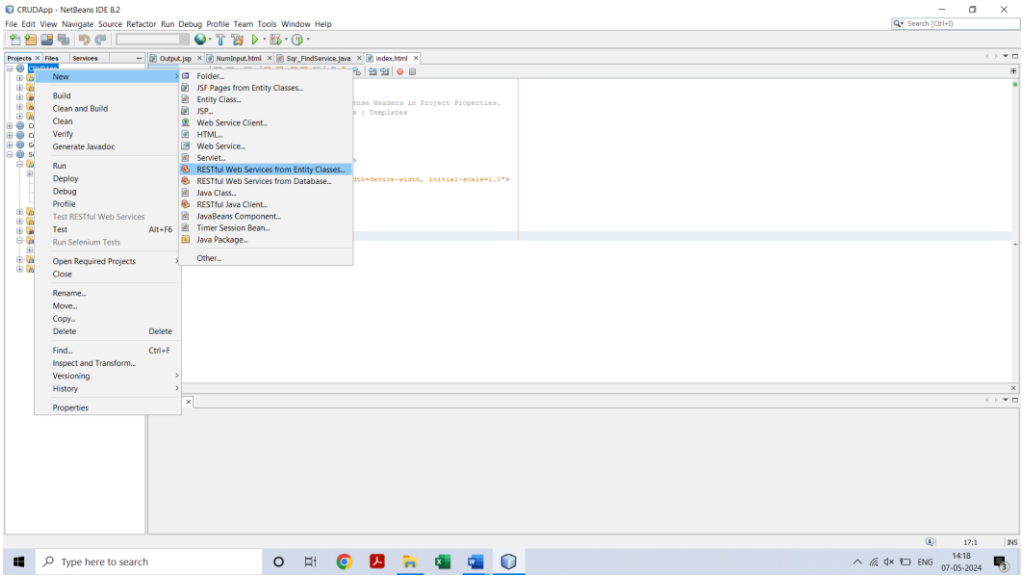
10. A window like below will appear on the screen. Enter the data into that window as entered in below pic and click on Next button.

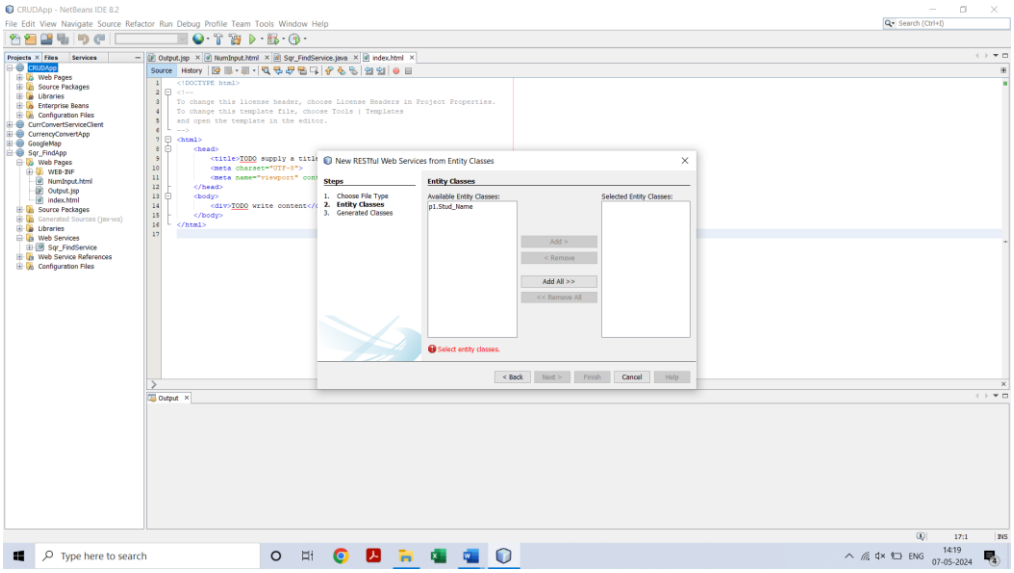


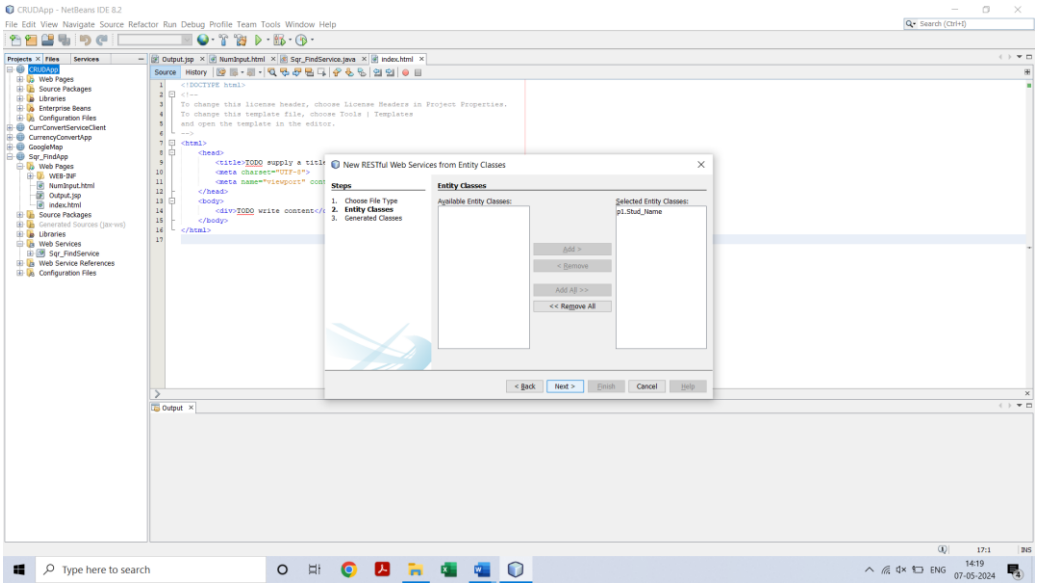
11. **Now click on Finish.**



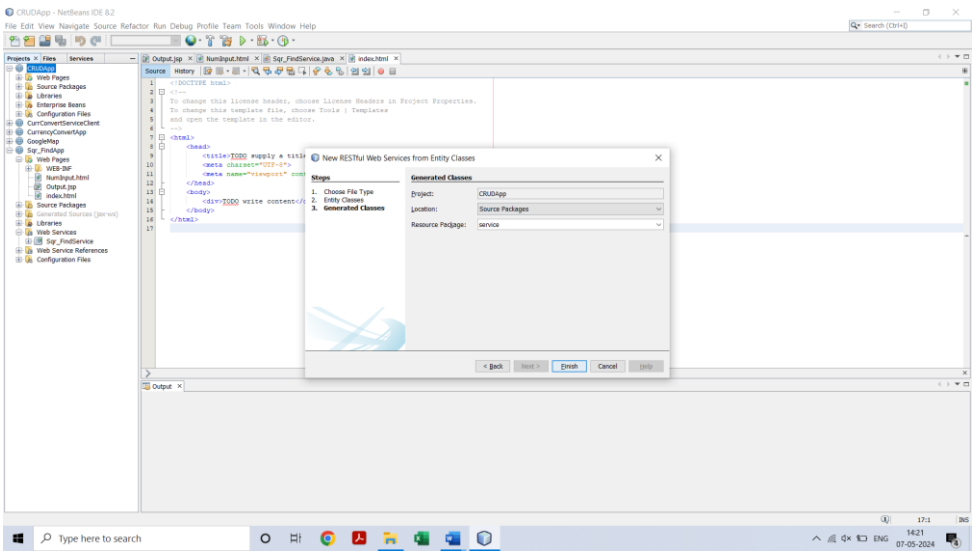
12. **Right click on project name and create RESTful Web Services from Entity Classes.**







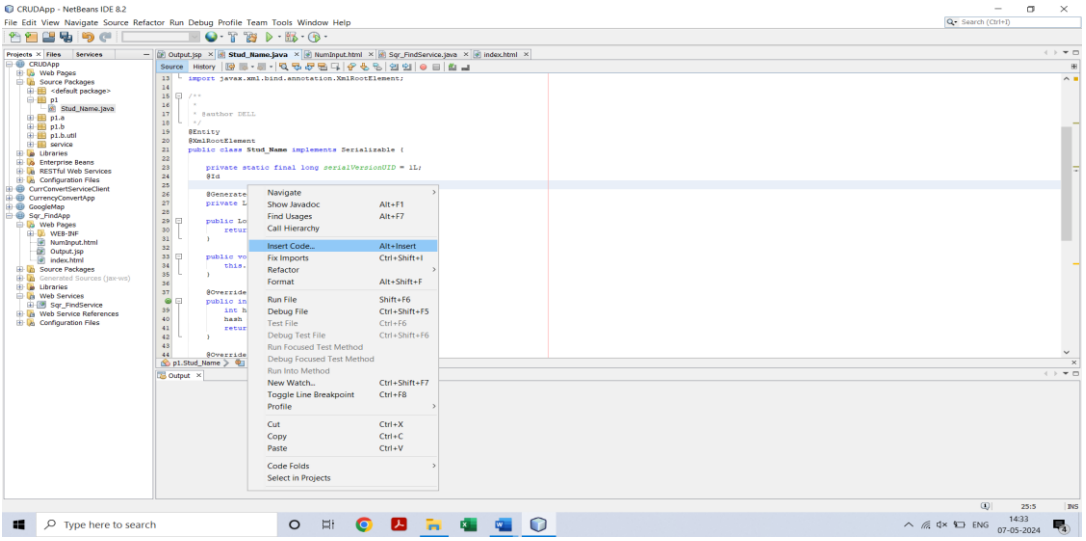
13. **Repeat step 9 and then it will go on next page. Then enter the p1.service in Resource Package and then click on Finish button.**

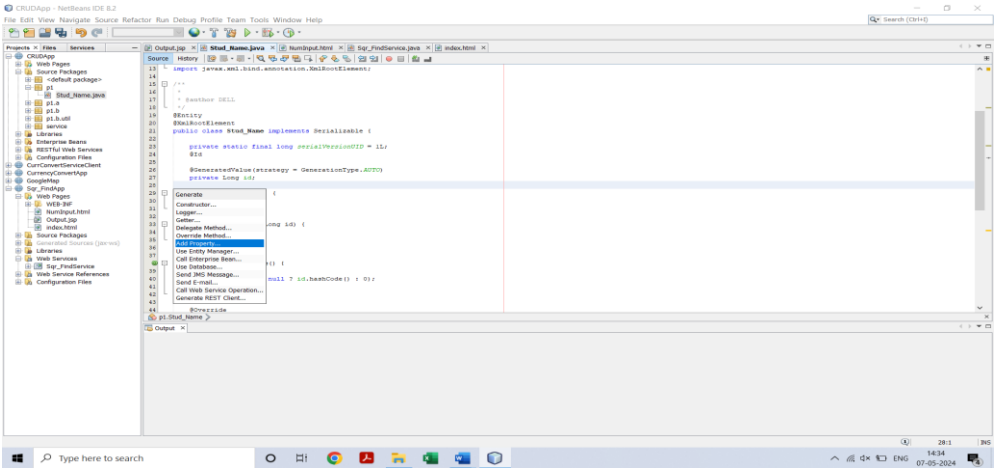


14.Now open Stud_Name.java file under p1 package.

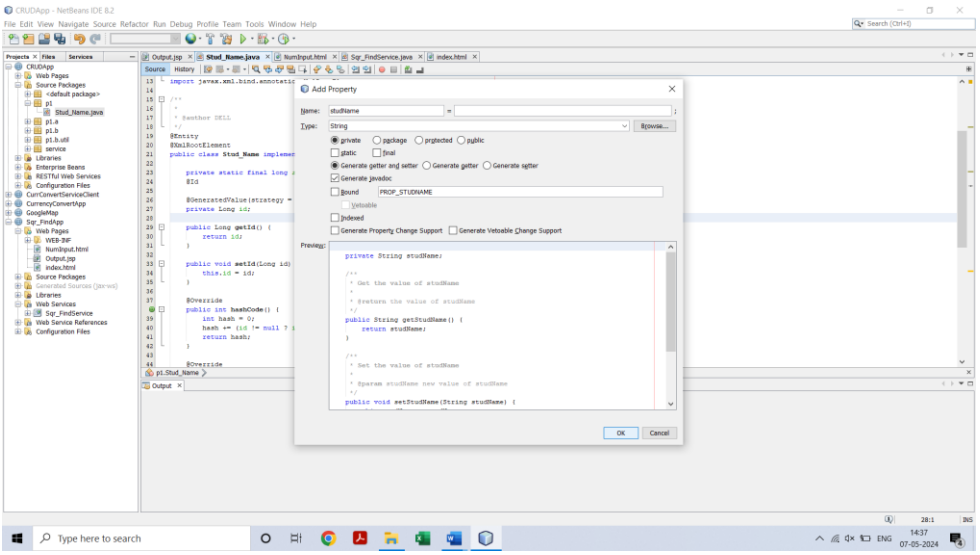
15. In this file at line number 24, do the right click and select Insert Code.

16. A new list will appear. Click on Add Property

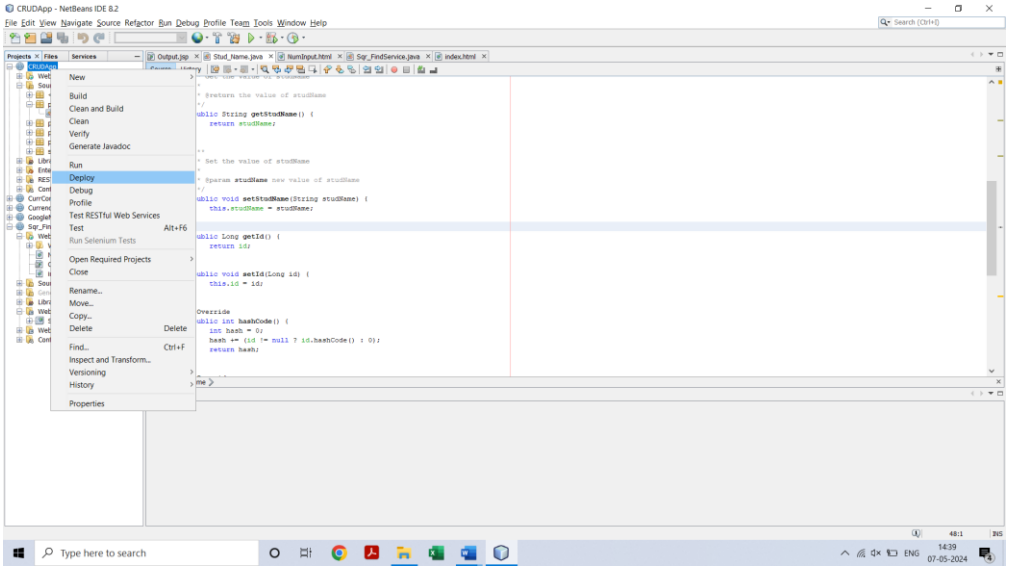




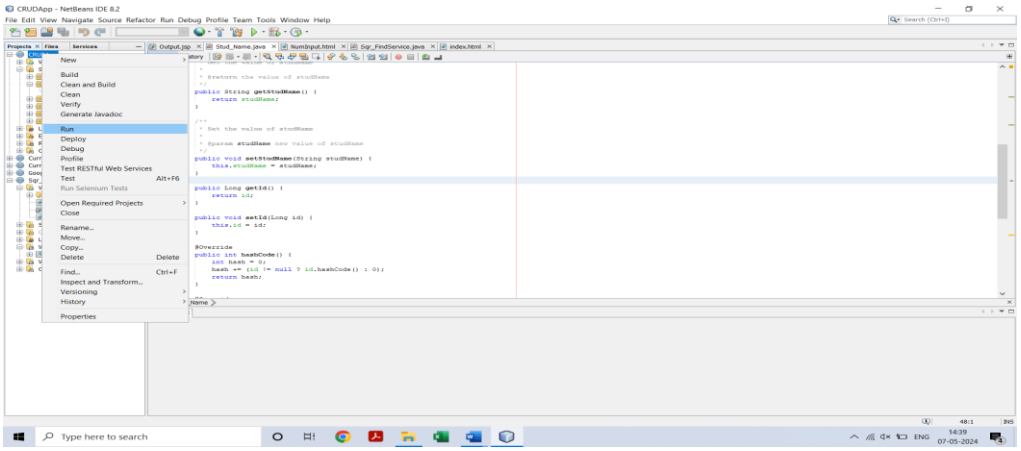
17. A new window will open. Enter name as studName. Make sure name should be exact same as of mine and then click on OK button. Actually we are setting getter and setter method for studName.



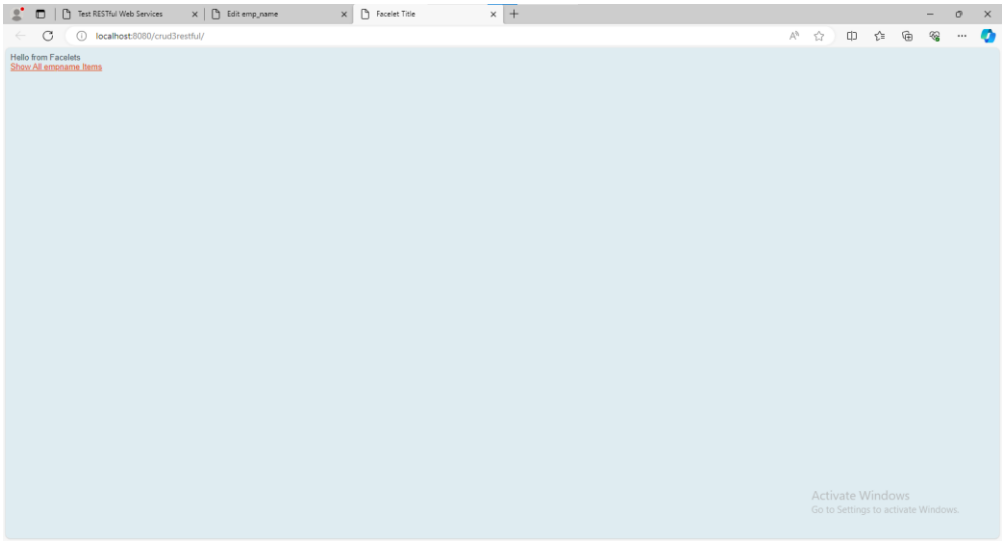
18. Now right click on web application name and Deploy it.

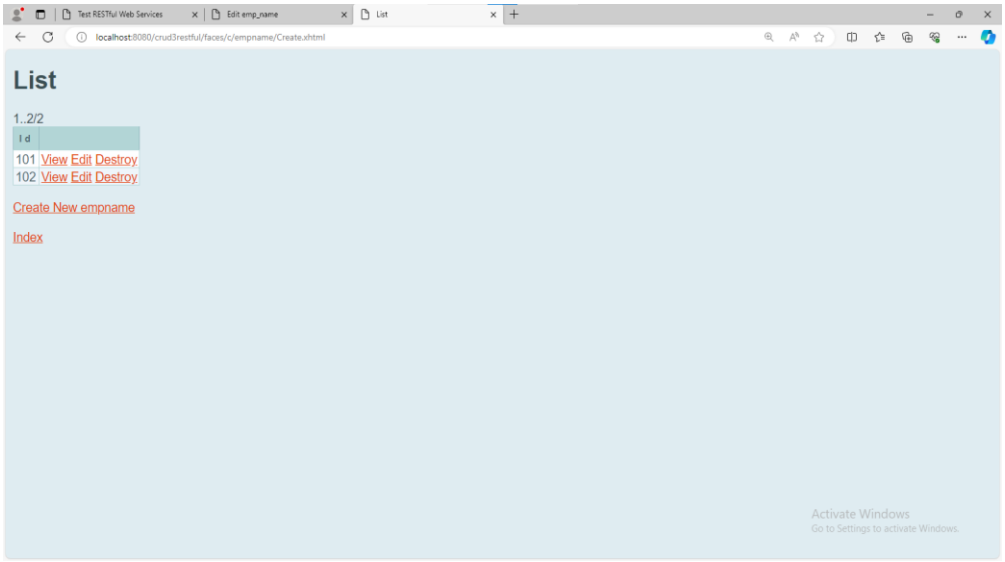


19. Now right click on project name and run it



20. A window will open in browser like below….

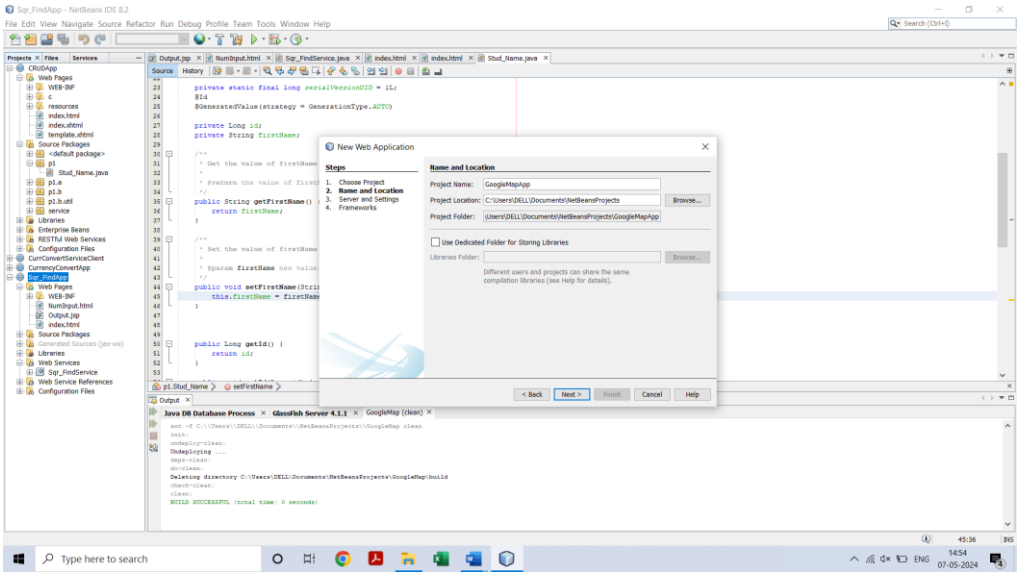



Test RESTful Web Services
Edit emp_name
List
localhost:8080/crud3restful/faces/c/empname/Create.xhtml
List
1..2/2
Id
101 View Edit Destroy
102 View Edit Destroy
Create New empname
Index
Activate Windows
Go to Settings to activate Windows.

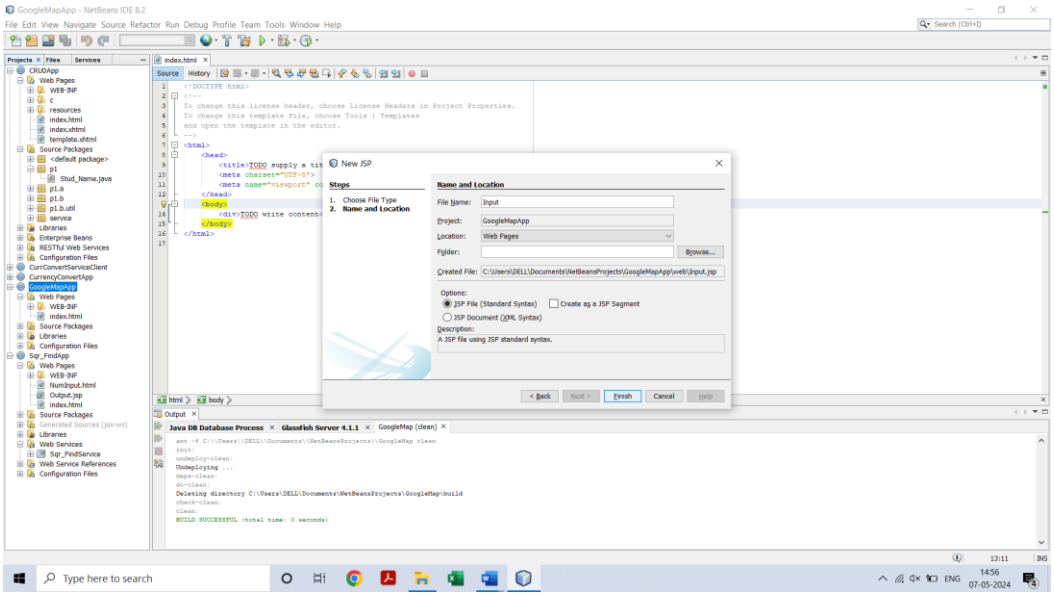
Practical-4

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

1. **First of all we need to create an Java Web Application with any name, let it be GoogleMap here using Netbeans IDE.**



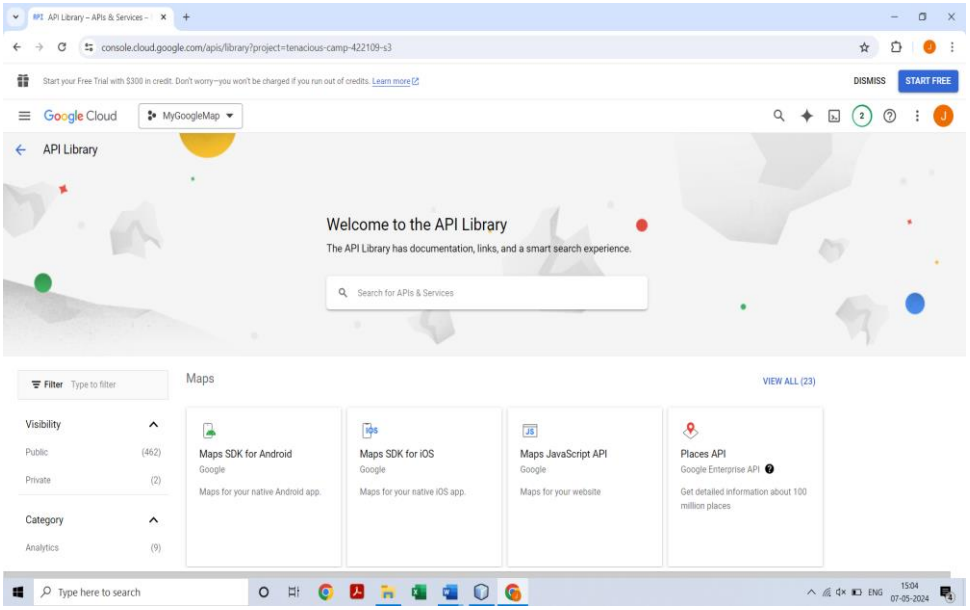
2. **Create a Input.jsp page to get Latitude and Longitude**

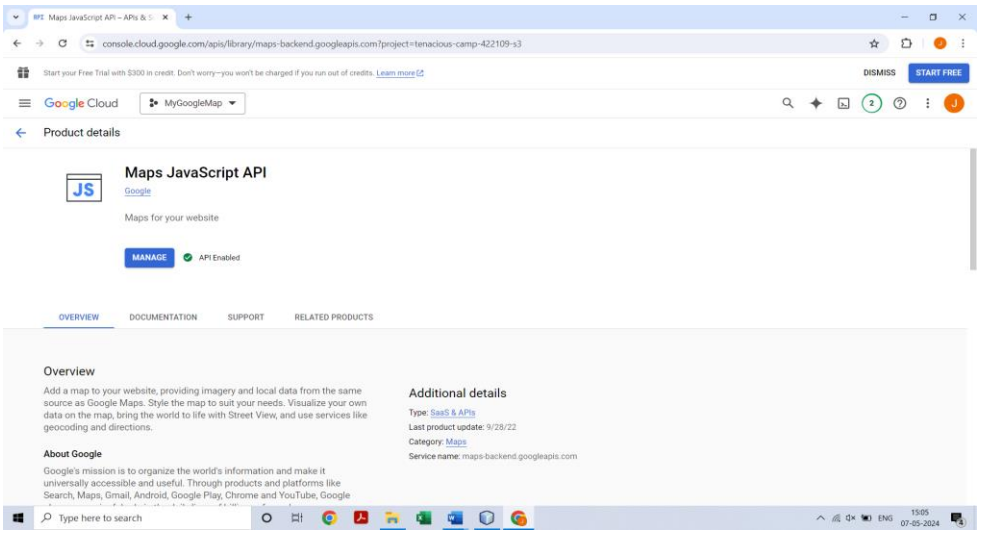


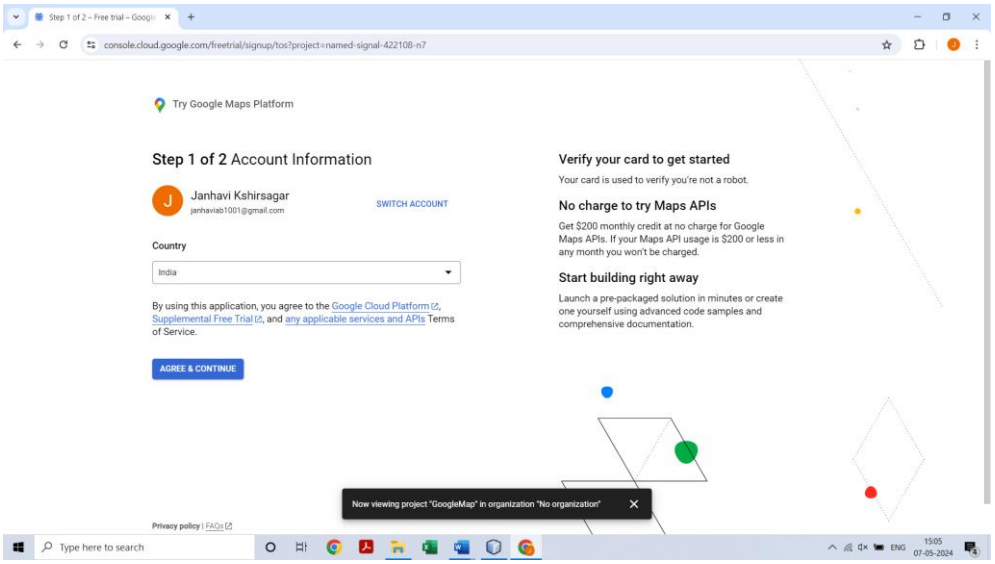
3. **The following code is inserted in that Input.jsp page**

```
<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>
```

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).**







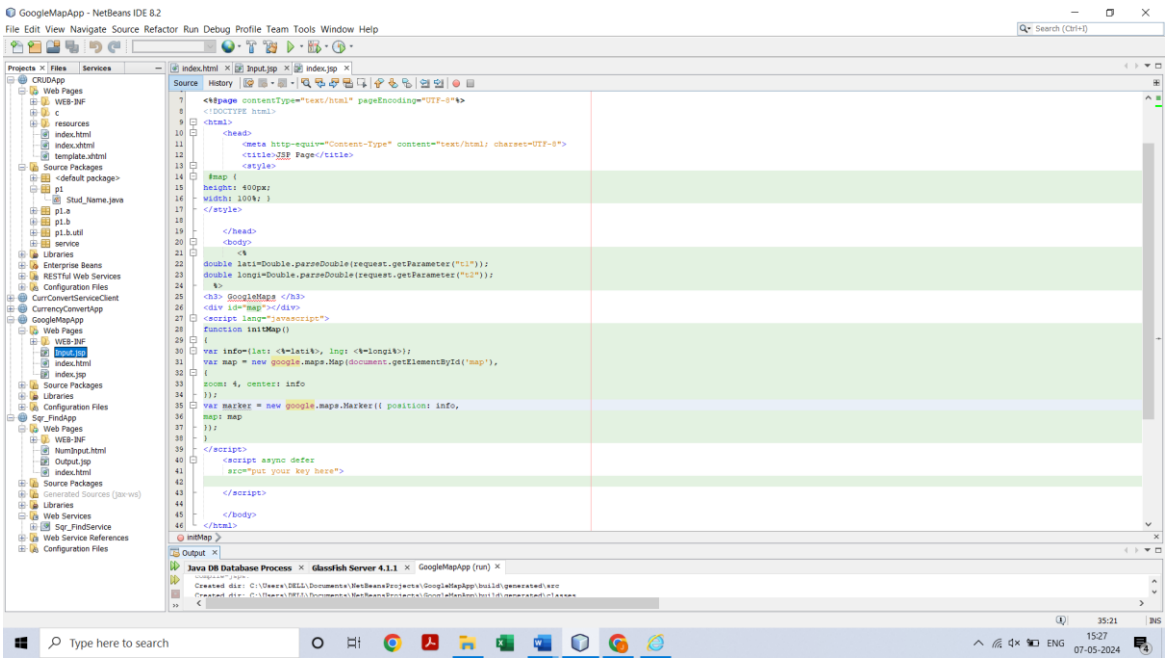
5. Enable the Google Maps API V3

6. To Create Index.jsp page

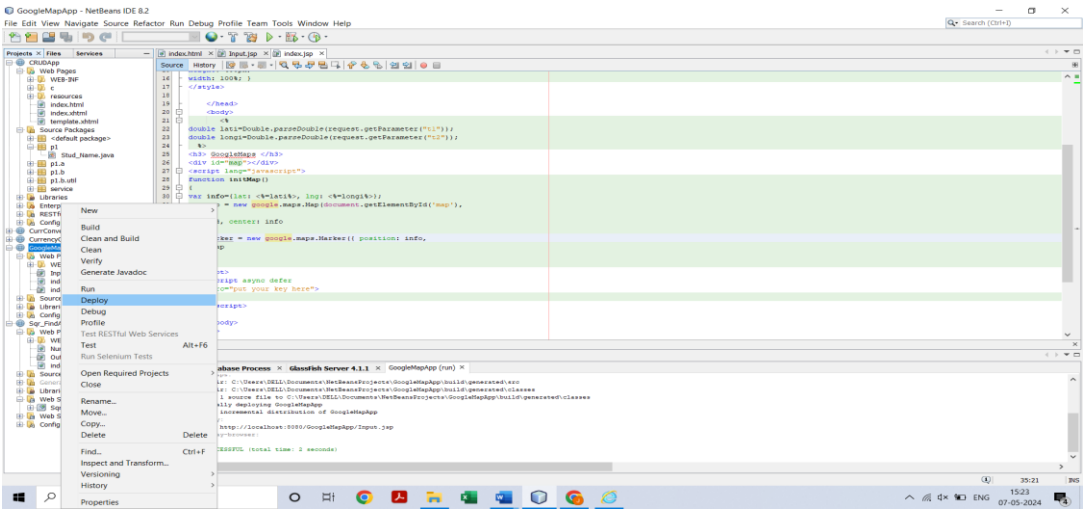
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
   <style>
          #map {
         height: 400px;
         width: 100%; }
   </style>
</head>
   <body>
  <%
          double lati=Double.parseDouble(request.getParameter("t1"));
          double longi=Double.parseDouble(request.getParameter("t2"));
    %>
<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>
```

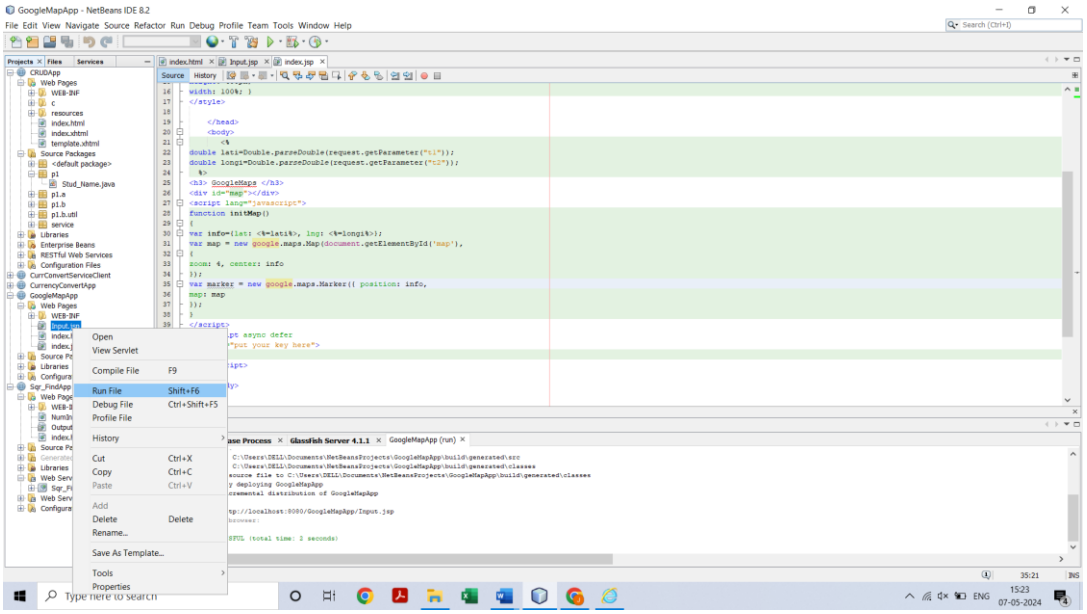
```
<script async defer
                src="put your key here">

</script>
</body>
</html>
```

Index. Jsp

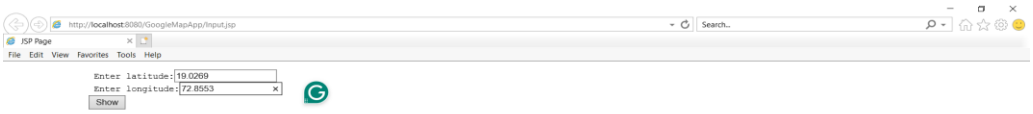


To run application :

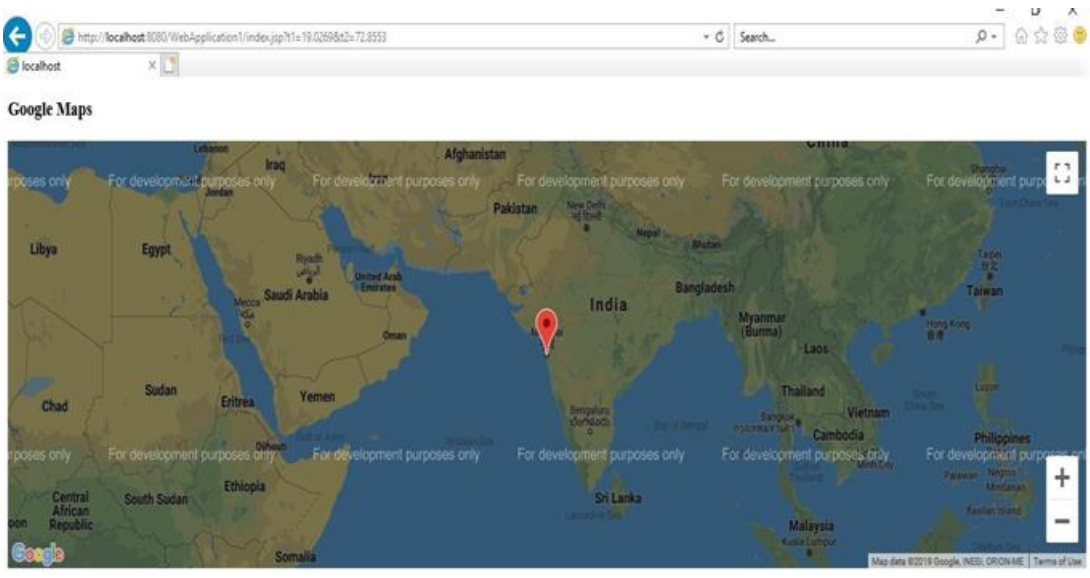



GoogleMapApp - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Open
View Servlet
Compile File F9
Run File Shift+F6
Debug File Ctrl+Shift+F5
Profile File
History
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Add
Delete Delete
Rename...
Save As Template...
Tools
Properties

Output :


http://localhost:8080/GoogleMapApp/Input.jsp
JSP Page
File Edit View Favorites Tools Help
Enter latitude: 19.0269
Enter longitude: 72.8553
Show


Type here to search

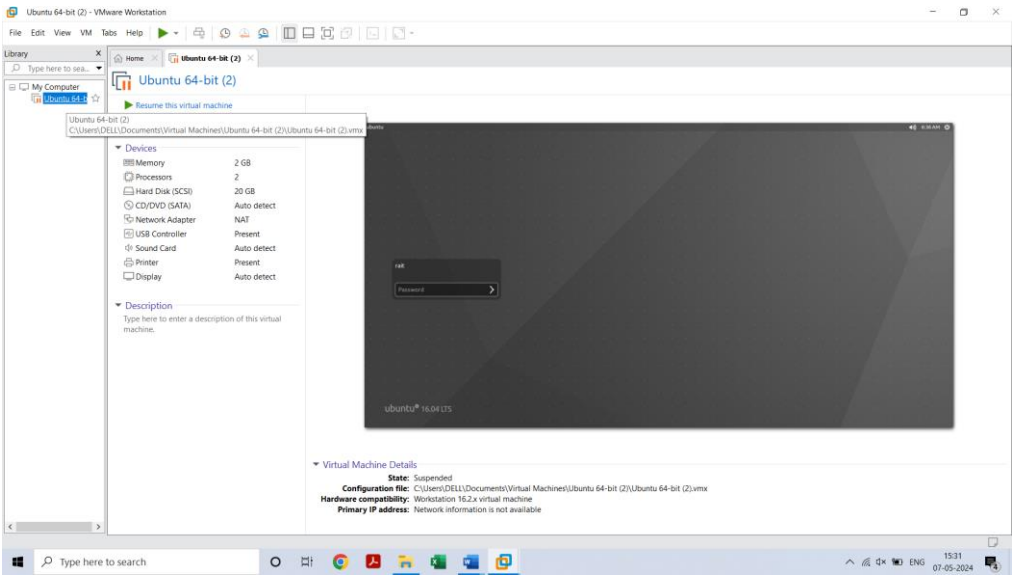

http://localhost:8080/WebApplication1/index.jsp?l1=19.0269&l2=72.8553
localhost
Google Maps
Afghanistan
Pakistan
India
Bangladesh
Saudi Arabia
Libya
Egypt
Sudan
Chad
Eritrea
Yemen
Ethiopia
Somalia
Sri Lanka
Thailand
Vietnam
Cambodia
Laos
Myanmar (Burma)
Malaysia
Philippines
Taiwan
For development purposes only

Practical-5

Installation and Configuration of virtualization using KVM

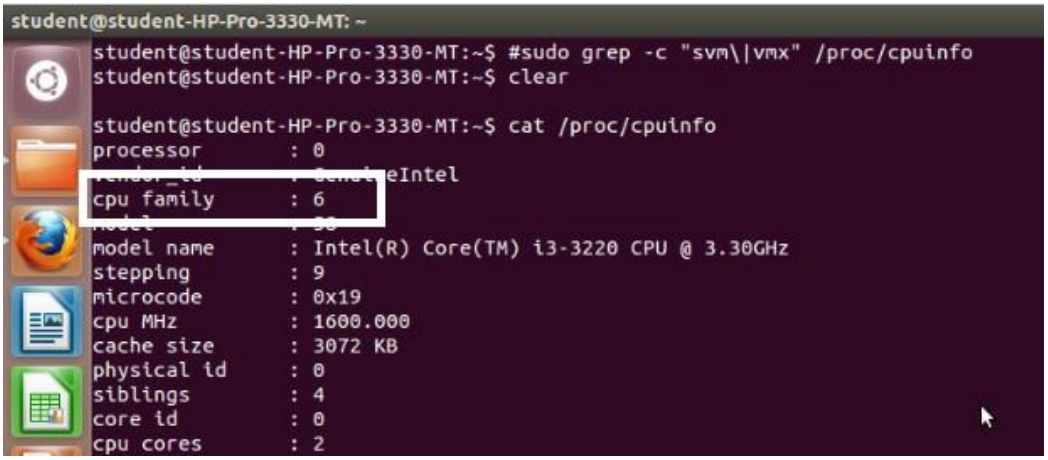
Installation Steps :

First we need to open VMware WorkStation

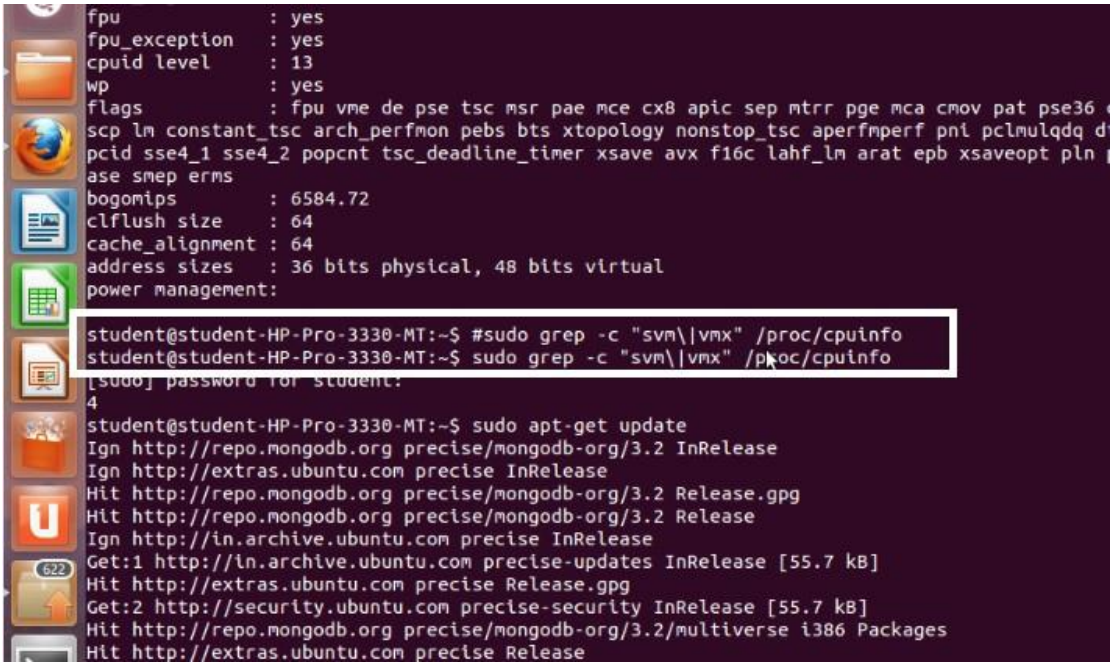


For the installation following commands are to be executed:

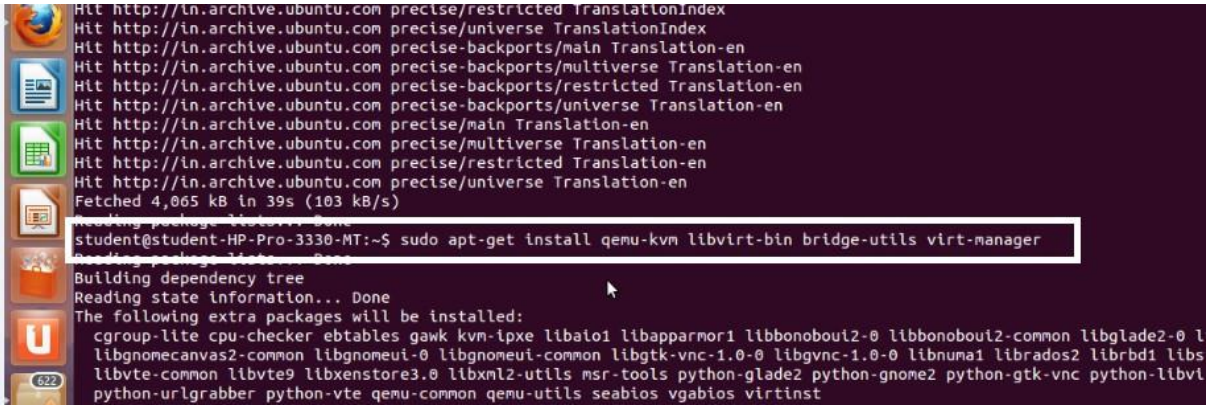
Step 1 : #sudo grep -c "svm\|vmx" /proc/cpuinfo



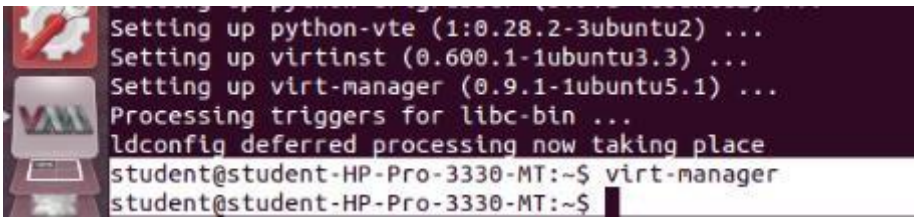
Step 2 : #sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager



Step 3 : #sudo adduser rait

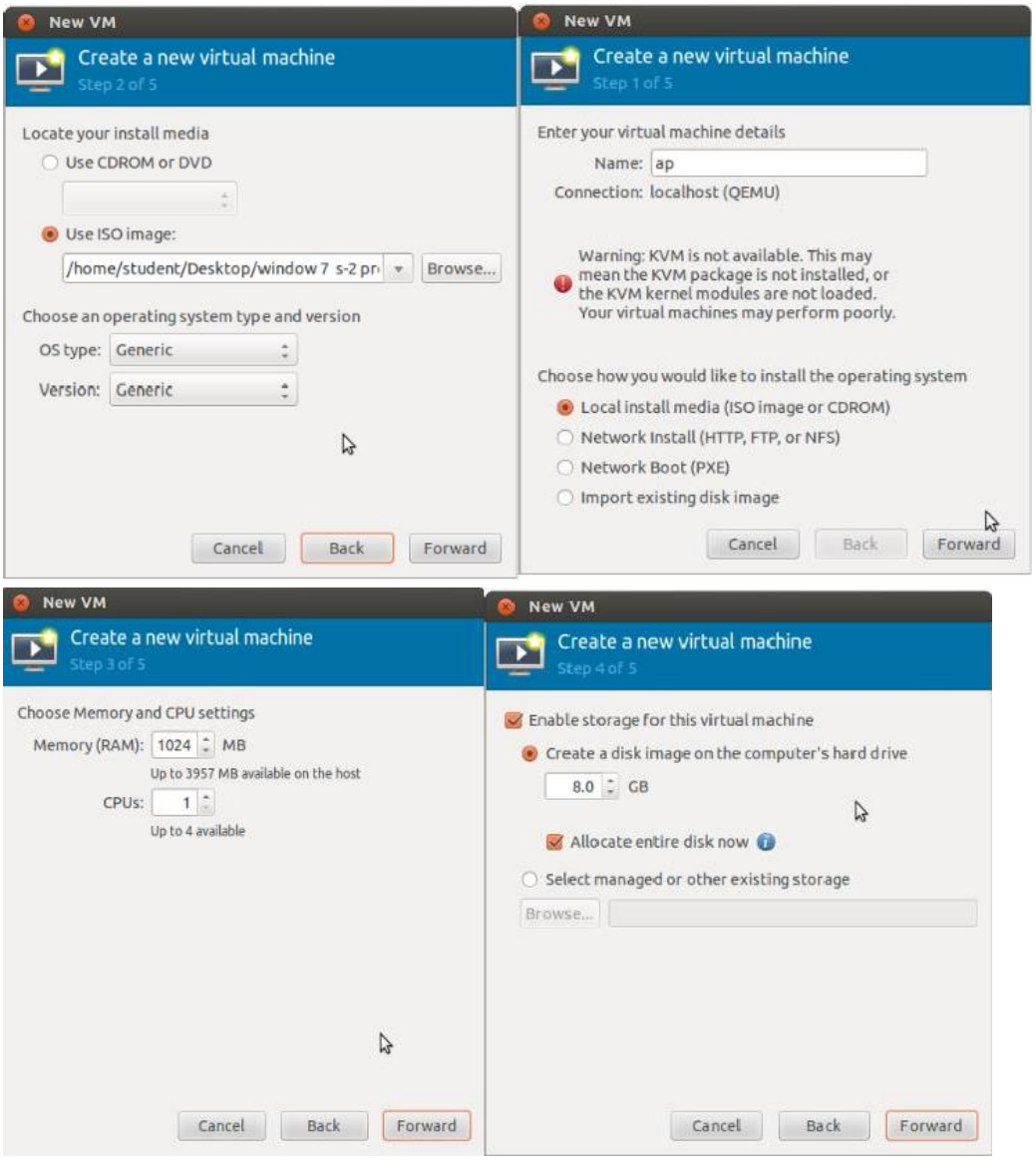


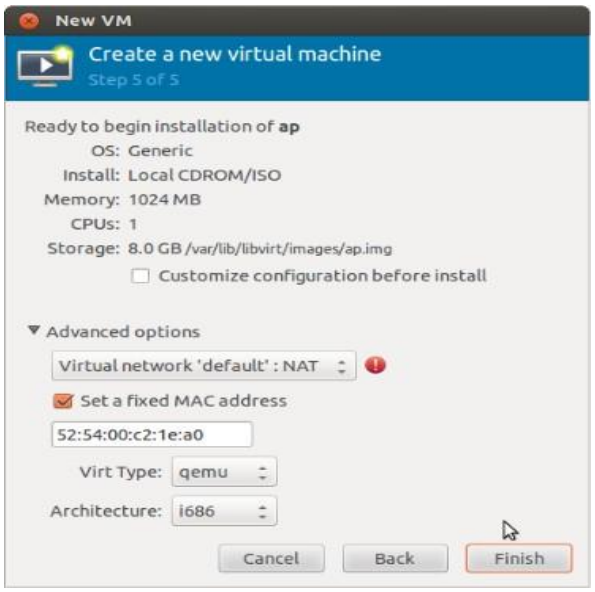
Step 4 : #sudo adduser rait libvirt
After running this command, log out and log back in as rait



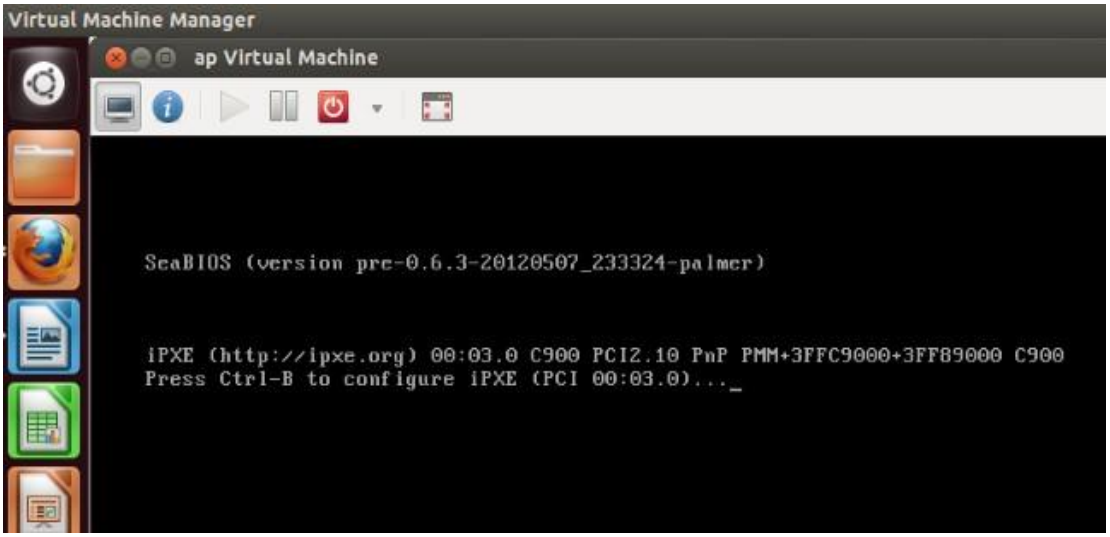
Step 5 : Open Virtual Machine Manager application and Create Virtual Machine #virt-manager as shown below

Step 6 : Create a new virtual machine as shown below

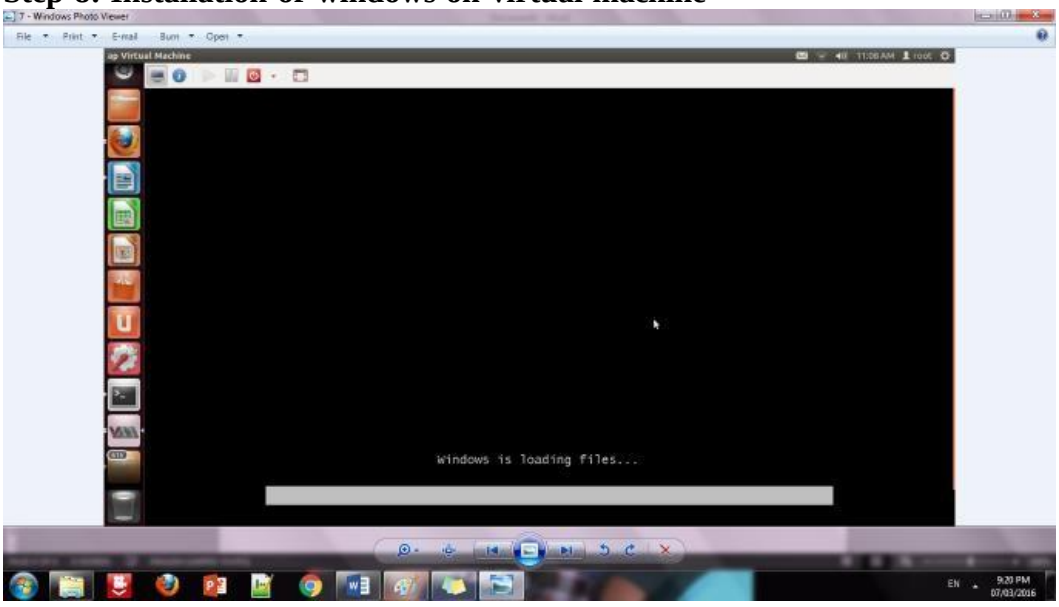




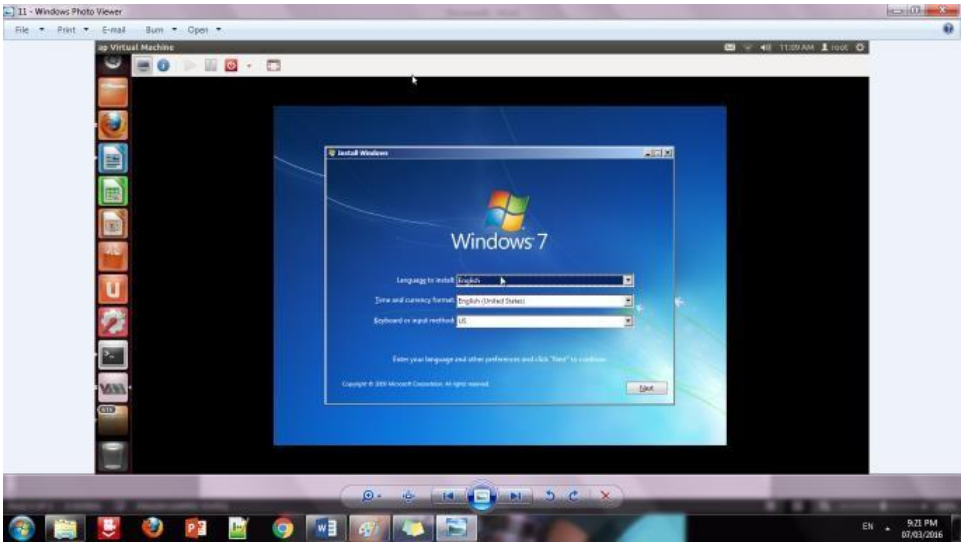
Step 7 : Install windows operating system on virtual machine



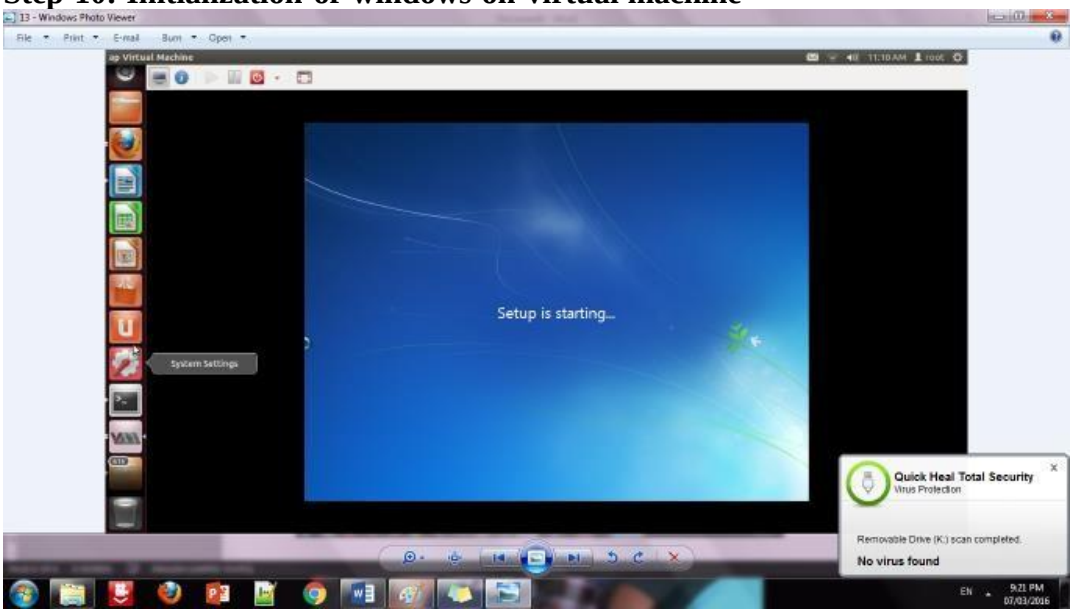
Step 8: Installation of windows on virtual machine



Step 9: Installation of windows 7 on virtual machine



Step 10: Initialization of windows on virtual machine

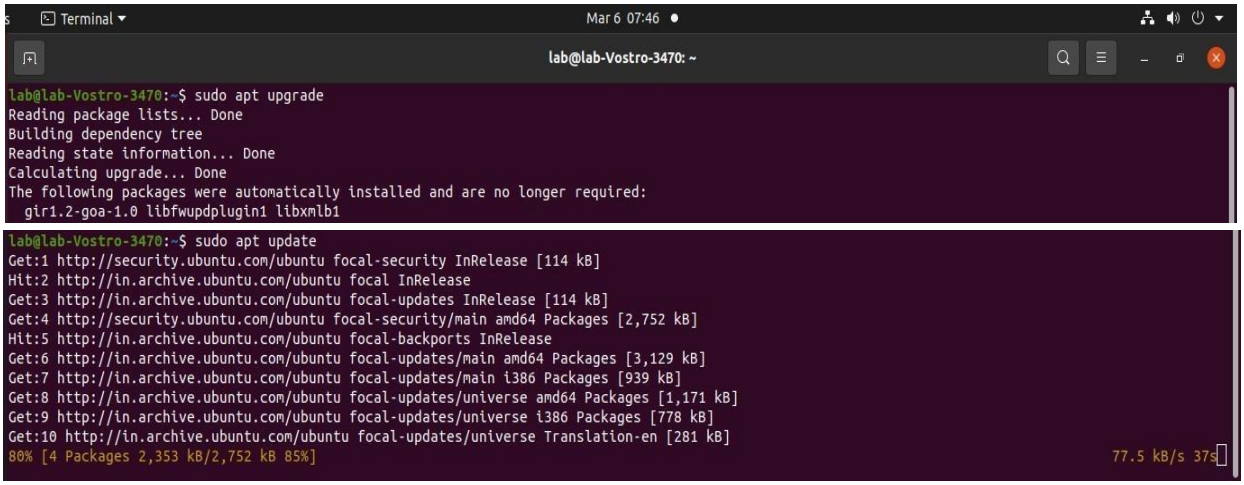


Practical-7

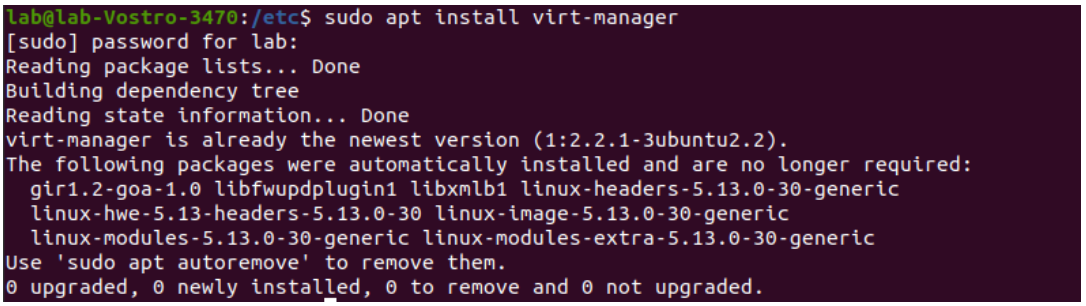
Implement FOSS-Cloud Functionality VSI (Virtual Server Infrastructure) Infrastructure as a Service (IaaS), Creating Virtual Machine or Storage

Steps:

1. Open terminal and enter the following Command ‘’sudo apt upgrade” and “sudo apt update”

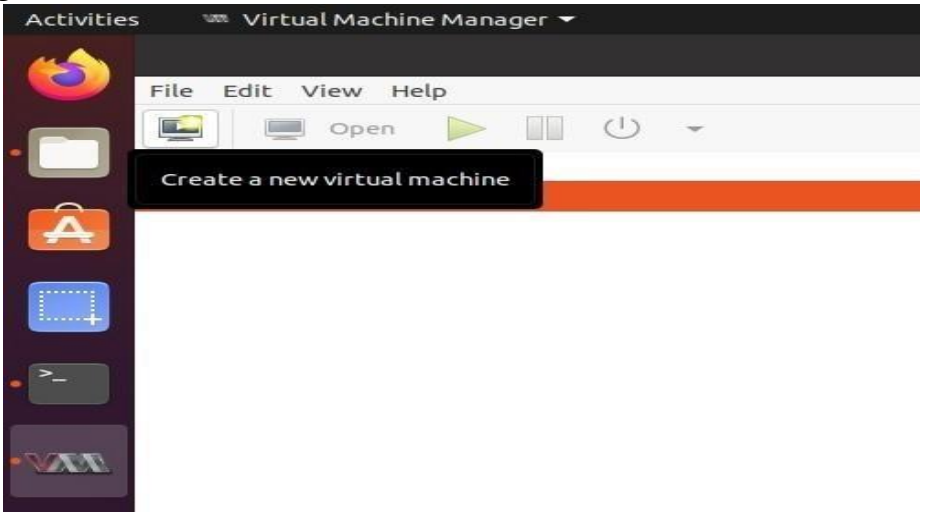


2. Enter the following command to install virt manager

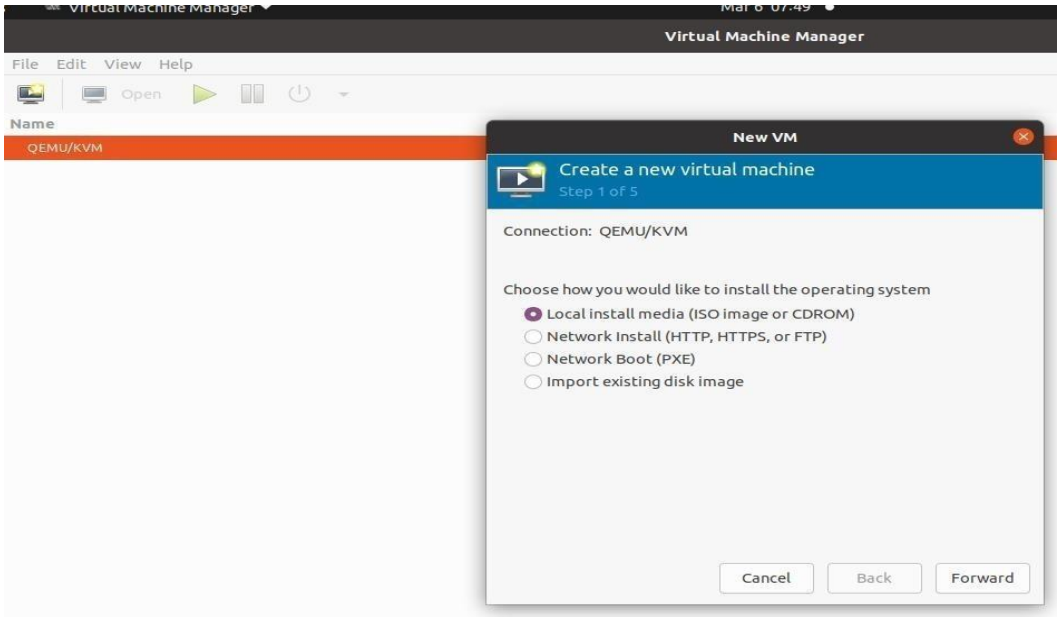


3. Launch Virt Manager using Following Command:
   $ Sudo virt-manager

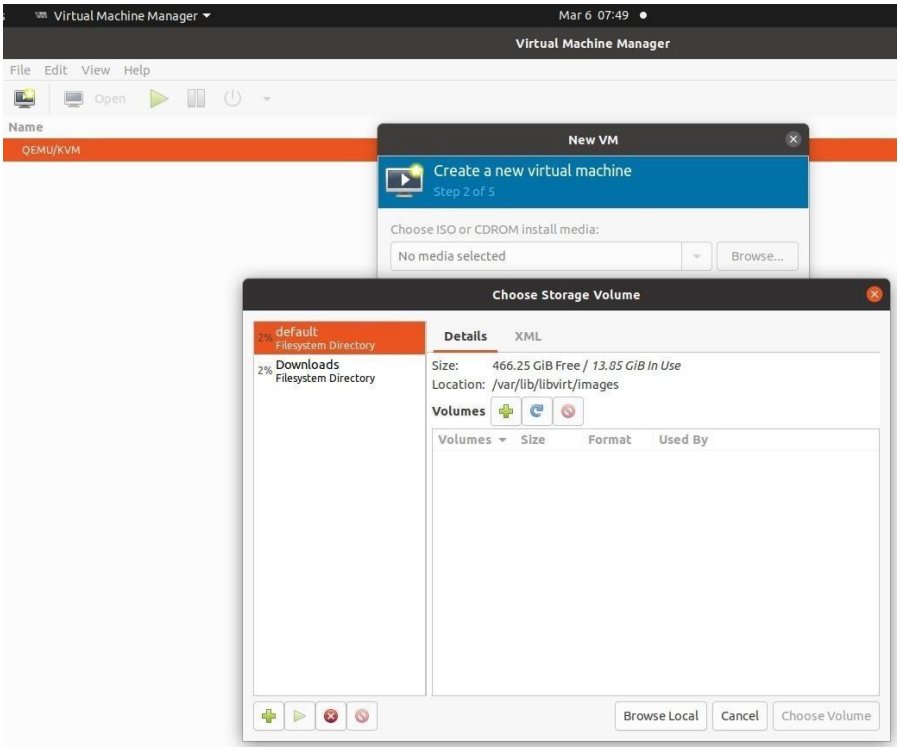
4. Once the virt manager is launched click on file➔ new virtual machine



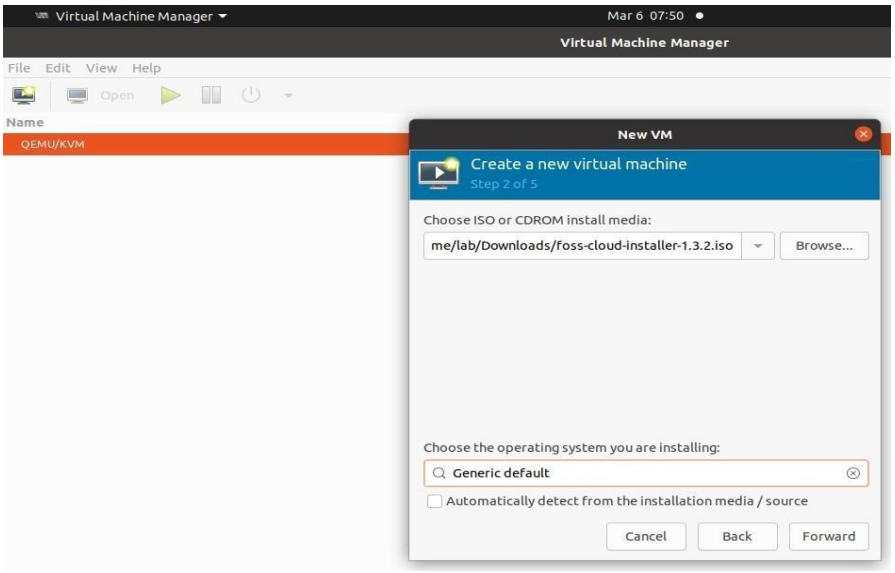
Select local install media (ISO image or CDROM) and click Forward



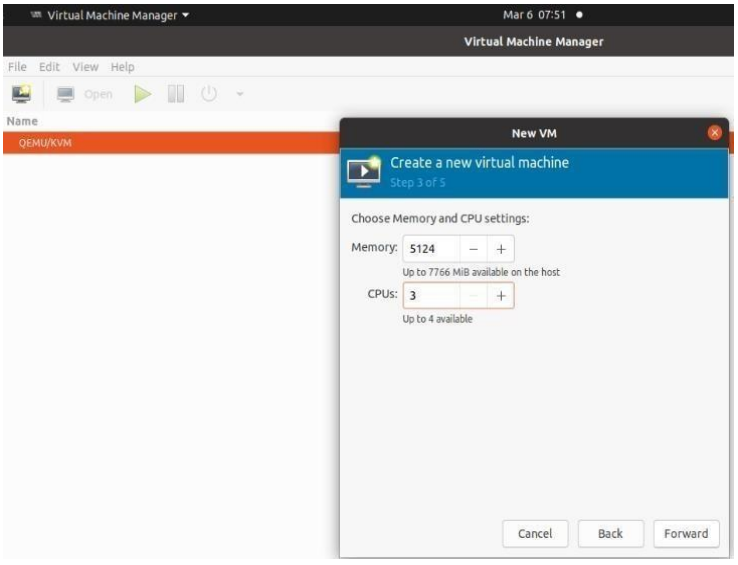
Click on Browse➔ Browse local ➔ select downloaded iso file ➔ Open

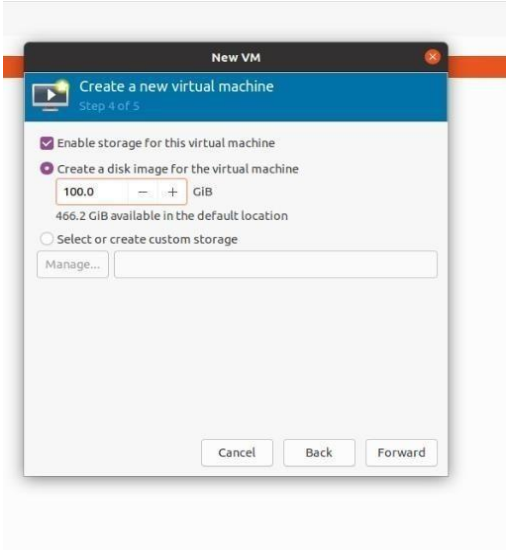


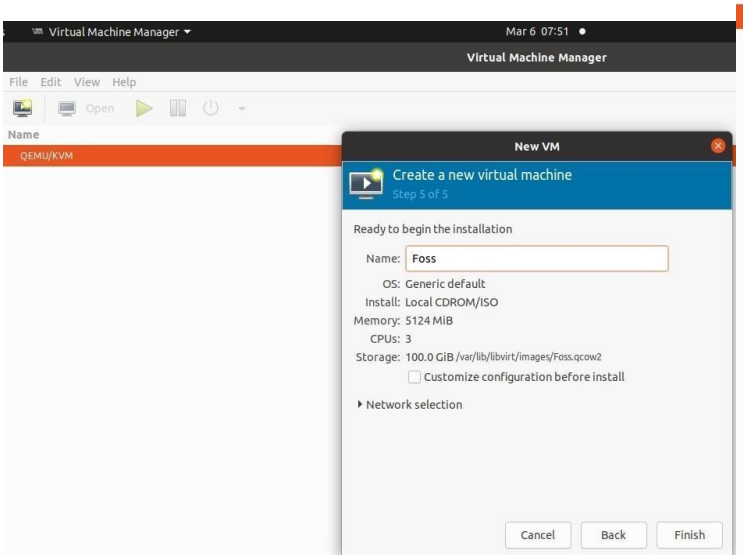
Uncheck The checkbox and Choose the OS you are installing as “Generic default” and click“Forward”

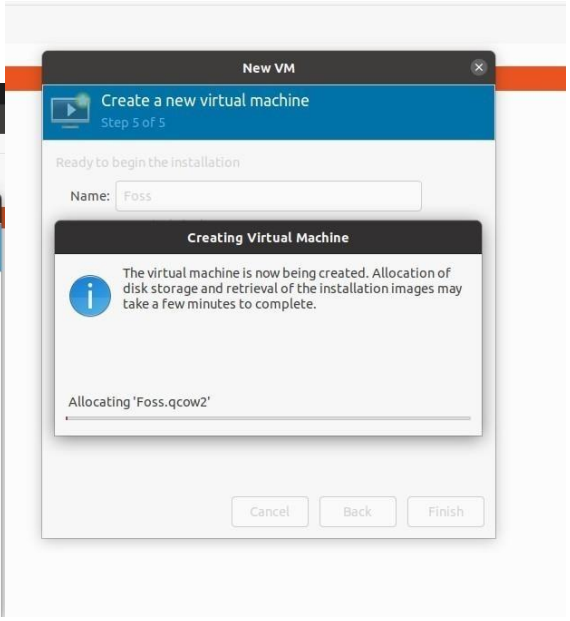


5. Provide the Memory, CPU,Storage,Name For Virtual Machine and click “Forward”



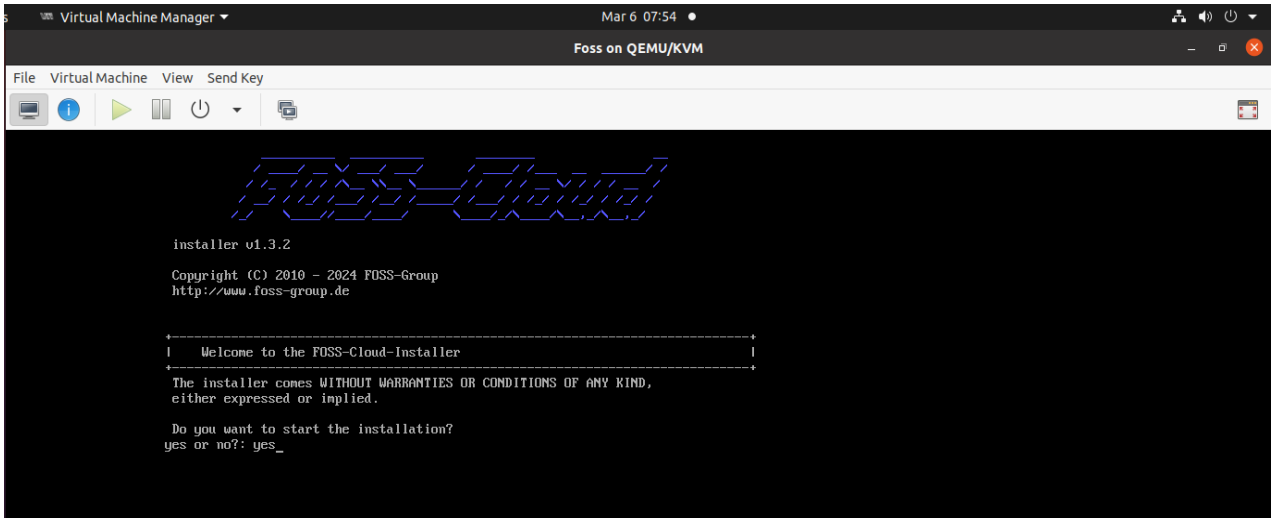




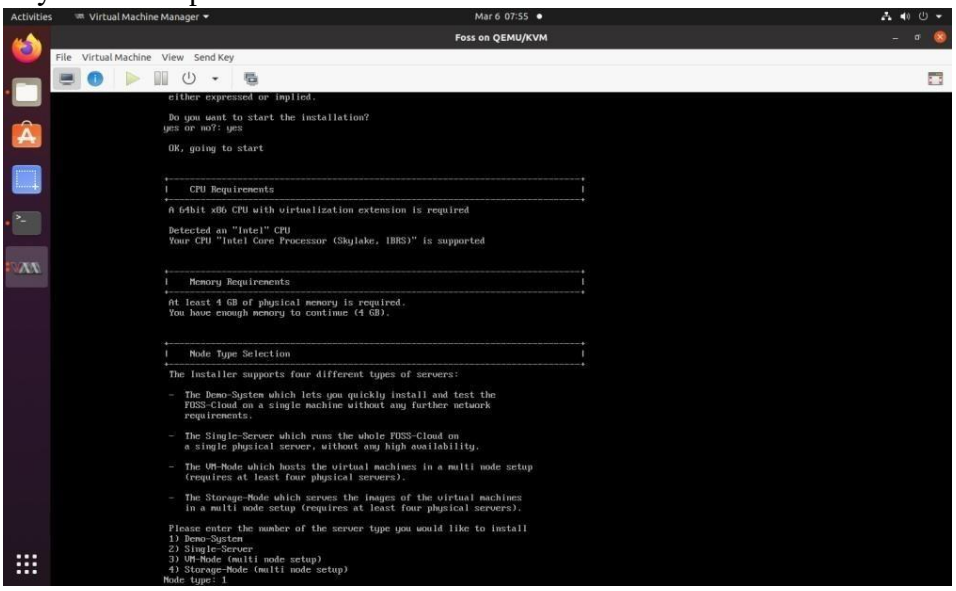


6. Once the foss Cloud is launched select “Foss-Cloud installer :default boot options” and press Enter

Types yes to start the installation

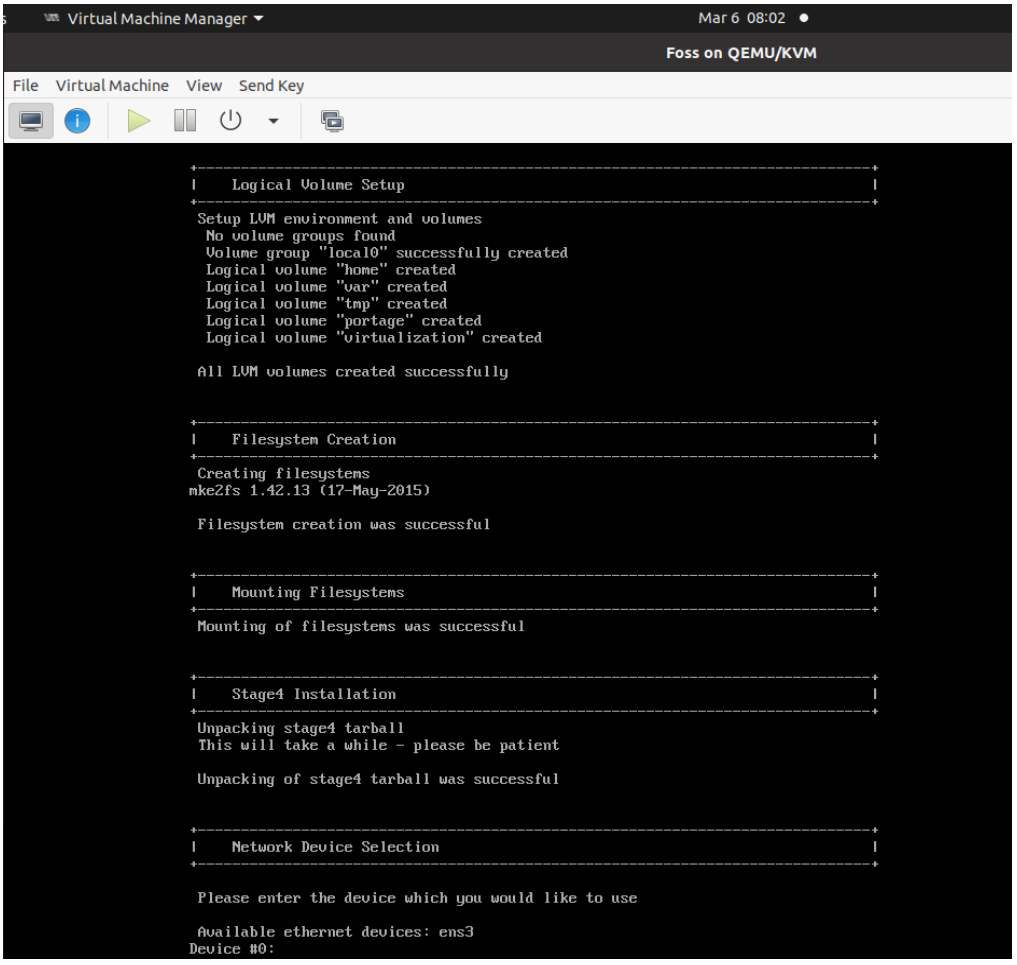


7. Select “Demo System” and press Enter



Enter the Device name as “sda” and press enter

```
+-----------------------------------------------------------------------------+
|    Installation Device Selection                                            |
+-----------------------------------------------------------------------------+
 A dedicated SCSI, SATA or PATA disk is required for the installation
 The disk has to be at least 130 GB in size

 Found sda (150 GB). Size is OK

 Below you will find a list of all detected and supported disks
 sda (150 GB)

 Please enter the device name on which you would like to install
Device: sda
 'sda' will be used as the installation device.


+-----------------------------------------------------------------------------+
|    Logical Volume Cleanup and Preparation                                   |
+-----------------------------------------------------------------------------+
 Checking for existing volume groups and physical volumes
  No volume groups found
  No volume groups found


+-----------------------------------------------------------------------------+
|    Installation Device Partitioning                                         |
+-----------------------------------------------------------------------------+
 Below is the existing partition layout of your selected device
```

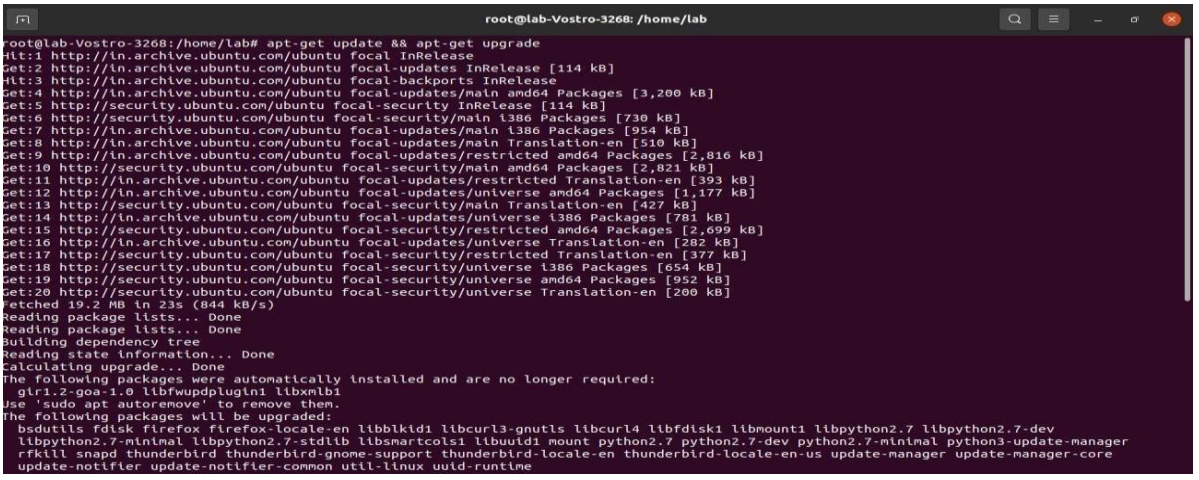


Practical-9

Using AWS Flow Framework develop application that includes a simple workflow. Workflow calls an activity to print hello world to the console. It must define the basic usage of AWS Flow Framework, including defining contracts, implementation of activities and workflow coordination logic and worker programs to host them

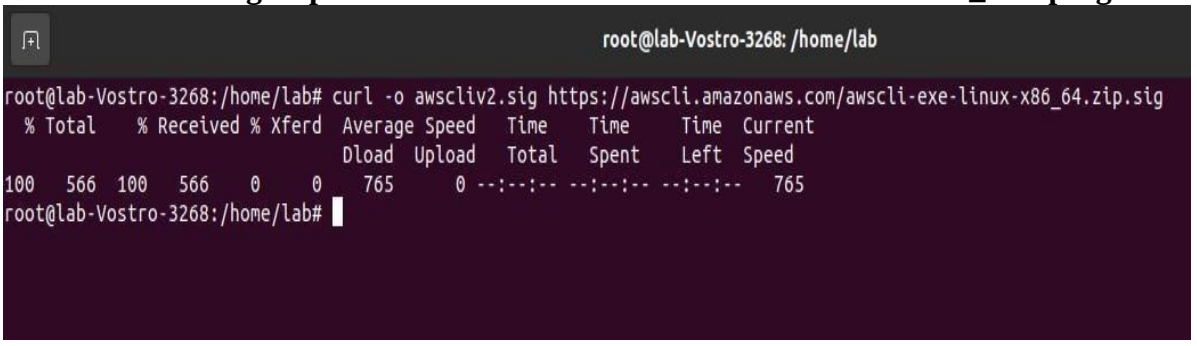
Step 1: Open Terminal and Update and Upgrade your system by command

sudo apt-get update && sudo apt-get upgrade



Step 2: Download awscliv2.zip with command
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

Step 3: Download awscliv2.sig file with command
curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip.sig



Step 4: unzip awscliv2.zip with command

unzip awscliv2.zip

```
root@lab-Vostro-3268:/home/lab# unzip awscliv2.zip
Archive:  awscliv2.zip
   creating: aws/
   creating: aws/dist/
  inflating: aws/THIRD_PARTY_LICENSES
  inflating: aws/install
  inflating: aws/README.md
   creating: aws/dist/awscli/
   creating: aws/dist/cryptography/
   creating: aws/dist/docutils/
   creating: aws/dist/lib-dynload/
  inflating: aws/dist/aws
  inflating: aws/dist/aws_completer
  inflating: aws/dist/libpython3.11.so.1.0
  inflating: aws/dist/_awscrt.abi3.so
  inflating: aws/dist/_cffi_backend.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/_ruamel_yaml.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/libz.so.1
  inflating: aws/dist/liblzma.so.0
  inflating: aws/dist/libbz2.so.1
  inflating: aws/dist/libffi.so.5
  inflating: aws/dist/libsqlite3.so.0
  inflating: aws/dist/base_library.zip
  inflating: aws/dist/lib-dynload/_pickle.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_hashlib.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_sha3.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_blake2.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_sha256.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_md5.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_sha1.cpython-311-x86_64-linux-gnu.so
  inflating: aws/dist/lib-dynload/_sha512.cpython-311-x86_64-linux-gnu.so
```

Step 5: Run command
sudo ./aws/install

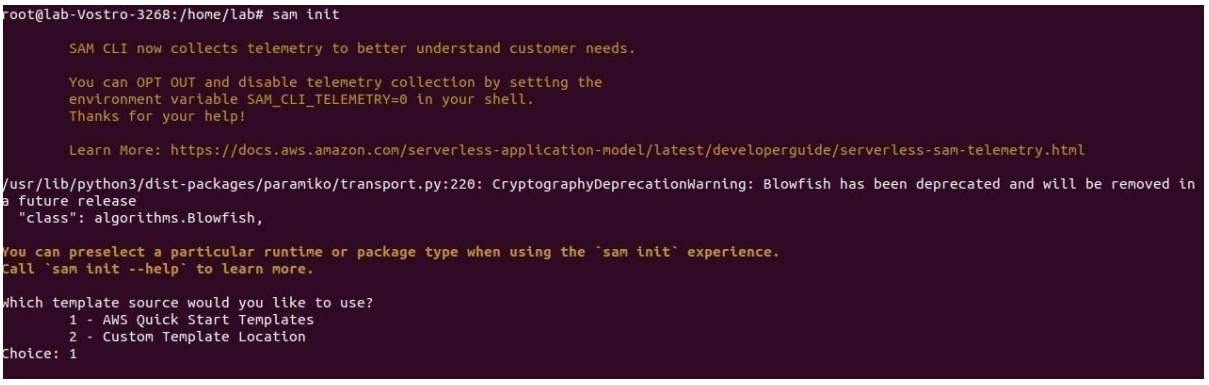
```
root@lab-Vostro-3268:/home/lab#  sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

Step 6: Type command
pip3 install aws-sam-cli

```
root@lab-Vostro-3268:/home/lab# pip3 install aws-sam-cli
Collecting aws-sam-cli
  Downloading aws_sam_cli-1.113.0-py3-none-any.whl (5.9 MB)
     |████████████████████████████████| 5.9 MB 21 kB/s
Collecting aws-lambda-builders==1.47.0
  Downloading aws_lambda_builders-1.47.0-py3-none-any.whl (130 kB)
     |████████████████████████████████| 130 kB 547 kB/s
Collecting tzlocal==5.2
  Downloading tzlocal-5.2-py3-none-any.whl (17 kB)
Collecting dateparser~=1.2
  Downloading dateparser-1.2.0-py2.py3-none-any.whl (294 kB)
     |████████████████████████████████| 294 kB 2.4 MB/s
Collecting Flask<3.1
  Downloading flask-3.0.2-py3-none-any.whl (101 kB)
     |████████████████████████████████| 101 kB 447 kB/s
Collecting boto3<2,>=1.29.2
  Downloading boto3-1.34.76-py3-none-any.whl (139 kB)
     |████████████████████████████████| 139 kB 558 kB/s
Collecting pyopenssl~=24.1.0
  Downloading pyOpenSSL-24.1.0-py3-none-any.whl (56 kB)
     |████████████████████████████████| 56 kB 580 kB/s
Collecting requests~=2.31.0
  Using cached requests-2.31.0-py3-none-any.whl (62 kB)
Collecting click~=8.1
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
     |████████████████████████████████| 97 kB 377 kB/s
Collecting watchdog==4.0.0
  Downloading watchdog-4.0.0-py3-none-manylinux2014_x86_64.whl (82 kB)
     |████████████████████████████████| 82 kB 113 kB/s
Collecting chevron~=0.12
  Downloading chevron-0.14.0-py3-none-any.whl (11 kB)
Collecting ruamel-yaml~=0.18.6
  Downloading ruamel.yaml-0.18.6-py3-none-any.whl (117 kB)
     |████████████████████████████████| 117 kB 363 kB/s
Collecting aws-sam-translator==1.86.0
```

Step 7: Type command sam init in terminal to launch Sam

CLISelect 1st option to use AWS Quick Ttart Templates



Step 8: Select Template no.1 Hello World Example

Step 9: Type "N" if it ask to use most popular runtime and package type

Open new terminal by pressing ctrl+shift+T and check for python version by command python –version

Select the option according to your python version in my case its option 19- python 3.11

```
Use the most popular runtime and package type? (Python and zip) [y/N]: n

Which runtime would you like to use?
        1 - aot.dotnet7 (provided.al2)
        2 - dotnet8
        3 - dotnet6
        4 - go1.x
        5 - go (provided.al2)
        6 - go (provided.al2023)
        7 - graalvm.java11 (provided.al2)
        8 - graalvm.java17 (provided.al2)
        9 - java21
        10 - java17
        11 - java11
        12 - java8.al2
        13 - nodejs20.x
        14 - nodejs18.x
        15 - nodejs16.x
        16 - python3.9
        17 - python3.8
        18 - python3.12
        19 - python3.11
        20 - python3.10
        21 - ruby3.2
        22 - rust (provided.al2)
        23 - rust (provided.al2023)
Runtime: 17
```

Step 10: Select package type as <u>Zip</u>

```
What package type would you like to use?
        1 - Zip
        2 - Image
Package type: 1
```

Step 11: Now choose Yes option everytime it ask .

Give project name as per your preference in my case its <u>sam-app-test</u>

```
Based on your selections, the only dependency manager available is pip.
We will proceed copying the template using pip.

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/N]: y
X-Ray will incur an additional cost. View https://aws.amazon.com/xray/pricing/ for more details

Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch-application-insights.html [y/N]: y
AppInsights monitoring may incur additional cost. View https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/appinsights-what-is.html
#appinsights-pricing for more details

Would you like to set Structured Logging in JSON format on your Lambda functions?  [y/N]: y
Structured Logging in JSON format might incur an additional cost. View https://docs.aws.amazon.com/lambda/latest/dg/monitoring-cloudwatchlogs.
html#monitoring-cloudwatchlogs-pricing for more details

Project name [san-app]: san-app-test

Cloning from https://github.com/aws/aws-sam-cli-app-templates (process may take a moment)

    -----------------------
    Generating application:
    -----------------------
    Name: sam-app-test
    Runtime: python3.8
    Architectures: x86_64
    Dependency Manager: pip
    Application Template: hello-world
    Output Directory: .
    Configuration file: san-app-test/samconfig.toml

    Next steps can be found in the README file at sam-app-test/README.md
```

Step 12: Now one folder will be created by your provided project name go into that folder by command cd (folder name)

After entering the project folder we will invoke the HelloWorldFunction by using commandsam local invoke „HelloWorldFunction“

```
root@lab-Vostro-3268:/home/lab/sam-app-test# sam local invoke 'HelloWorldFunction'
/usr/lib/python3/dist-packages/paramiko/transport.py:220: CryptographyDeprecationWarning: Blowfish has been deprecated and will be removed in
a future release
  "class": algorithms.Blowfish,
Invoking app.lambda_handler (python3.8)
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-python3.8
Building image.....................................................................................................................
.....................................................................................................................
.....................................................................................................................
.....................................................................................................
Using local image: public.ecr.aws/lambda/python:3.8-rapid-x86_64.

Mounting /home/lab/sam-app-test/hello_world as /var/task:ro,delegated, inside runtime container
START RequestId: 065fefd3-5d41-4b87-bb31-8d820fb635e6 Version: $LATEST
END RequestId: 4b9baa5c-2523-443d-b340-f86e2b139f6a
REPORT RequestId: 4b9baa5c-2523-443d-b340-f86e2b139f6a  Init Duration: 0.06 ms  Duration: 99.11 ms      Billed Duration: 100 ms Memory Size: 1
28 MB   Max Memory Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"hello world\"}"}
root@lab-Vostro-3268:/home/lab/sam-app-test#
```

It should give you StatusCode:200

Use command sudo snap install docker if docker error occurs.

Step 13: Type command sam local start-api this will give you URL open it in any browser.

```
root@lab-Vostro-3268:/home/lab/sam-app-test# sam local start-api
/usr/lib/python3/dist-packages/paramiko/transport.py:220: CryptographyDeprecationWarning: Blowfish has been deprecated and will be removed in
a future release
  "class": algorithms.Blowfish,
Initializing the lambda functions containers.
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-python3.11
Building image.....................................................................................................................
.....................................................
Using local image: public.ecr.aws/lambda/python:3.11-rapid-x86_64.

Mounting /home/lab/sam-app-test/hello_world as /var/task:ro,delegated, inside runtime container
Containers Initialization is done.
Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
You can now browse to the above endpoints to invoke your functions. You do not need to restart/reload SAM CLI while working on your functions,
changes will be reflected instantly/automatically. If you used sam build before running local commands, you will need to re-run sam build for
the changes to be picked up. You only need to restart SAM CLI if you update your AWS SAM template
2024-04-03 09:26:18 WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
2024-04-03 09:26:18 Press CTRL+C to quit
Invoking app.lambda_handler (python3.11)
Reuse the created warm container for Lambda function 'HelloWorldFunction'
Lambda function 'HelloWorldFunction' is already running
START RequestId: 0652f01c-89a7-43d9-85fb-3844c8f1a32b Version: $LATEST
END RequestId: b229bc70-c5b6-4bf2-b9a3-97b2cf9340a1
REPORT RequestId: b229bc70-c5b6-4bf2-b9a3-97b2cf9340a1  Init Duration: 0.04 ms  Duration: 1153.99 ms    Billed Duration: 1154 ms        Memory
 Size: 128 MB   Max Memory Used: 128 MB

No Content-Type given. Defaulting to 'application/json'.
2024-04-03 09:26:38 127.0.0.1 - - [03/Apr/2024 09:26:38] "GET /hello HTTP/1.1" 200 -
2024-04-03 09:26:39 127.0.0.1 - - [03/Apr/2024 09:26:39] "GET /favicon.ico HTTP/1.1" 403 -
```

Output:

Practical-10

Implementation of Openstack with user and private network creation.

Implementation of Openstack with user and private network creation. Minimum

Requirements

Before we begin, ensure you have the following minimum prerequisites

1. **A fresh Ubuntu 18.04 installation**
2. **User with sudo privileges**
3. **4 GB RAM**
4. **2 vCPUs**
5. **Hard disk capacity of 10 GB**
6. **Internet connection**

With the minimum requirements satisfied, we can now proceed.

Steps:

Step 1: Update and Upgrade the System
To start off, log into your Ubuntu 18.04 system using SSH protocol and update & upgra system repositories using the following command.

```
apt update -y && apt upgrade -y
```

Sample Output

```
root@ubuntu:/# apt update -y && apt ugrade -y
Hit:1 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [8570 kB]
Get:5 http://archive.canonical.com/ubuntu bionic InRelease [10.2 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:7 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/universe Translation-en [4941 kB]
Get:8 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [151 kB]
Get:9 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/multiverse Translation-en [108 kB]
Get:10 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [627 kB]
```

Next reboot the system using the command.

```
sudo reboot
```

Step 2: Create Stack user and assign sudo priviledge
Best practice demands that devstack should be run as a regular user with sudo privileg With that in mind, we are going to add a new user called “stack” and assign sudo privi To create stack user execute

```
sudo adduser -s /bin/bash -d /opt/stack -m stack
```

Next, run the command below to assign sudo privileges to the user Sample Output

```
echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
```

```
root@ubuntu:/# sudo useradd -s /bin/bash -d /opt/stack -m stack
root@ubuntu:/#
root@ubuntu:/# echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
stack ALL=(ALL) NOPASSWD: ALL
root@ubuntu:/#
```

Step 3: Install git and download DevStack

Once you have successfully created the user 'stack' and assigned sudo privileges, swit the user using the command.

```
su - stack
```

In most Ubuntu 18.04 systems, git comes already installed. If by any chance git is miss install it by running the following command.

```
sudo apt install git -y
```

Sample output

```
root@ubuntu:~# su - stack
stack@ubuntu:~$
stack@ubuntu:~$ sudo apt install git -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.17.1-1ubuntu0.4).
The following packages were automatically installed and are no longer required:
  grub-pc-bin libnuma1
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Using git, clone devstack's git repository as shown. Sample

```
git   clone   https://git.openstack.org/openstack-dev/devstack
```

output

```
stack@ubuntu:~$ git clone https://git.openstack.org/openstack-dev/devstack
Cloning into 'devstack'...
warning: redirecting to https://opendev.org/openstack/devstack/
remote: Enumerating objects: 43615, done.
remote: Counting objects: 100% (43615/43615), done.
remote: Compressing objects: 100% (12575/12575), done.
remote: Total 43615 (delta 31152), reused 42370 (delta 30360)
Receiving objects: 100% (43615/43615), 8.27 MiB | 24.61 MiB/s, done.
Resolving deltas: 100% (31152/31152), done.
stack@ubuntu:~$
stack@ubuntu:~$ ls
devstack
stack@ubuntu:~$
```

Step 4: Create devstack configuration file

In this step, navigate to the devstack directory. Then create

```
cd devstack
```

```
vim local.conf
```

Paste the following content

```
[[local|localrc]]

# Password for KeyStone, Database, RabbitMQ and Service
ADMIN_PASSWORD=StrongAdminSecret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
```

```
SERVICE_PASSWORD=$ADMIN_PASSWORD

# Host IP - get your Server/VM IP address from ip addr command
HOST_IP=10.208.0.10
```

Save and exit the text editor. NOTE:

1. **The ADMIN_PASSWORD is the password that you will use to log in to the OpenStack login page. The default username is admin.**
2. **The HOST_IP is your system's IP address that is obtained by running ifconfig or ip addr commands.**

Step 5: Install OpenStack with Devstack

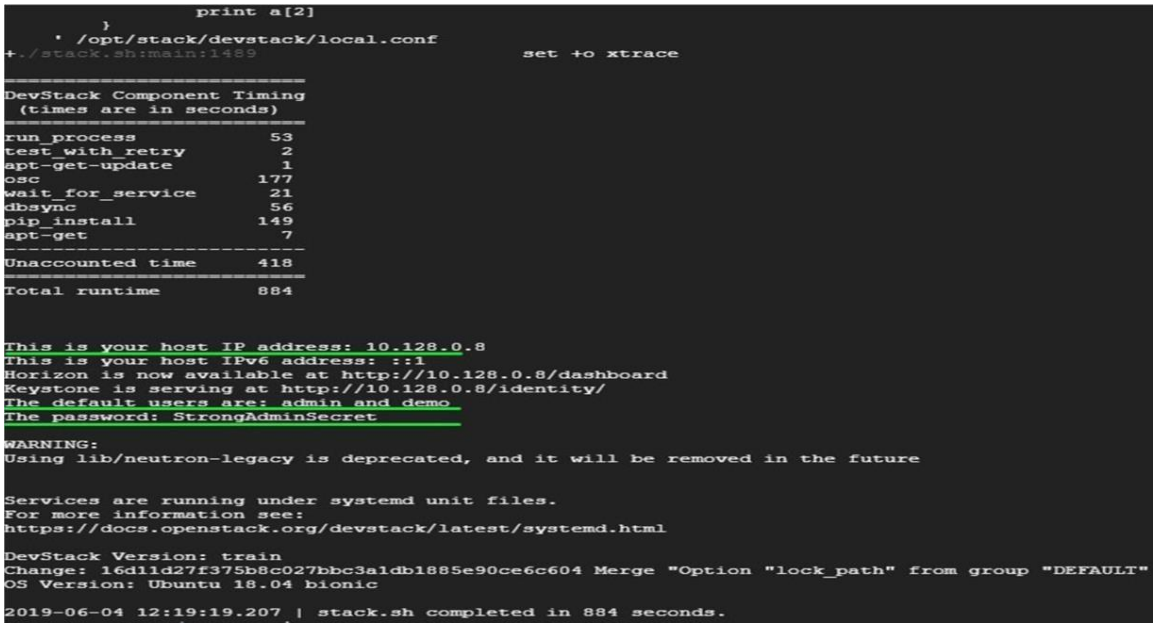
To commence the installation of OpenStack on Ubuntu 18.04, run the script below cont in devstack directory.

./stack.sh

The following features will be installed:

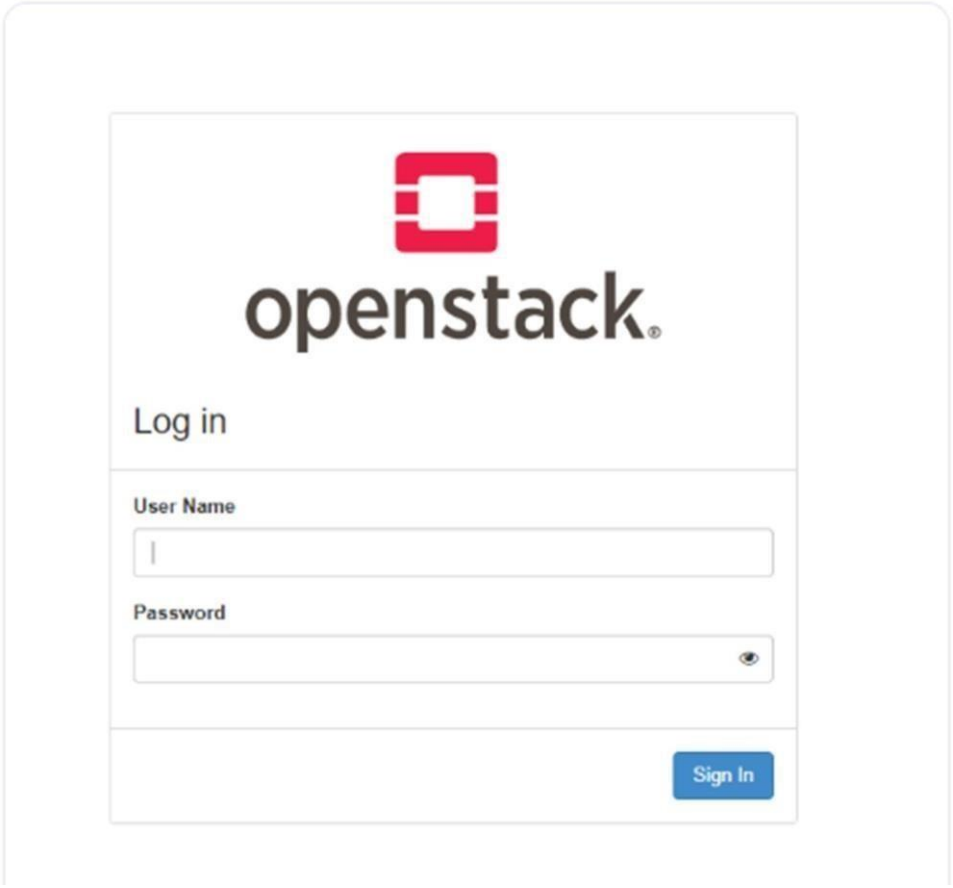
- **Horizon – OpenStack Dashboard**
- **Nova – Compute Service**
- **Glance – Image Service**
- **Neutron – Network Service**
- **Keystone – Identity Service**
- **Cinder – Block Storage Service**
- **Placement – Placement API**

The deployment takes about 10 to 15 minutes depending on the speed of your system internet connection. In our case, it took roughly 12 minutes. At the very end, you shoul output similar to what we have below.



This confirms that all went well and that we can proceed to access OpenStack via a we browser.

Step 6: Accessing OpenStack on a web browser
To access OpenStack via a web browser browse your Ubuntu's IP address as shown.
https://server-ip/dashboard This directs you to a login page as shown.



Enter the credentials and hit "Sign In" You should be able to see the Management console dashboard as shown below.