PRACTICAL - 1

Aim: Define a simple services like Converting Rs into Dollar and Call it from different platform like JAVA and .NET

Step 1: Start NetBeans8.0.2

Step 2: go to **file** -> **new project** -> **select java web** -> **web application**

Step 3: After that click on **Next** and give the **project name** and **finish.**

Step 4: **Right click on the project name page-> new-> webservice**

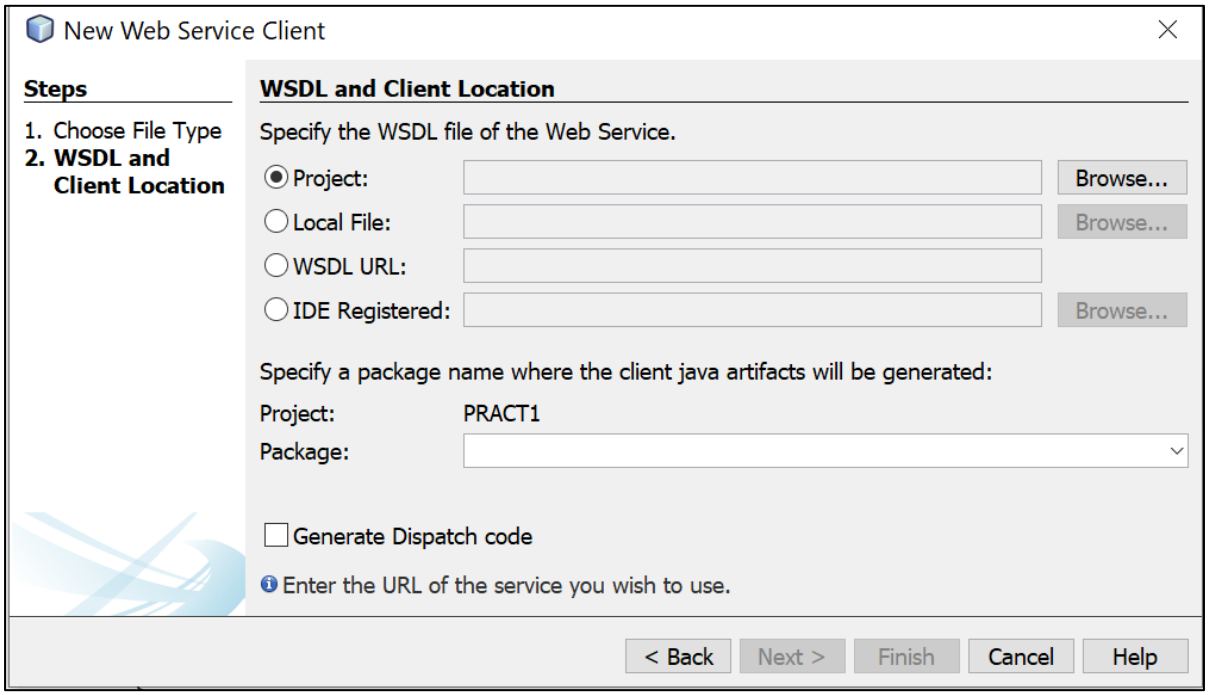
Give the **web service name (CurrencyConverter)** and **package name (server)**-> **finish**

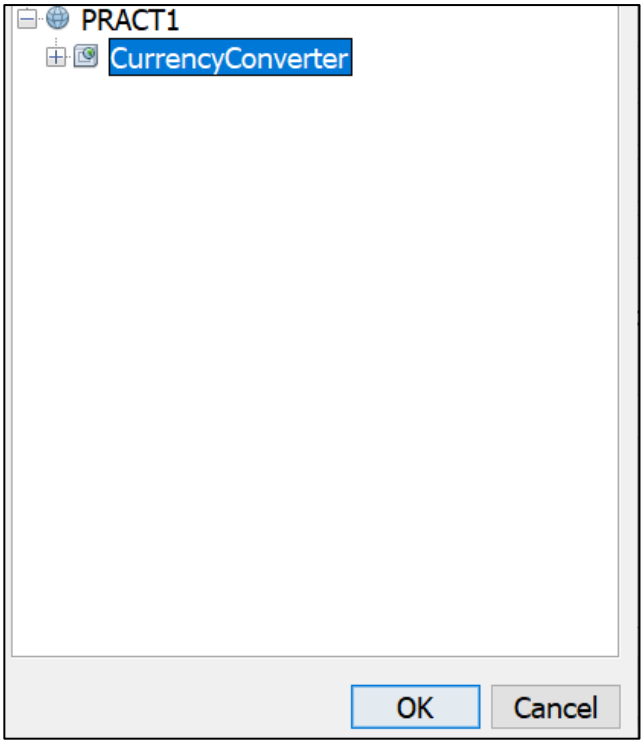
Step 5: **Right click on the web pages ->new -> jsp**

Give the **file name (input.jsp)** and finish it.

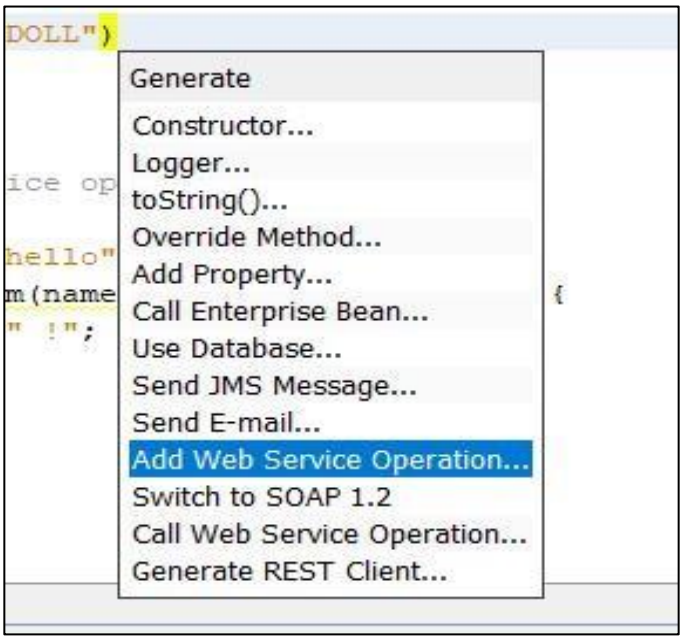
Step 6: Repeat step 5 and create another file (**output.jsp**).

Step 7: **Right click on the web page -> new -> web service client-> browse (for project name)**

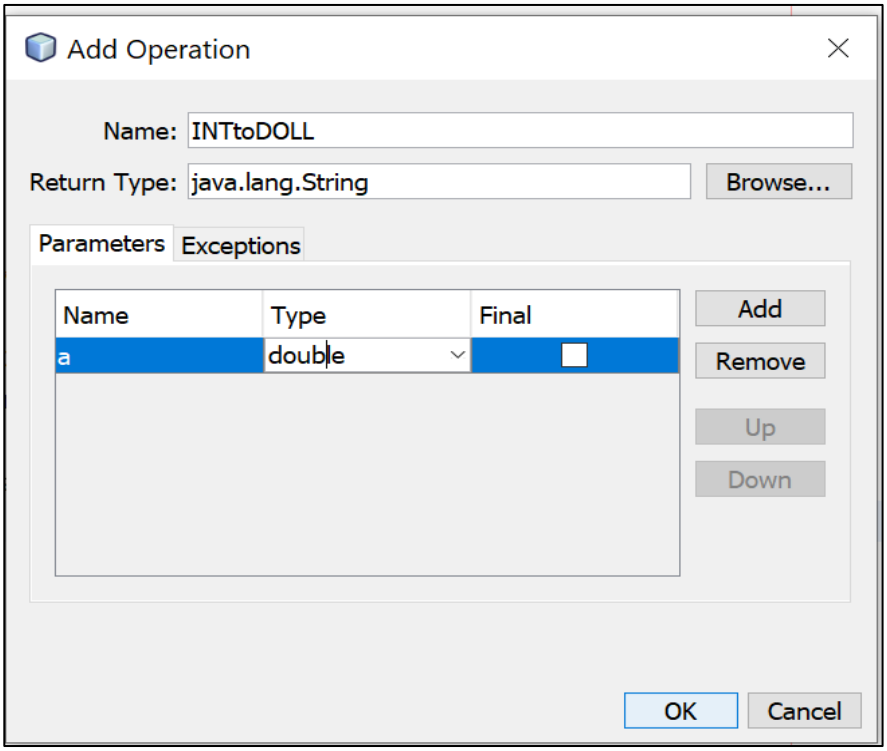




Step 8 : Go to java file **(CurrencyConverter)** and **right click on the screen -> insert code ->add web service operation**



Step 9: **Give the operation name-> click on add button ->give the name of variable and set the data type -> ok.**



CurrencyConverter.java

```
package server;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
@WebService(serviceName = "CurrencyConverter")
public class CurrencyConverter
{
    @WebMethod(operationName = "INRtoDOLL")
   public String INRtoDOLL(@WebParam(name = "a") double a)
   {
    return "INR"+a+"in DOLLAR:"+ (a/83.11);
    }
```

}

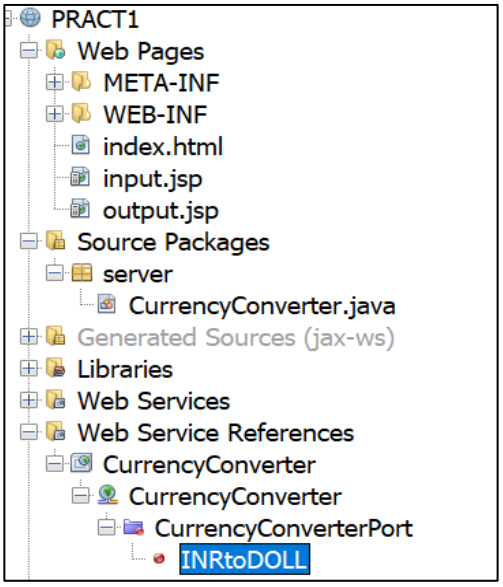
Client as Java

Step 10: Go to **input.jsp**

Code (**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>
<h1>Currency Converter</h1>
<form action="output.jsp">
ENTER CURRENCY IN INR:<input type="text" name="t1">
<br>
<input type="submit">
<input type="reset">
</form>
</body>
</html>
```

Step 11: open **output.jsp** . on the left side **web service reference** and select nestedly till the INRtoDOLL



Drag and Drop the INRtoDOLL In (output.jsp)

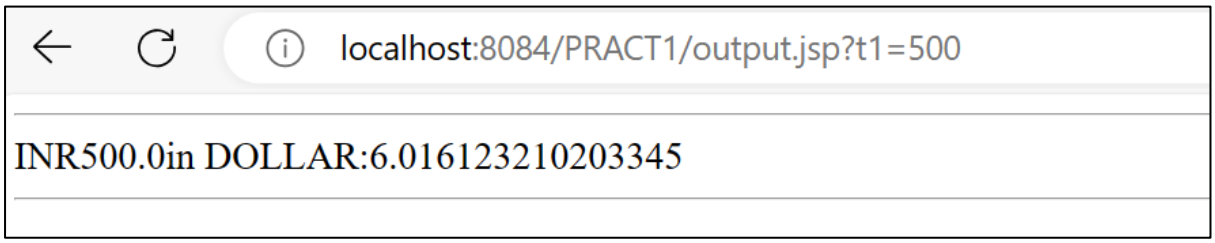
```
    <body>
            <%-- start web service invocation --%><hr/>
    <%
    try {
        client.CurrencyConverter_Service service = new client.CurrencyConverter_Service();
        client.CurrencyConverter port = service.getCurrencyConverterPort();
         // TODO initialize WS operation arguments here
        double a = Double.parseDouble(request.getParameter("t1"));
        // TODO process result here
        java.lang.String result = port.inRtoDOLL(a);
        out.println(result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>

    </body>
</html>
```

Step 12: Right click on the **project name** and **deploy** it

Step 13: Right click on the **input.jsp** and select **run file**

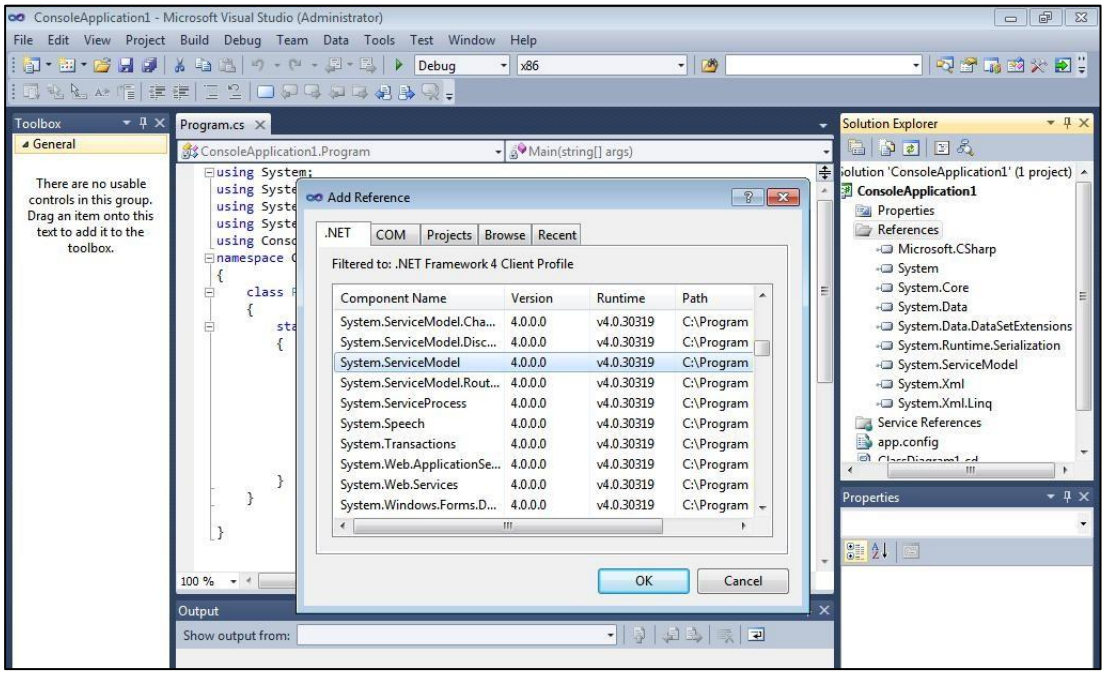
Output:


localhost:8084/PRACT1/input.jsp
Currency Converter
ENTER CURRENCY IN INR: 500
Submit
Reset

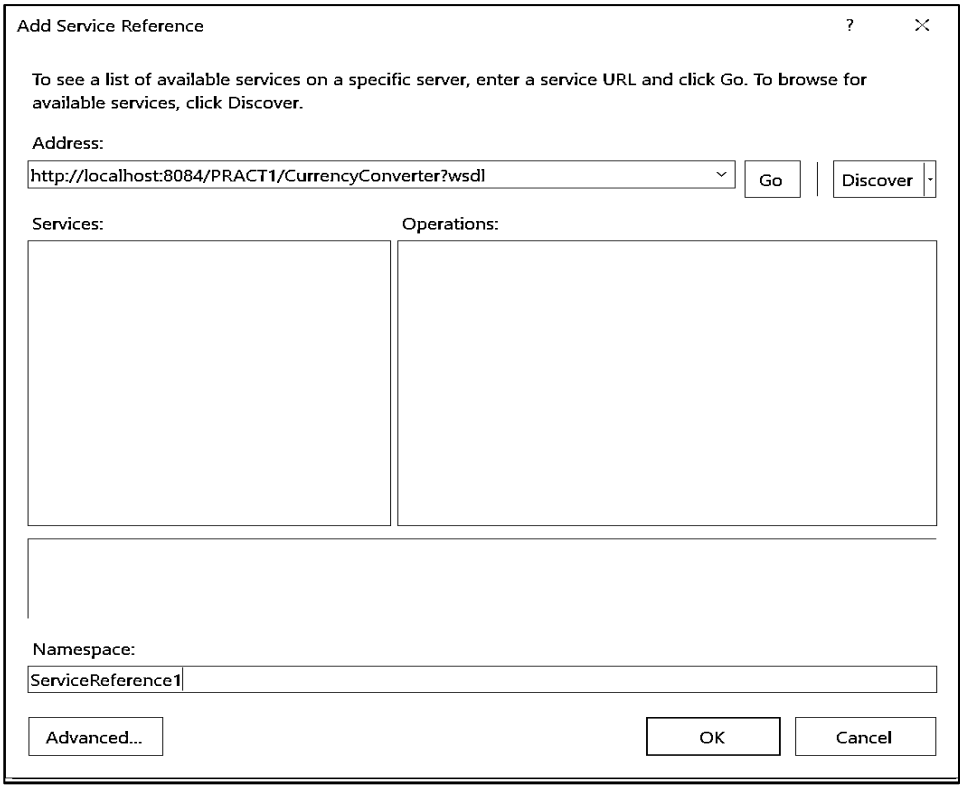

localhost:8084/PRACT1/output.jsp?t1=500
INR500.0in DOLLAR:6.016123210203345

Client as .Net:

File->new project-> c#-> Console Application->ok
Solution explorer-> Project name -> Right Click -> Add References -> .NetTab -> Select System.ServiceModel -> OK



Solution explorer-> Project Name-> Right Click -> Add Service References -> Copy WSDL Path from browser -> Go ->Expand New Web Service ->OK



Code (program.cs):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using currencyconvert.ServiceReference1;
namespace currencyconvert
{
    class Program
    {
        static void Main(string[] args)
        {
            CurrencyConverterClient client = new CurrencyConverterClient();
            Console.WriteLine("Enter the Currency in INR:");
            double d = double.Parse(Console.ReadLine());
            Console.WriteLine(client.INRtoDOLL(d));
            Console.ReadLine();
            Console.WriteLine();
        }
    }
}
```

Output:

Select file:///C:/Users/Comp-28/Documents/Visual Studio 2012/Projects/currencyconvert/currency

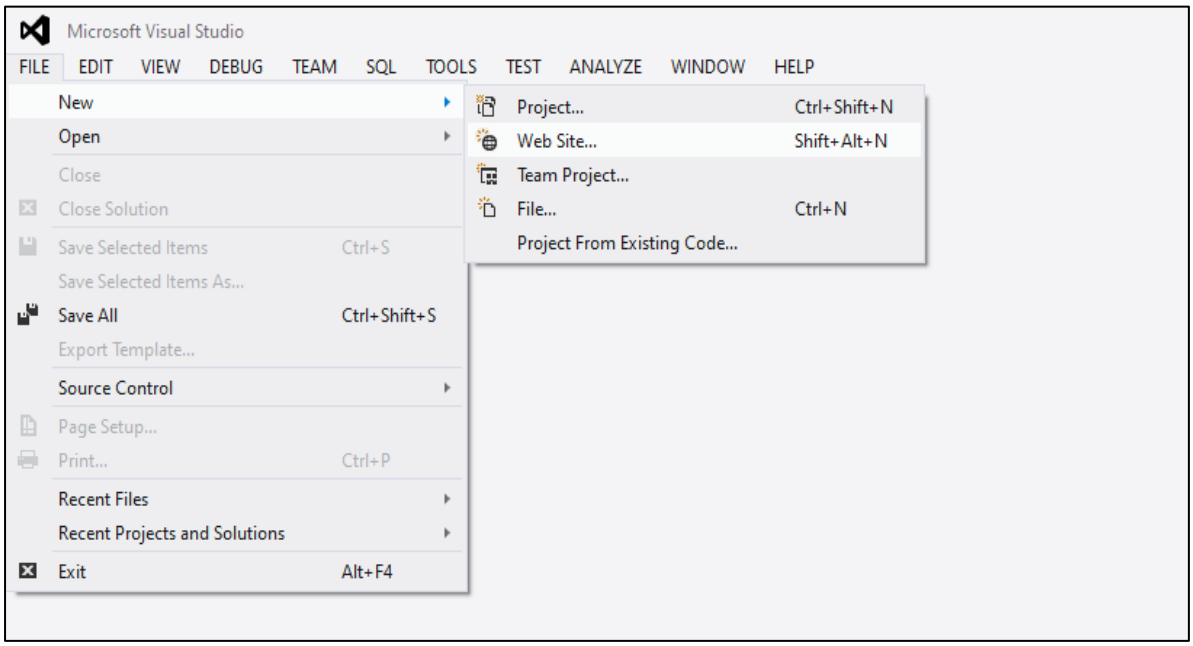
```
Enter the Currency in INR:
500
INR500.0in DOLLAR:6.016123210203345
```

PRACTICAL - 2

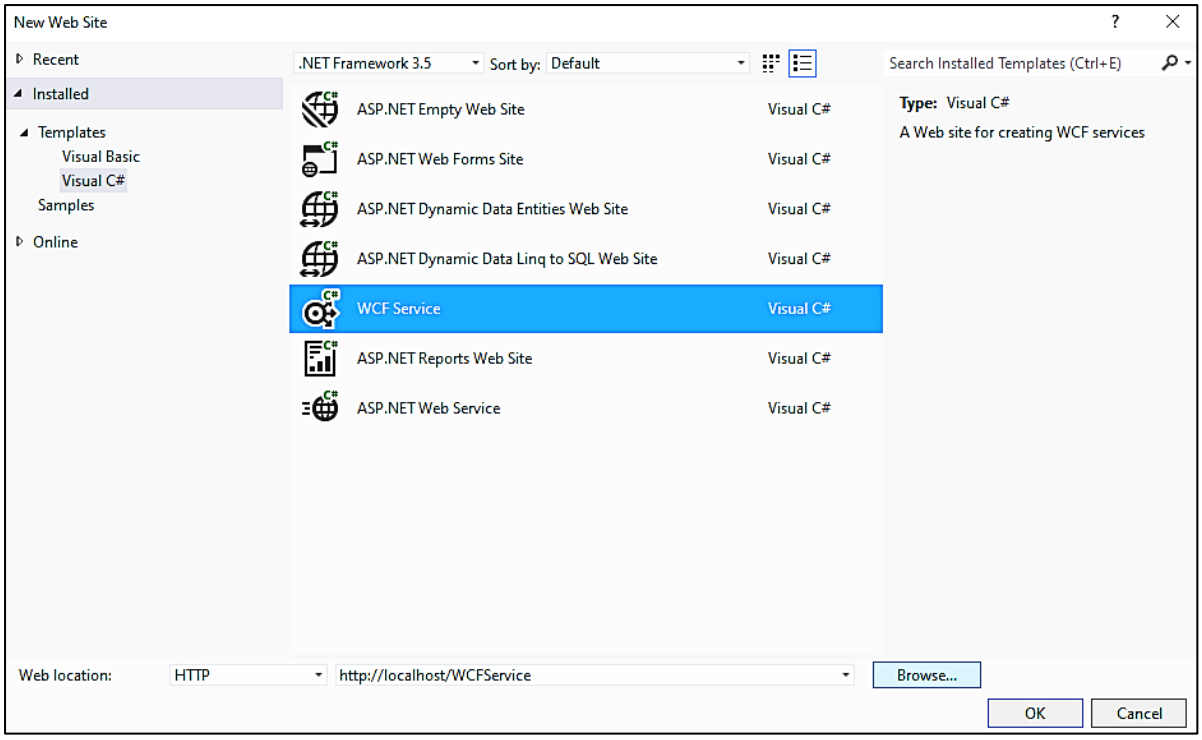
Aim: Create a Simple SOAP service

Step 1: Start **Visual Studio 2012** (Run as administrator)

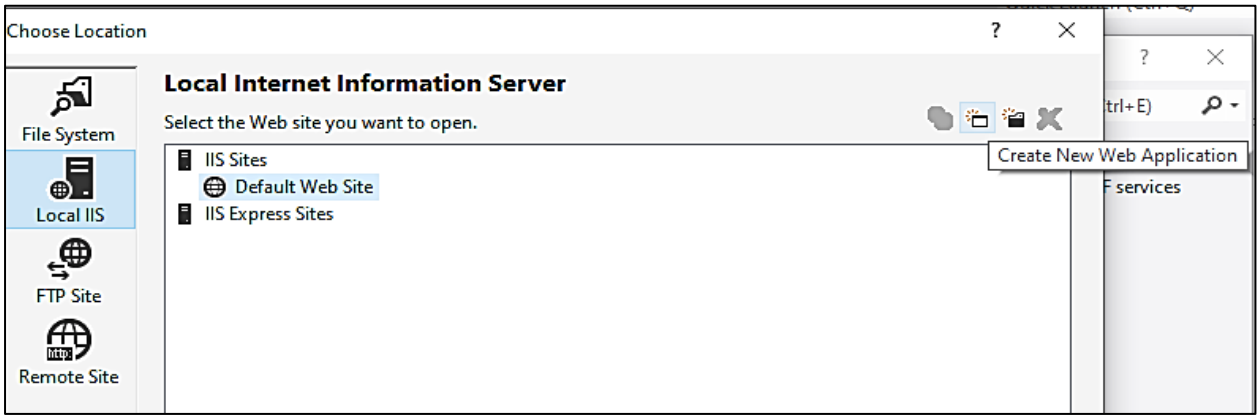
Step 2: Go to **File -> New ->Web Site**



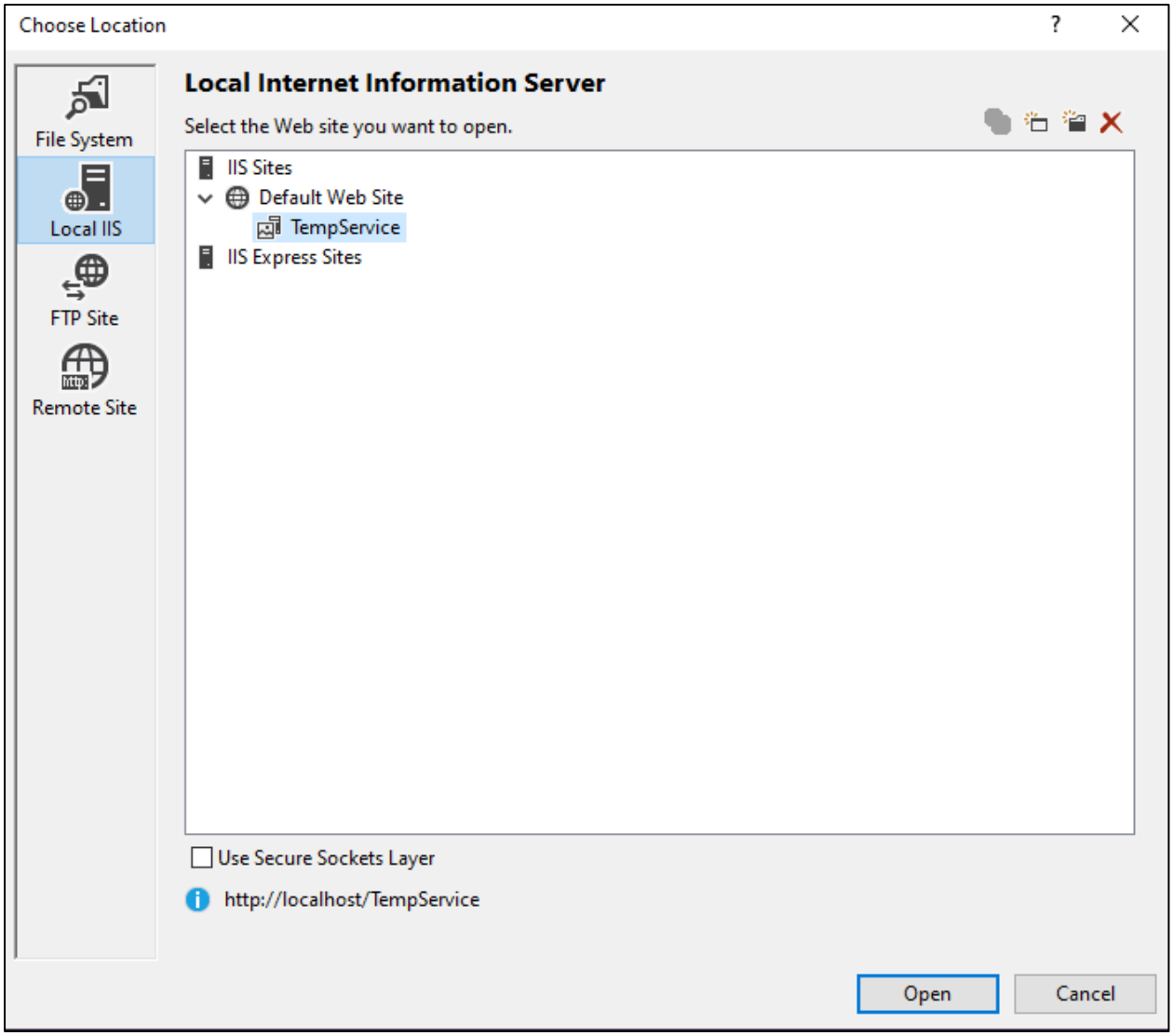
Step 3: Select **WCF Service** Visual C# then click on Browse



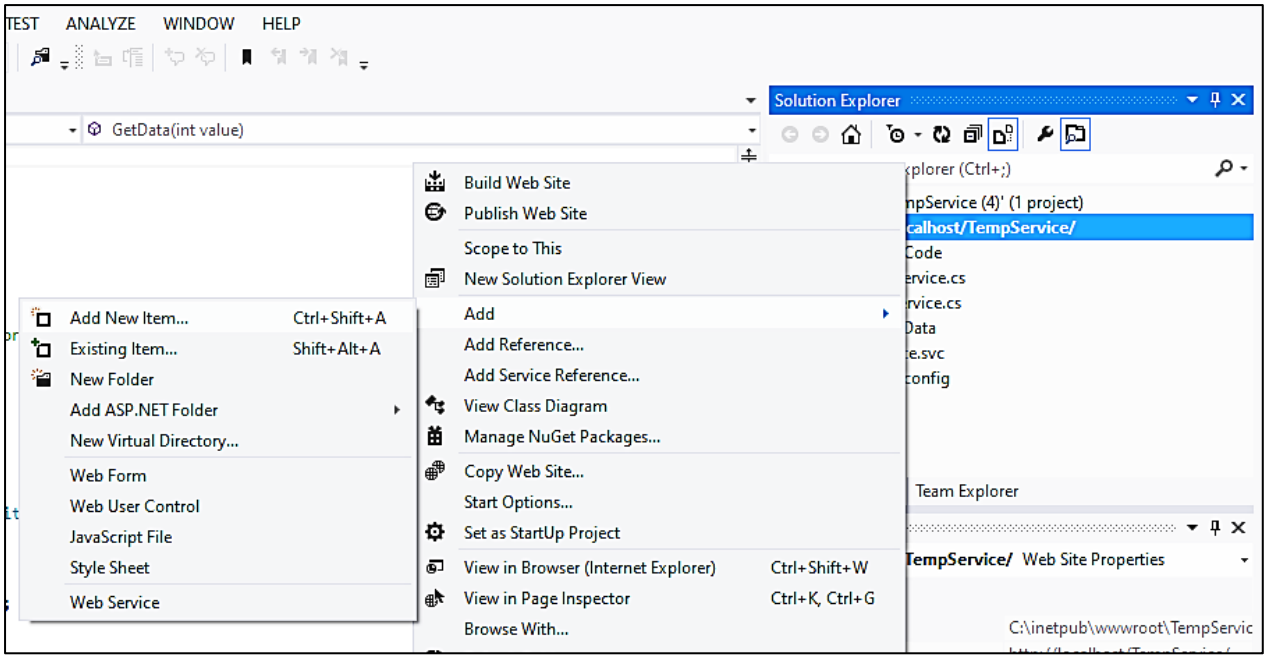
Step 4: **Local IIS -> create New Web Application**



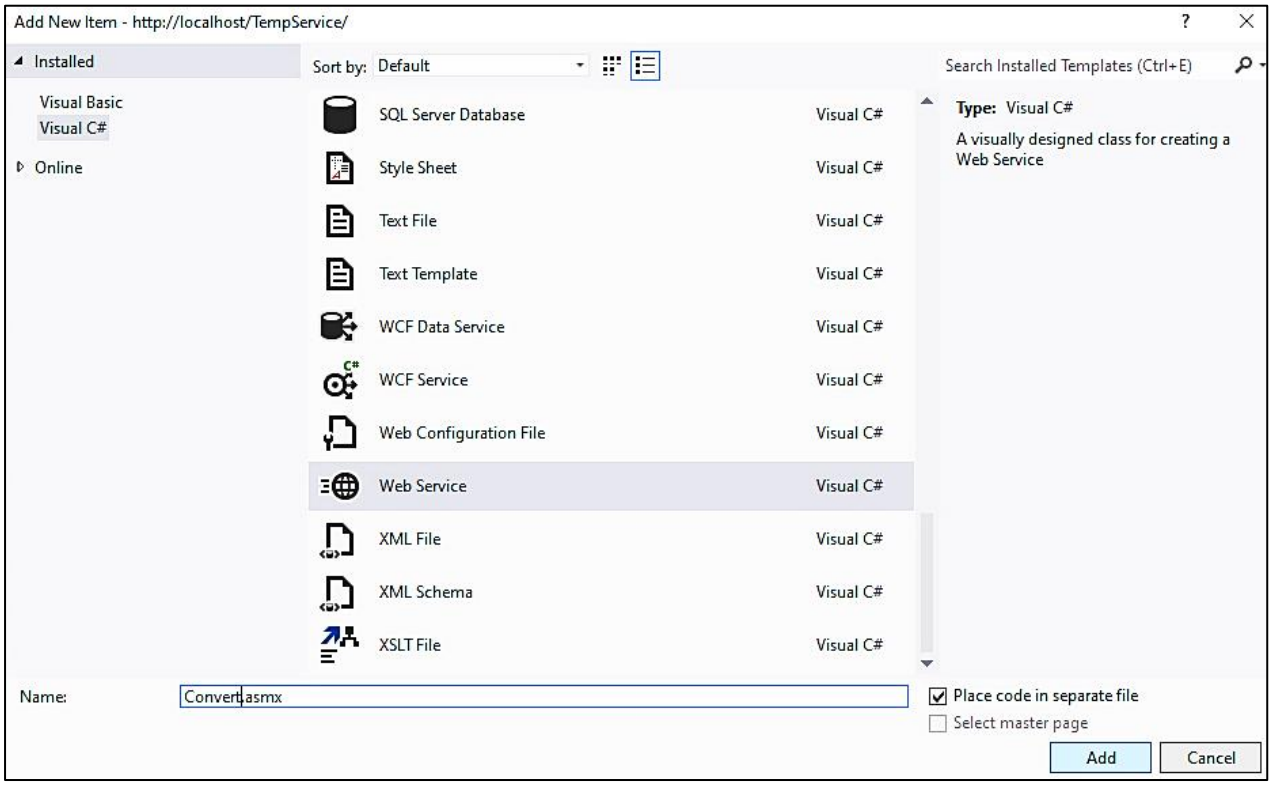
Give Name of New Web Application e.g (**TempService**) ->**open**



Step 5: In **solution Explorer ->Right click on http://localhost/TempService- >Add ->Add New item**



Step 6: Select **Web Service** and give name of the web service (**Convert.asmx**) -> click on **Add**



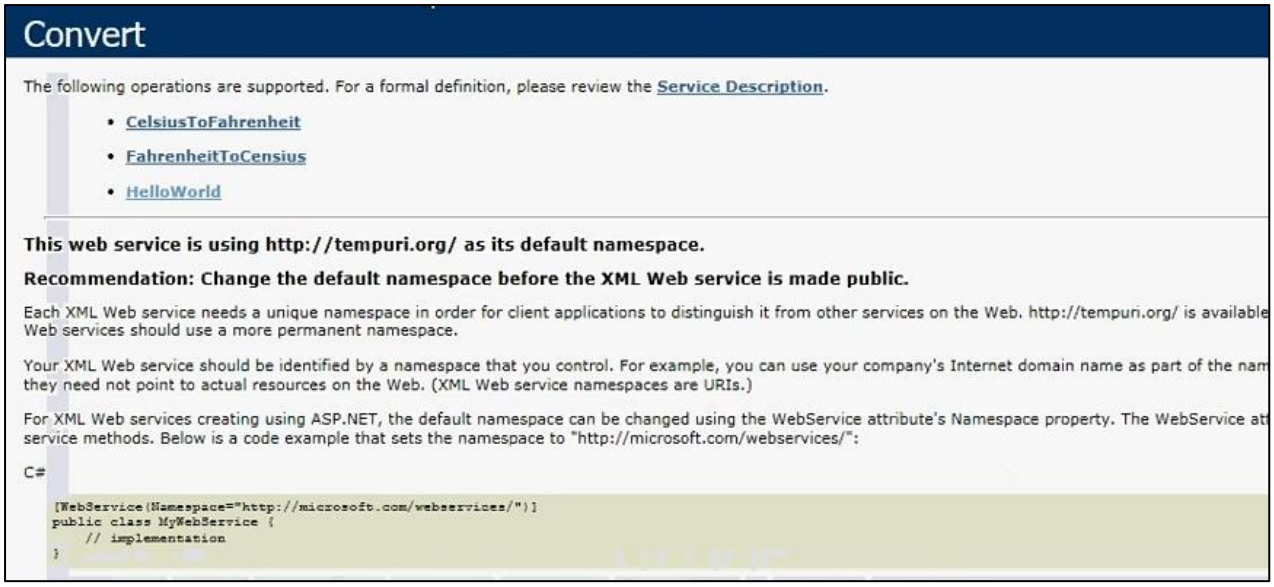
Code: Convert.cs

[WebMethod}

```
public double FahrenheitToCelsius(double Farhenhit)
{
   return ((Fahrenheit – 32 ) * 5)/9;
}
[WebMethod}
public double CelsiusToFahrenheit(double Celsius)
{
   return ((Celsius * 9 ) /5)+32;
}
```

```
Internet Explorer   Debug   Any CPU
Convert.cs
http://localhost/TempService/   Convert
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncom
// [System.Web.Script.Services.ScriptService]
1 reference
public class Convert : System.Web.Services.WebService {

    0 references
    public Convert () {

        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
    0 references
    public string HelloWorld() {
        return "Hello World";
    }

    [WebMethod]
    0 references
    public double FahrenheitToCensius(double Fahrenheit)
    {
        return ((Fahrenheit - 32) * 5) / 9;
    }

    [WebMethod]
    0 references
    public double CelsiusToFahrenheit (double Celsius)
    {
        return ((Celsius * 9) / 5) + 32;
    }
}
```

Step 7: Run Web Service **Convert.cs**



Convert

The following operations are supported. For a formal definition, please review the Service Description.

- CelsiusToFahrenheit
- FahrenheitToCensius
- HelloWorld

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available
Web services should use a more permanent namespace.

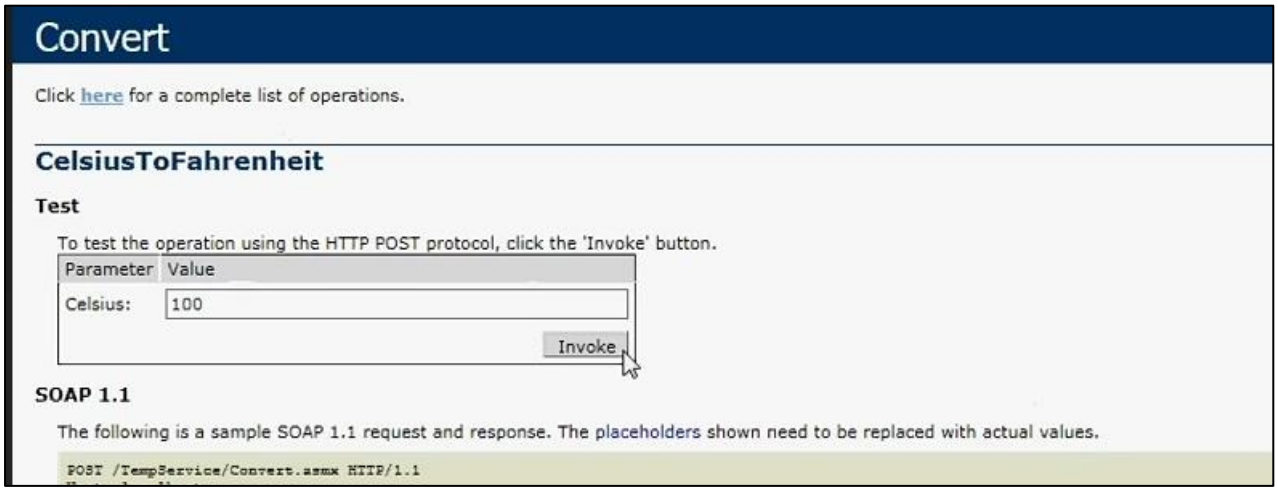
Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the nam
they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService att
service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Test Web Service by calling its web Methods



Convert

Click here for a complete list of operations.

CelsiusToFahrenheit

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

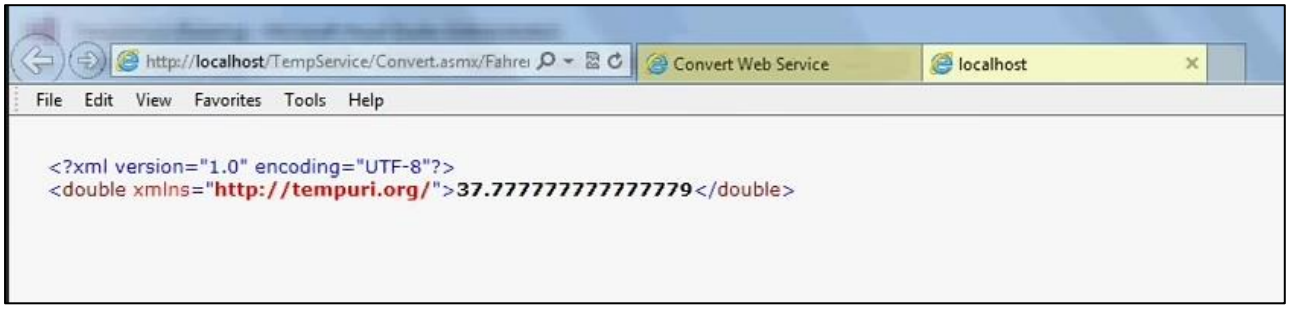
Parameter	Value
Celsius:	100

Invoke

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /TempService/Convert.asmx HTTP/1.1
```



http://localhost/TempService/Convert.asmx/Fahrei

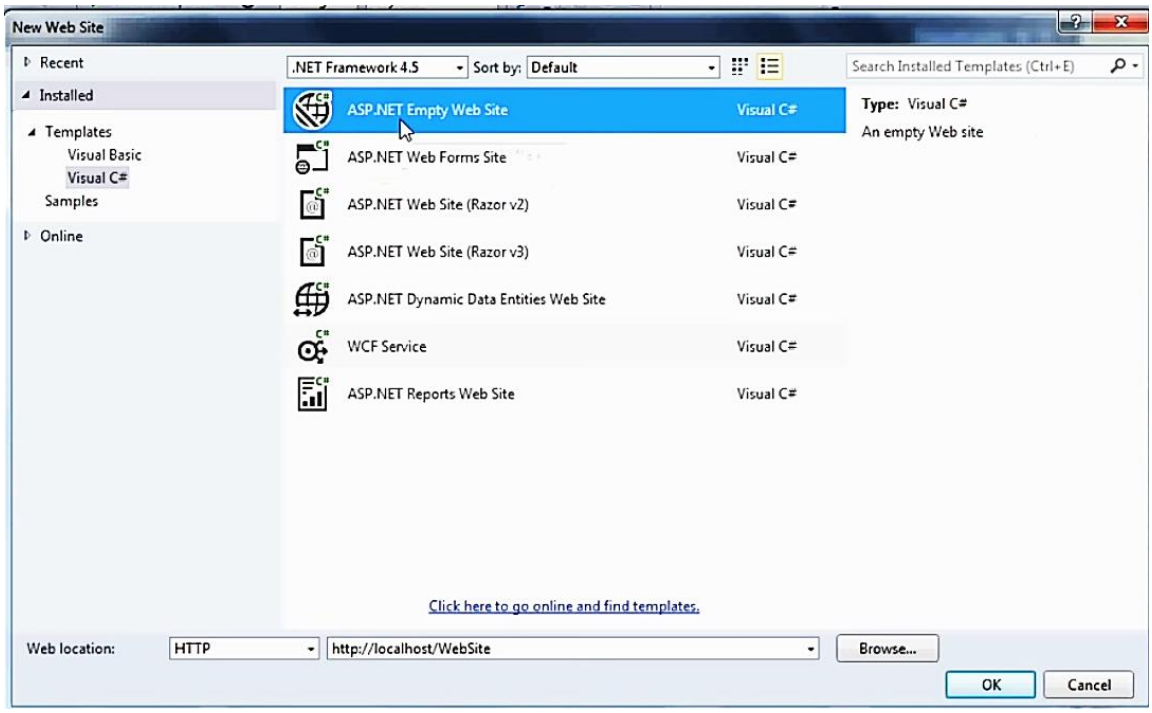
Convert Web Service | localhost

File Edit View Favorites Tools Help

```
<?xml version="1.0" encoding="UTF-8"?>
<double xmlns="http://tempuri.org/">37.777777777777779</double>
```

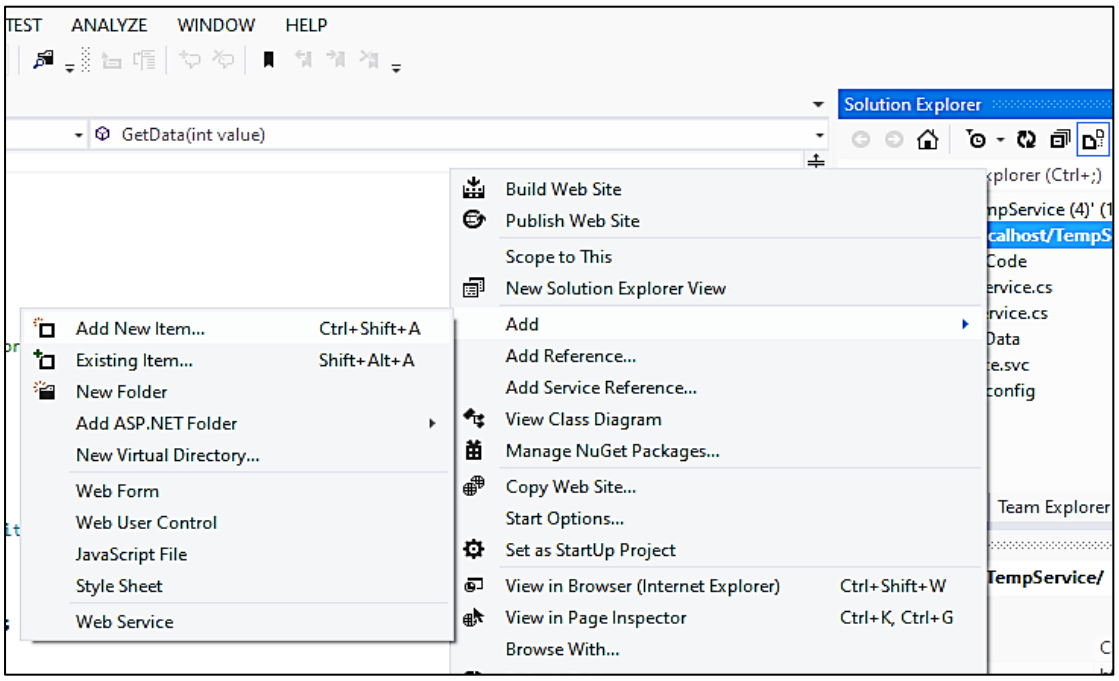
We consume this web Service in website

Step 8: Go to **File -> New ->Web Site** then Select **ASP.NET Empty Web Site**

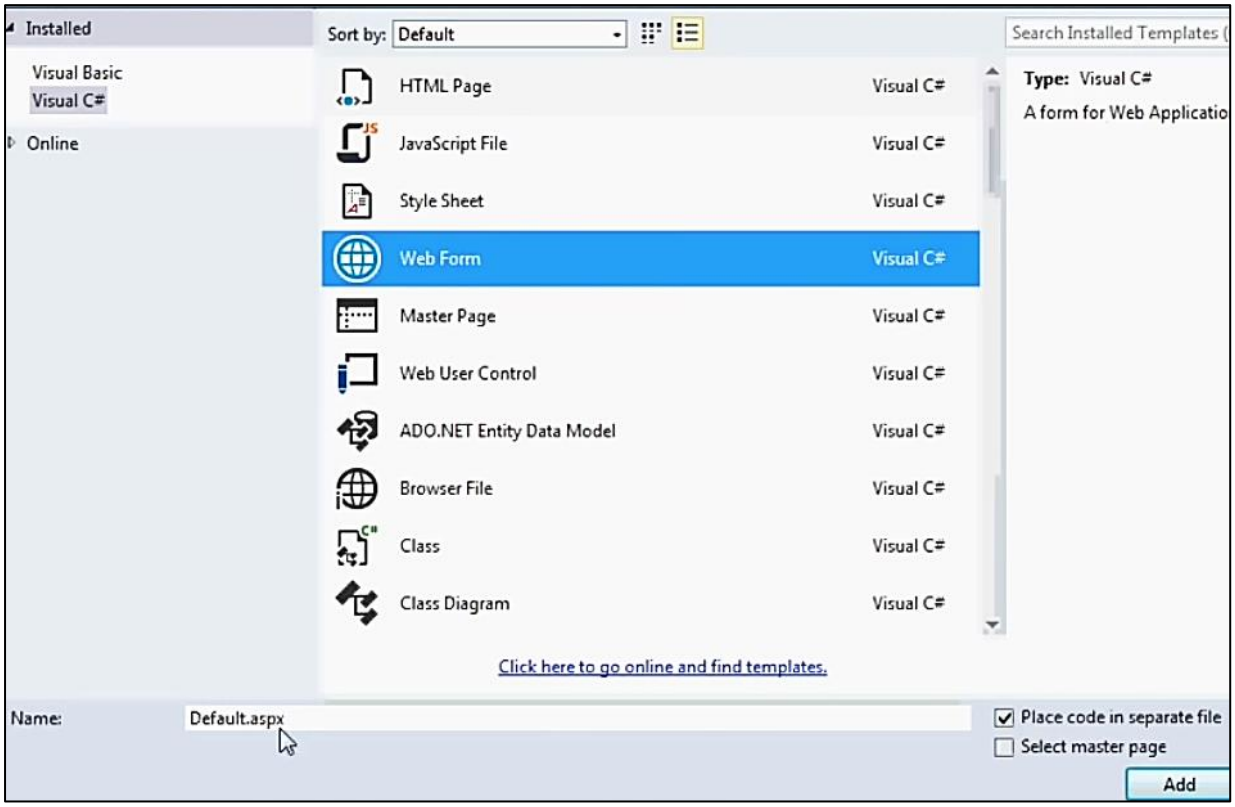


Step 9: **Local IIS -> create New Web Application a**nd Give Name of Web Application (**TempSite)**

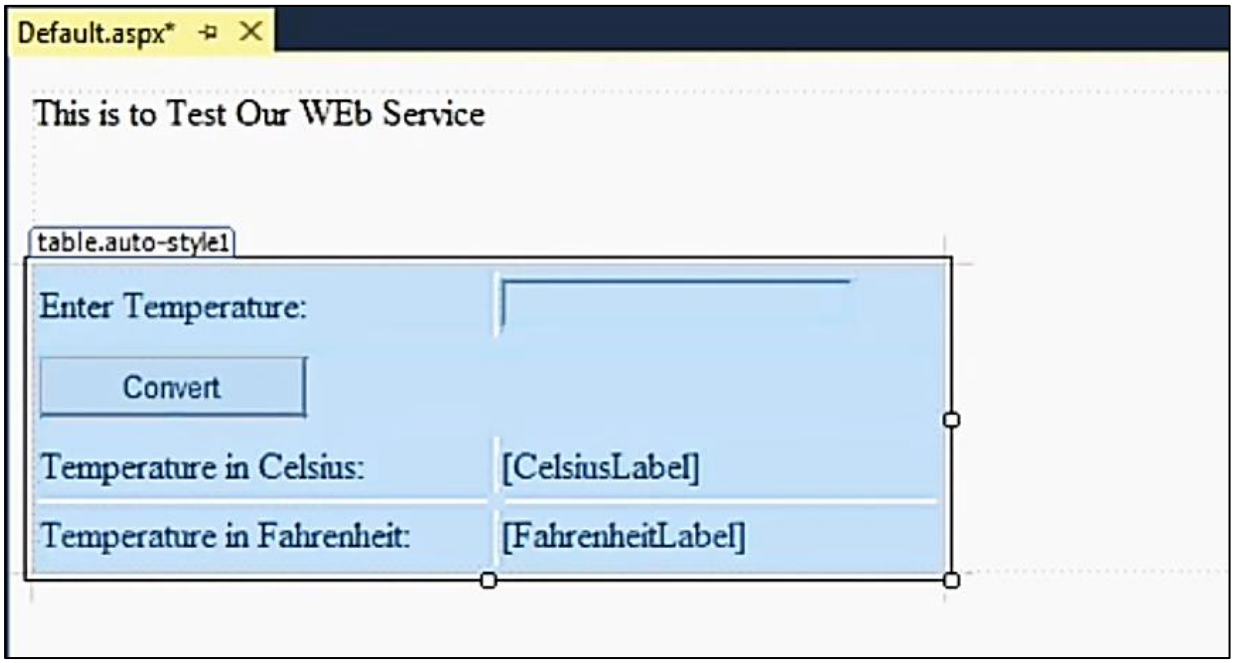
In **solution Explorer ->Right click on http://localhost/TempSite- >Add ->Add New Item**



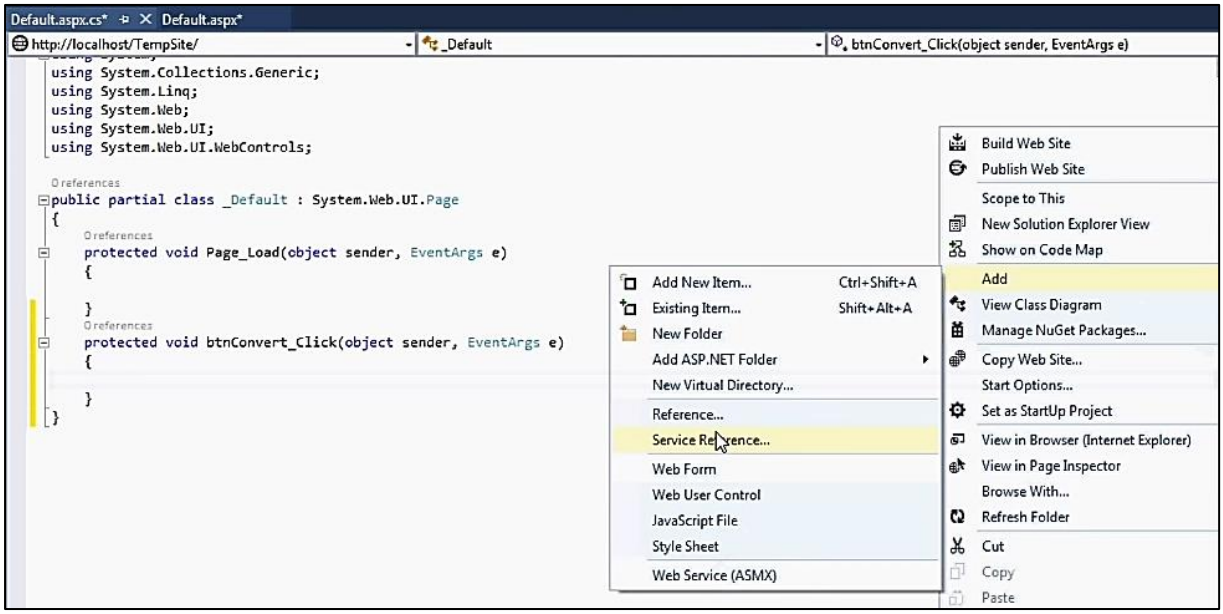
Step 10: Select-> **Web Form->** click on **Adds**



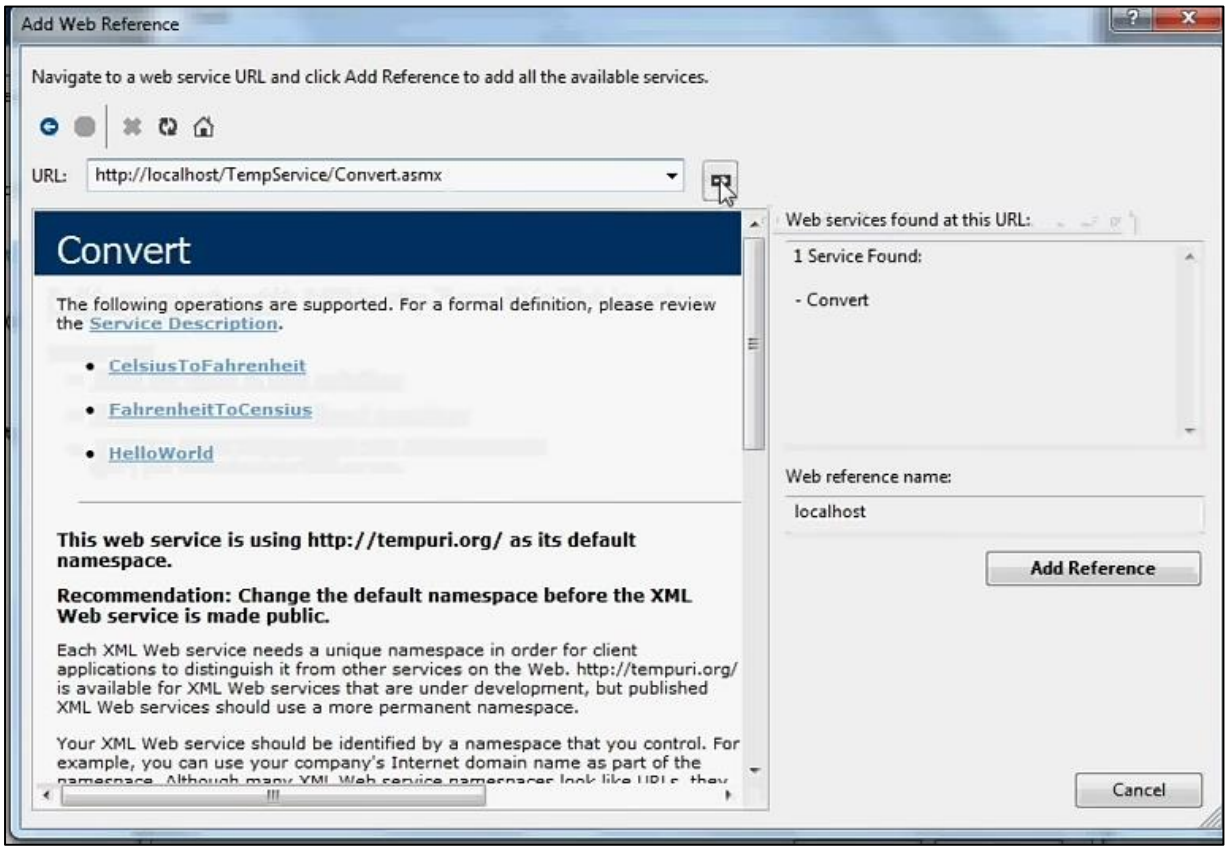
Step 10: Design the page (**Default.aspx**)



Step 11: In **solution Explorer ->Right click on http://localhost/TempSite- >Add -> Service Reference**



Step 12: Enter the url of Web Service (**Convert.asmx**)



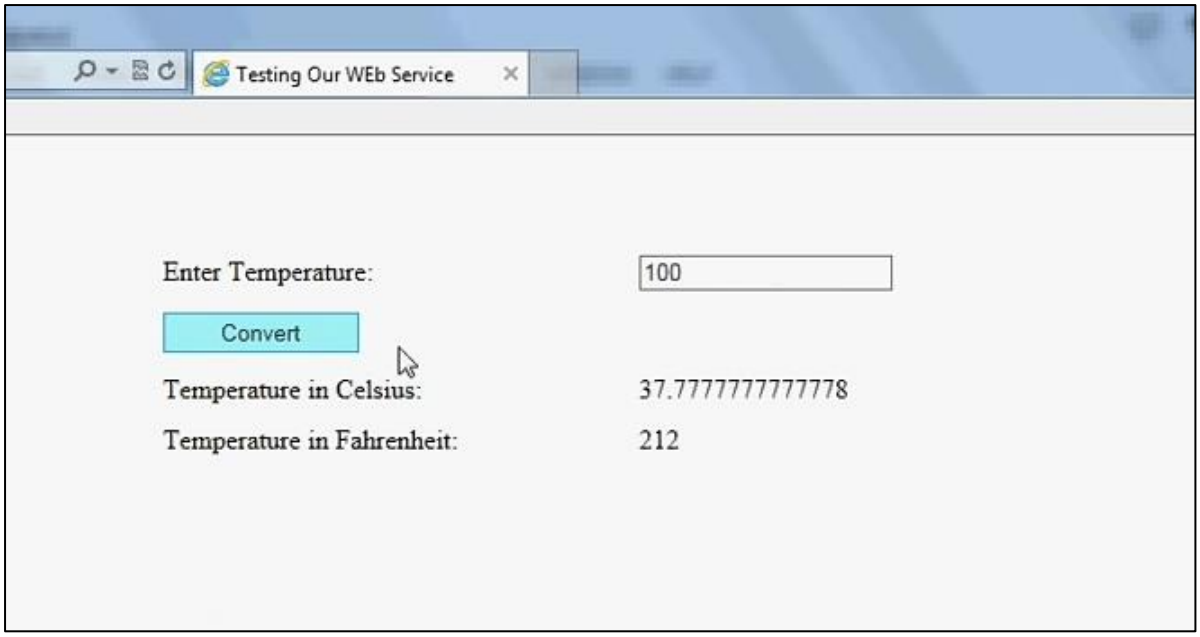
Step 13: Click on convert button (created in design page) to write code

User click on convert button to get the result of their entered input values

```
Default.aspx.cs*   Default.aspx*
http://localhost/TempSite/                    _Default

using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

0 references
public partial class _Default : System.Web.UI.Page
{
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    0 references
    protected void btnConvert_Click(object sender, EventArgs e)
    {
        localhost.Convert wsConvert = new localhost.Convert();

        double temperature = Convert.ToDouble(TemperatureTextbox.Text);

        FahrenheitLabel.Text = wsConvert.CelsiusToFahrenheit(temperature).ToString();

        CelsiusLabel.Text = wsConvert.FahrenheitToCensius(temperature).ToString();

    }
}
```

Step 14: Run the file

Output:


Testing Our WEb Service
Enter Temperature:
100
Convert
Temperature in Celsius:
37.7777777777778
Temperature in Fahrenheit:
212

PRACTICAL - 3

Aim: Create a Simple REST Service

Creating a simple REST service in Python can be done using a lightweight framework like Flask. Flask is a micro web framework that is easy to use and well-suited for building small to medium-sized web applications, including RESTful services.

A basic example of a simple REST service using Flask:

Install Flask:

pip install flask

Create a file named **app.py**

Code:

```
from flask import Flask,request
app = Flask(__name__)
products= [
    {
        'id': 1,
        'title': 'iphone',
        'price': 456
    },
    {
        'id': 2,
        'title': 'Android',
        'price': 587
    }
]


@app.get("/products")
def get_products():
    return{"products":products}

@app.post("/products")
def add_product():
    request_data=request.form.to_dict()
    new_product={"id":request_data["id"],"title":request_data["title"],
"price":request_data["price"]}
    products.append(new_product)
    return  new_product
```

```
@app.get("/product/<int:id>")
def get_specific_product(id):
    for product in products:
        if product["id"]==id:
            return product
    return {"message": "product not found"},404
app.run()
```

Output:

First we have to run flask for this

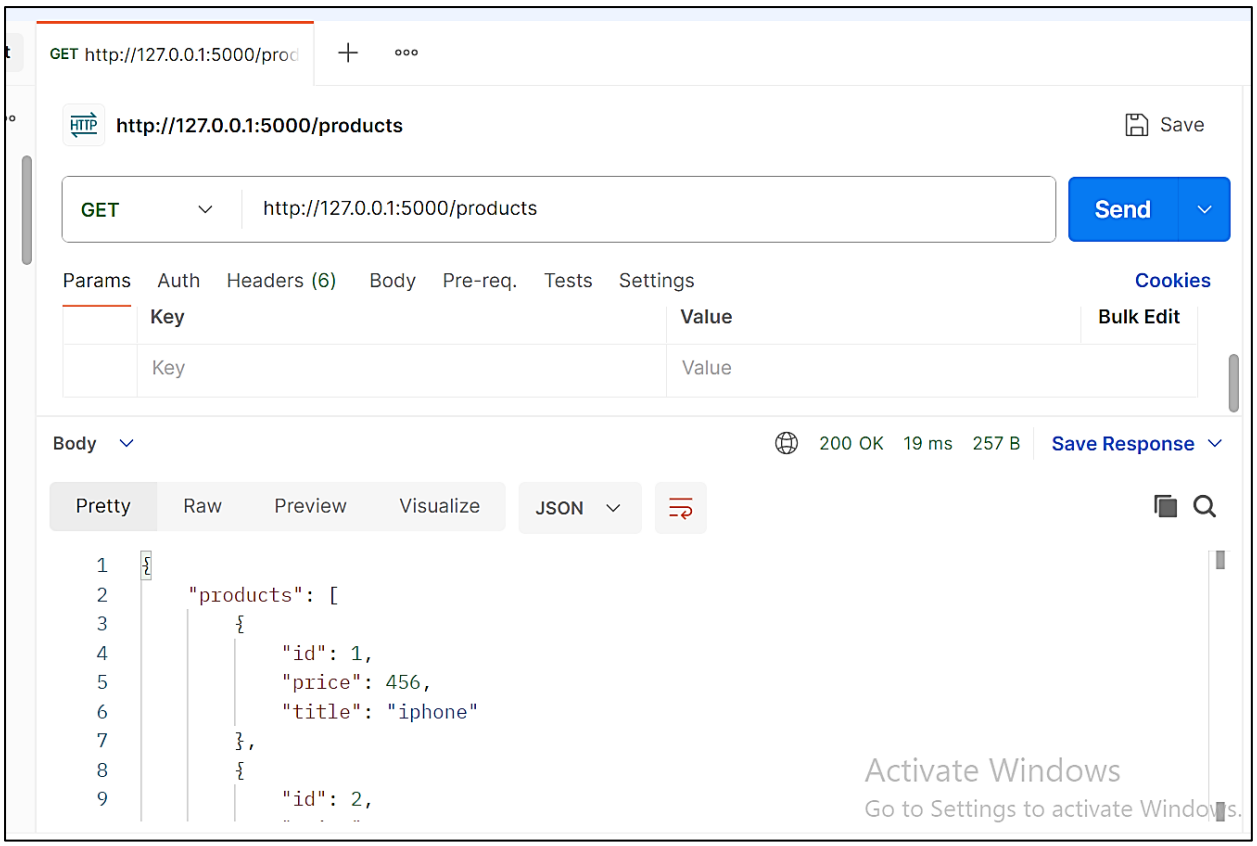
Go to New Terminal ->Type Command (**flask run**)

```
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deploymen
t. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

We can use **Postman /crul** to test the functionality.

<u>To View all the products</u>

send a **GET** request to **http://localhost:5000/products**

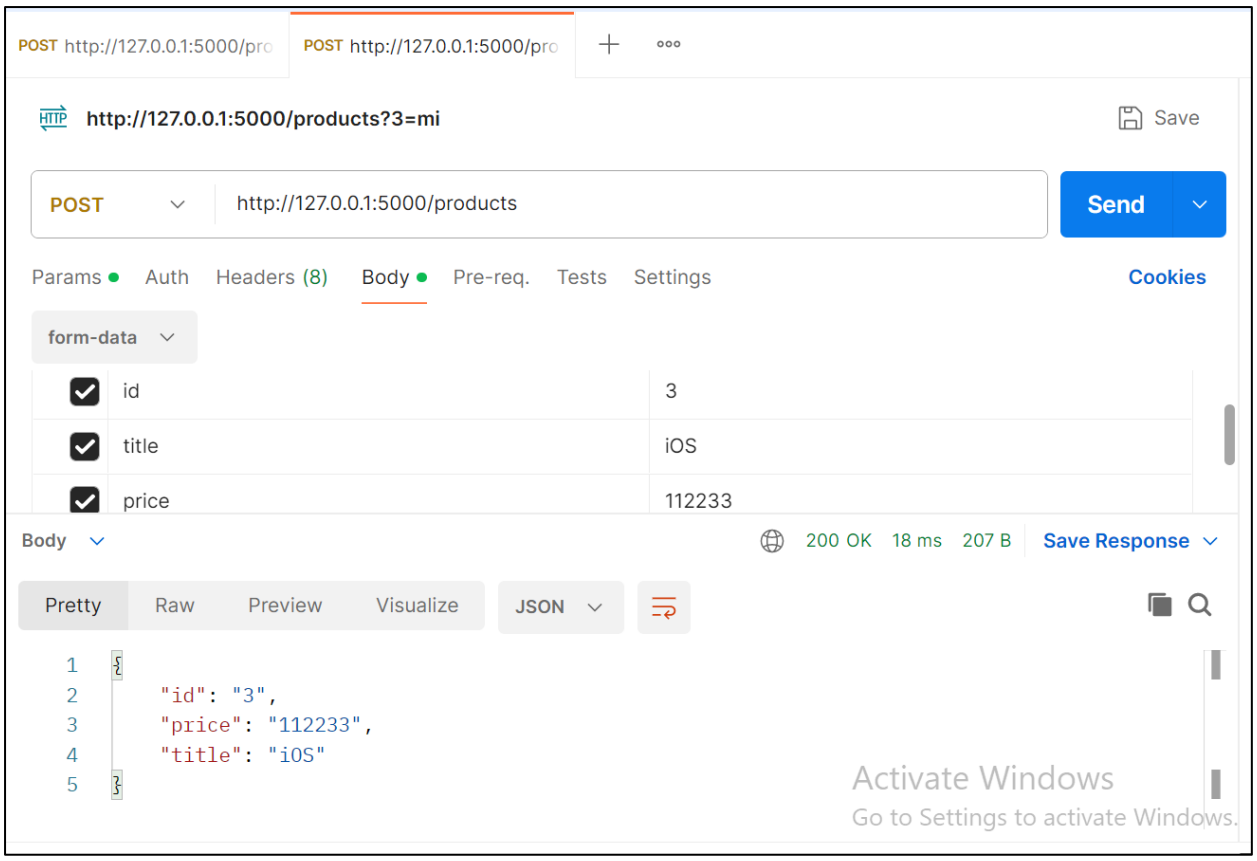


<u>For adding New Products</u>

Before sending request click on **Body ->** from dropdown list select **form-data**

Enter details of a new product by Key and Value

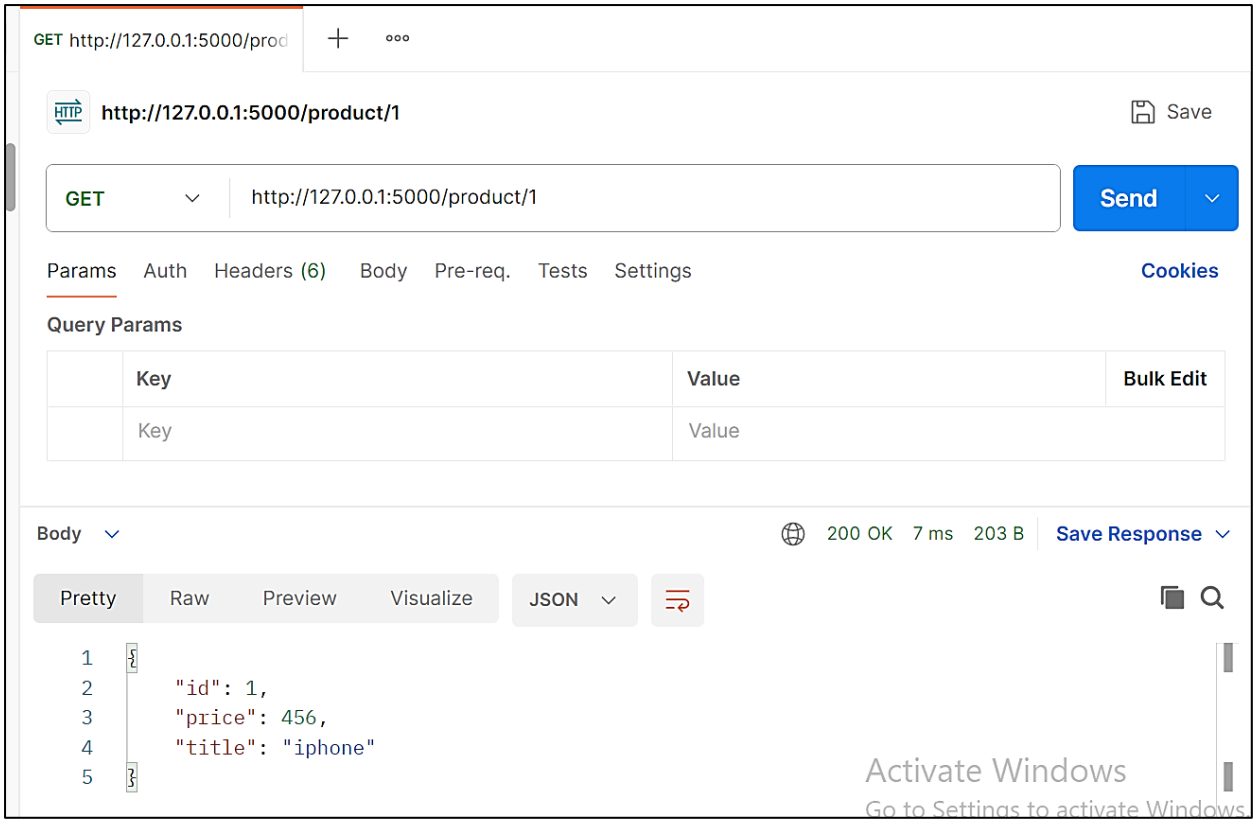
send a **POST** request to **http://localhost:5000/products**



<u>To view details of a particular product</u>

send a **GET** request to **http://localhost:5000/product/1**

(here **1** is **id): for first product details**


GET http://127.0.0.1:5000/prod
http://127.0.0.1:5000/product/1
Save
GET
http://127.0.0.1:5000/product/1
Send
Params
Auth
Headers (6)
Body
Pre-req.
Tests
Settings
Cookies
Query Params
Key
Value
Bulk Edit
Key
Value
Body
200 OK
7 ms
203 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"price": 456,
"title": "iphone"
}
Activate Windows
Go to Settings to activate Windows.

PRACTICAL - 4

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

Step 1: Start NetBeans8.0.2

Step 2: Go to **file** -> **new project** -> **select java web** -> **web application**

Step 3: After that click on **Next** and give the **project name** and **finish.**

Step 4: **Right click on the web pages ->new -> jsp**

Give the **file name (in.jsp)** and finish it.

Step 5: Repeat step 4 and create another file (**index.jsp**).

in.jsp

code:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>JSP Page</title>
</head>
<body>
<form action="index.jsp">
Enter latitude:<input type="text" name="t1" />
Enter longitude:<input type="text" name="t2" />
<input type="submit" >
</body>
</html>
```

Before running the application, we need the Google API key

(We use Keyless-Google-Maps-API for performing this practical)

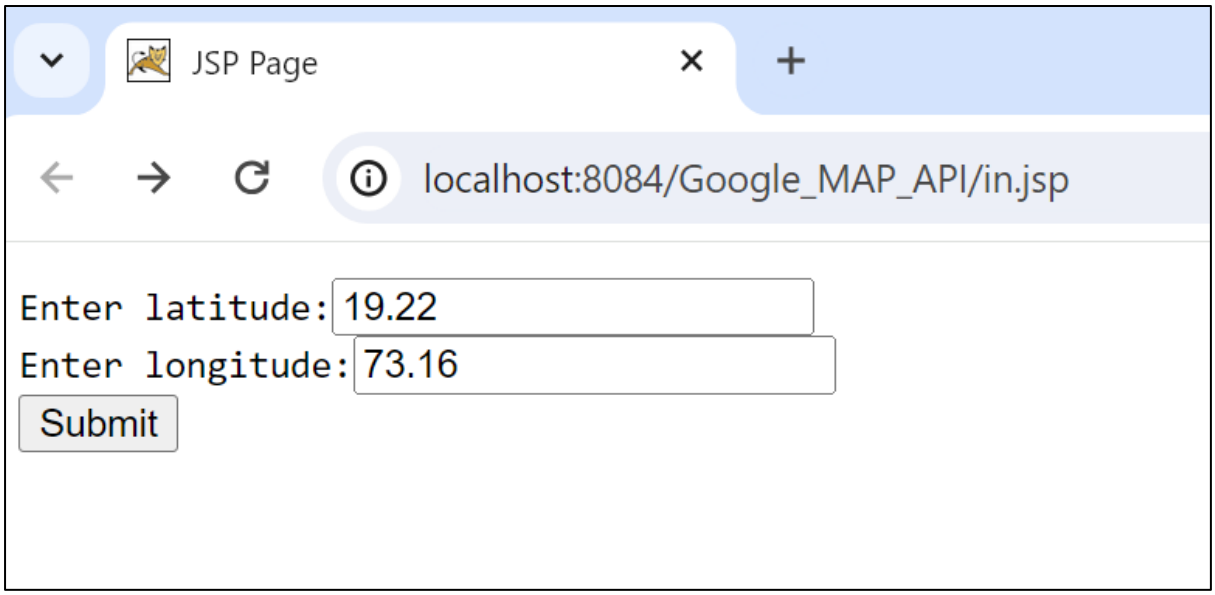
index.jsp

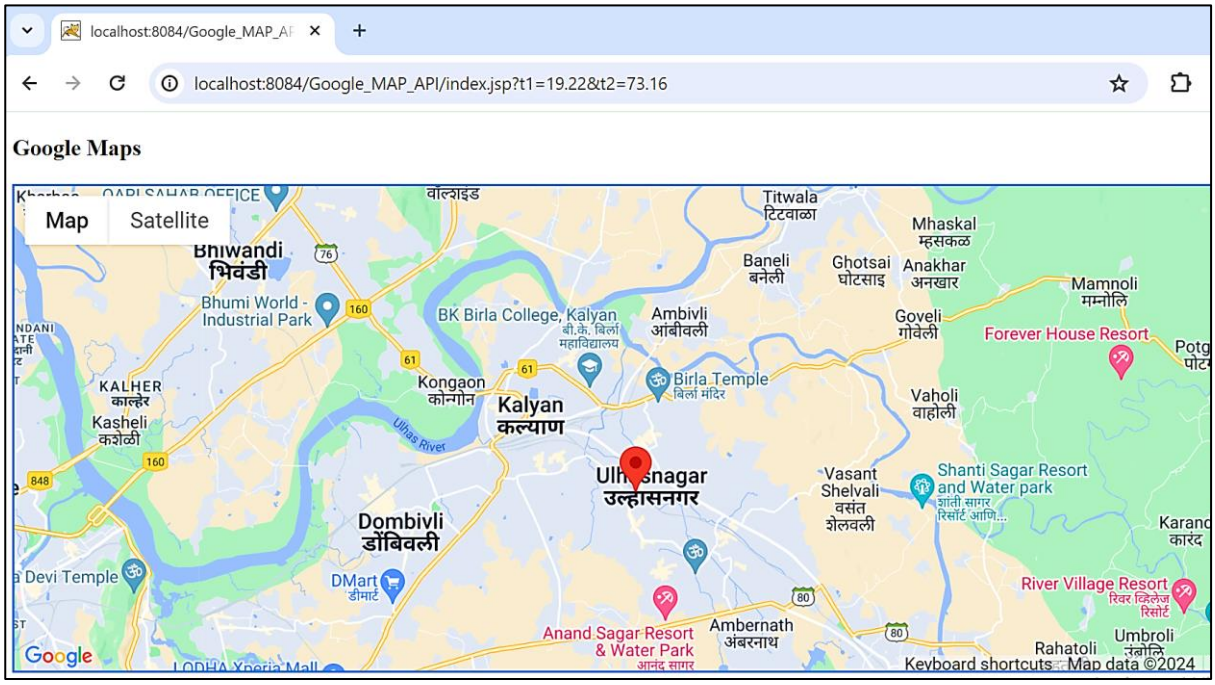
code:

```
<%@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: 10, center: info
});
var marker = new google.maps.Marker({
position: info, map: map
});
```

```
}
</script>
<script src="https://cdn.jsdelivr.net/gh/somanchiu/Keyless-Google-Maps-API@v6.6/mapsJavaScriptAPI.js"
async defer></script>
</body>
</html>
```

Output:





PRACTICAL - 5

Aim: Installation and Configuration of virtualization using KVM.

Virtualization is software that separates physical infrastructures to create various dedicated resources. It is the fundamental technology that powers cloud computing.

The technology behind virtualization is known as a virtual machine monitor (VMM) or virtual manager, which separates compute environments from the actual physical infrastructure.

Virtualization makes servers, workstations, storage and other systems independent of the physical hardware layer. This is done by installing a Hypervisor on top of the hardware layer, where the systems are then installed.

There are three areas of IT where virtualization is making head roads, network virtualization, storage virtualization and server virtualization:

- Network virtualization is a method of combining the available resources in a network by splitting up the available bandwidth into channels, each of which is independent from the others, and each of which can be assigned (or reassigned) to a particular server or device in real time. The idea is that virtualization disguises the true complexity of the network by separating it into manageable parts, much like your partitioned hard drive makes it easier to manage your files.

- Storage virtualization is the pooling of physical storage from multiple network storage devices into what appears to be a single storage device that is managed from a central console. Storage virtualization is commonly used in storage area networks (SANs).

- Server virtualization is the masking of server resources (including the number and identity of individual physical servers, processors, and operating systems) from server users. The intention is to spare the user from having to understand and manage complicated details of server resources while increasing resource sharing and utilization and maintaining the capacity to expand later.

Virtualization can be viewed as part of an overall trend in enterprise IT that includes autonomic computing, a scenario in which the IT environment will be able to manage itself based on perceived activity, and utility computing, in which computer processing power is seen as a utility that clients can pay for only as needed. The usual goal of virtualization is to centralize administrative tasks while improving scalability and workloads.

Hardware / Software Required: Ubuntu operating system, open-source software KVM, Internet.

<u>Installation Steps :</u>

1.sudo grep-c"svm\|vmx"/proc/cpuinfo

2. sudo apt-get update

3.sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients virt-manager bridge-utils

4.sudo apt-get install qemu-kvm libvirt-daemon-system libvirt-clients virt-manager bridge-utils

5.sudo systemctl start libvirtd

6.sudo usermod -aG kvm $USER

7.sudo systemctl is-active libvirtd

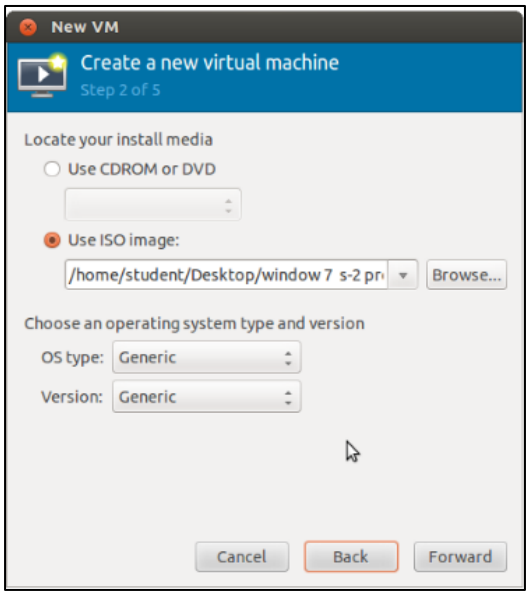
8.sudo usermod -aG libvirt $USER

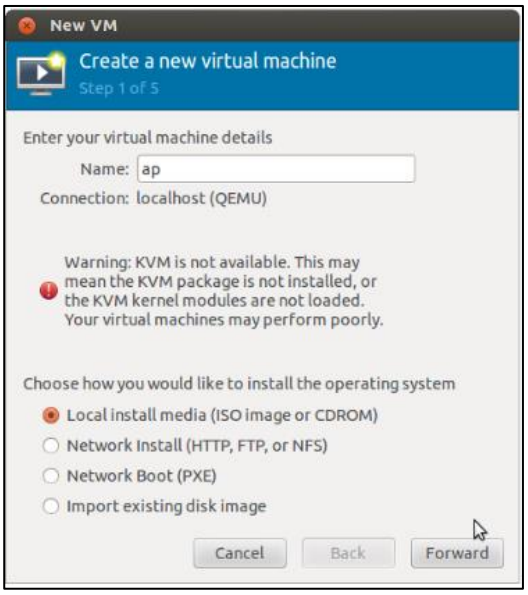
9. sudo usermod -aG kvm $USER

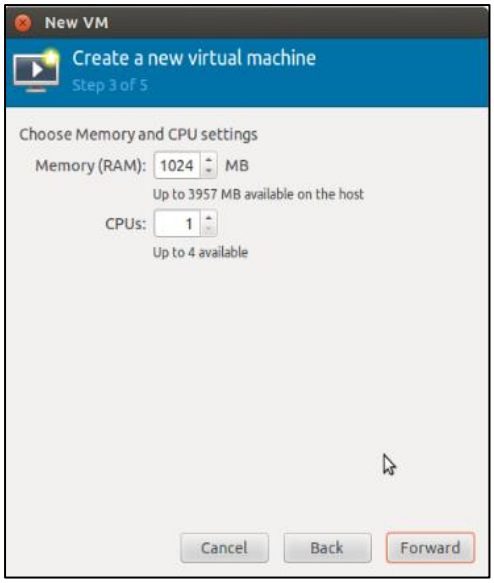
10.Open Virtual Machine Manager application and Create Virtual Machine

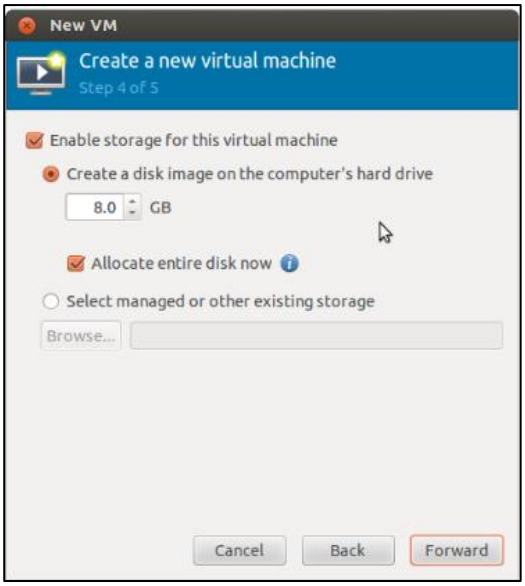
#virt-manager as shown below

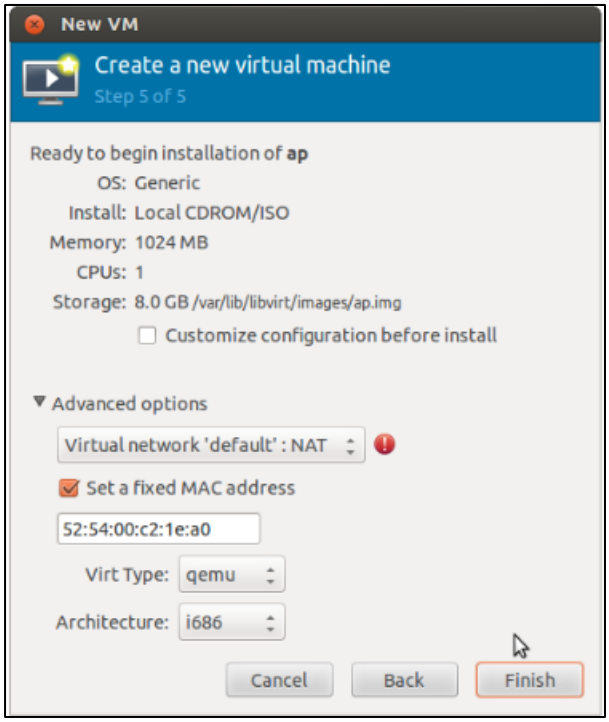
Create a new virtual machine as shown below



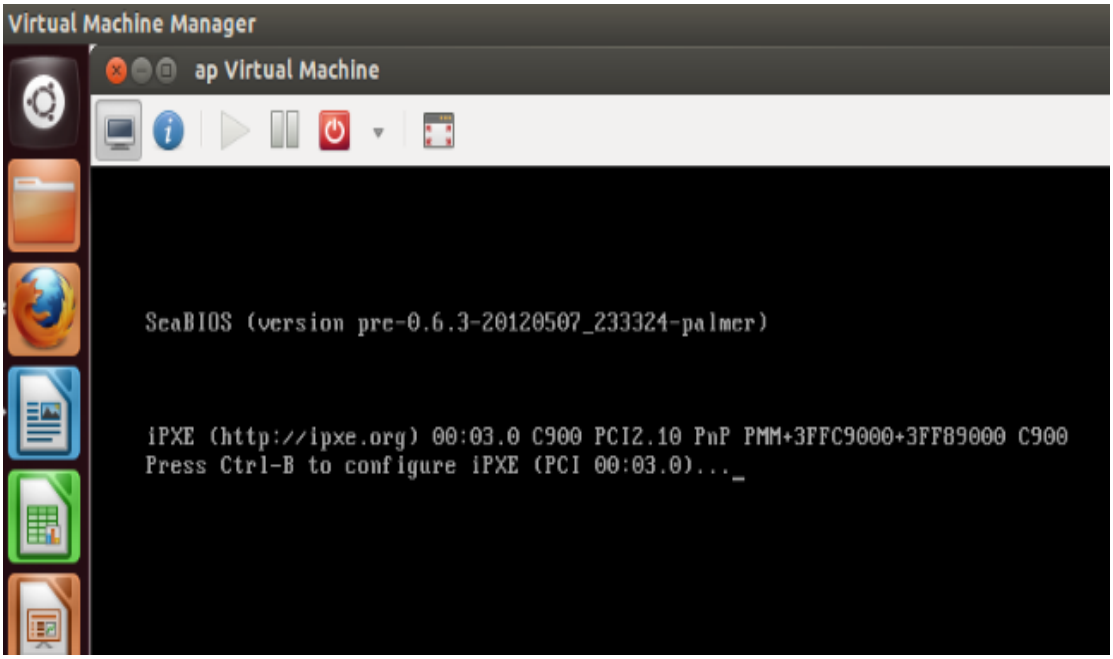



New VM
Create a new virtual machine
Step 3 of 5
Choose Memory and CPU settings
Memory (RAM): 1024 MB
Up to 3957 MB available on the host
CPUs: 1
Up to 4 available
Cancel
Back
Forward

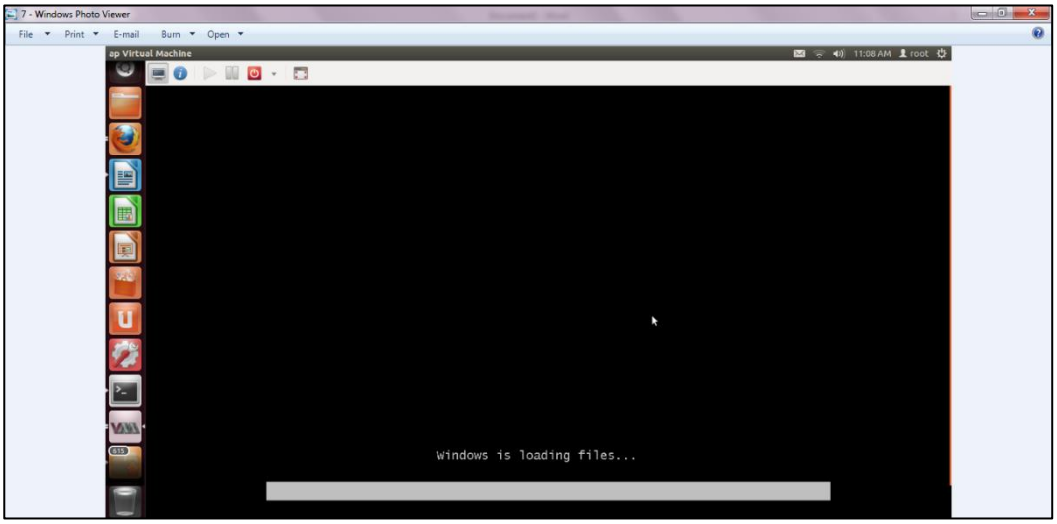

New VM
Create a new virtual machine
Step 4 of 5
Enable storage for this virtual machine
Create a disk image on the computer's hard drive
8.0 GB
Allocate entire disk now
Select managed or other existing storage
Browse...
Cancel
Back
Forward


New VM
Create a new virtual machine
Step 5 of 5
Ready to begin installation of ap
OS: Generic
Install: Local CDROM/ISO
Memory: 1024 MB
CPUs: 1
Storage: 8.0 GB /var/lib/libvirt/images/ap.img
Customize configuration before install
Advanced options
Virtual network 'default' : NAT
Set a fixed MAC address
52:54:00:c2:1e:a0
Virt Type: qemu
Architecture: i686
Cancel
Back
Finish

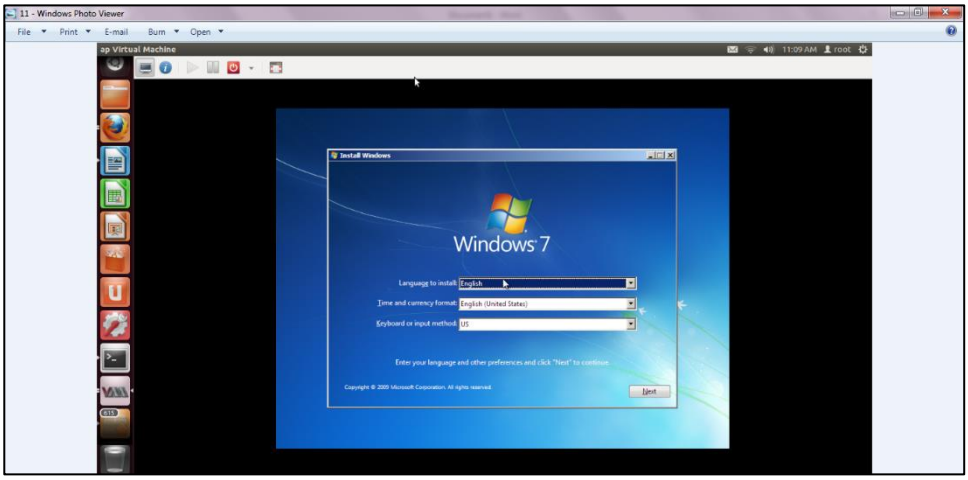
Install windows operating system on virtual machine



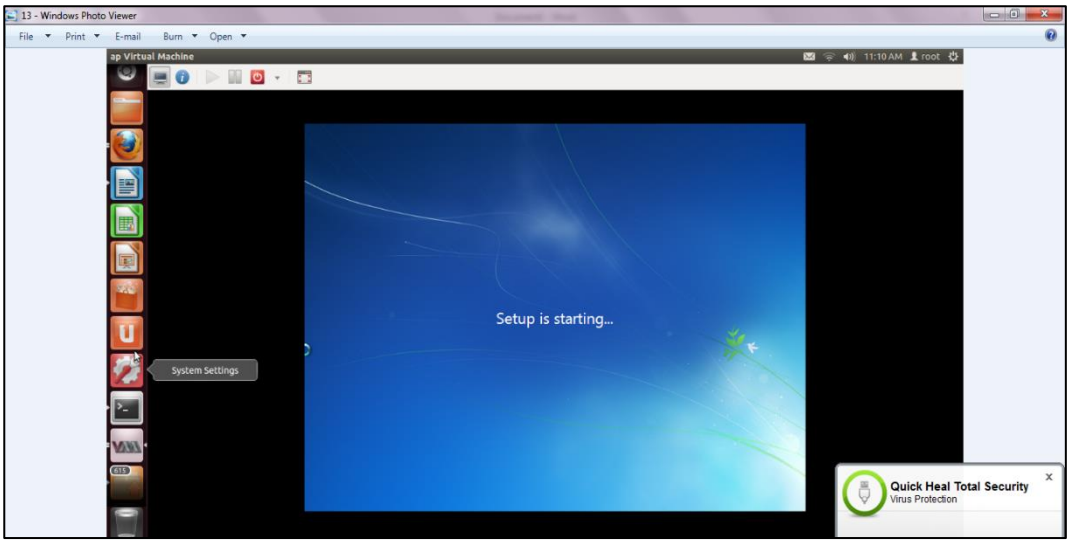
Installation of windows on virtual machine



Installation of windows 7 on virtual machine



Initialization of windows on virtual machine



PRACTICAL - 6

Aim: Develop an application to download image/video from server or upload image/video to server using MTOM techniques.

Developing a Python application that utilizes MTOM (Message Transmission Optimization Mechanism) for transferring binary data such as images or videos over HTTP, and we can test it using Postman.

MTOM (Message Transmission Optimization Mechanism) is a technique used in web services to efficiently transmit binary data, such as images or videos, over SOAP (Simple Object Access Protocol) messages. MTOM optimizes the transmission of binary data by sending it separately from the XML message envelope, which can significantly reduce overhead and improve performance.

Firstly, you need to have Flask installed in your Python environment. Flask is a micro web framework for Python.

Install Flask using pip:

pip install Flask

Now, let's create a basic Flask application that allows uploading and downloading images using MTOM.

app.py

code:

```
from flask import Flask, request, send_file

import os

app = Flask(__name__)

UPLOAD_FOLDER = 'uploads'

DOWNLOAD_FOLDER = 'downloads'

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

app.config['DOWNLOAD_FOLDER'] = DOWNLOAD_FOLDER

@app.route('/upload', methods=['POST'])

def upload_file():

    if 'file' not in request.files:

        return 'No file part'


    file = request.files['file']

    if file.filename == '':

        return 'No selected file'
```

```
    if file:
        filename = file.filename
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        return f'File {filename} uploaded successfully'

@app.route('/download/<filename>', methods=['GET'])
def download_file(filename):
    return send_file(os.path.join(app.config['UPLOAD_FOLDER'], filename),
as_attachment=True)

if __name__ == '__main__':
    app.run(debug=True)
```

(Save this code in a file named **app.py**)

This code creates a simple Flask application with two endpoints:

*Ensure that the paths for uploading and downloading files are correctly configured and that the specified directories (**uploads** and **downloads**) exist in the same directory as your Flask application script.*

/upload: This endpoint accepts **POST** requests with a file field named "file". It saves the uploaded file in the 'uploads' folder.

/download/<filename>: This endpoint accepts GET requests to download a file. It sends the requested file from the 'uploads' folder as an attachment.

To test this application, run it using the command:

python app.py

We can use Postman to test the upload and download functionality.

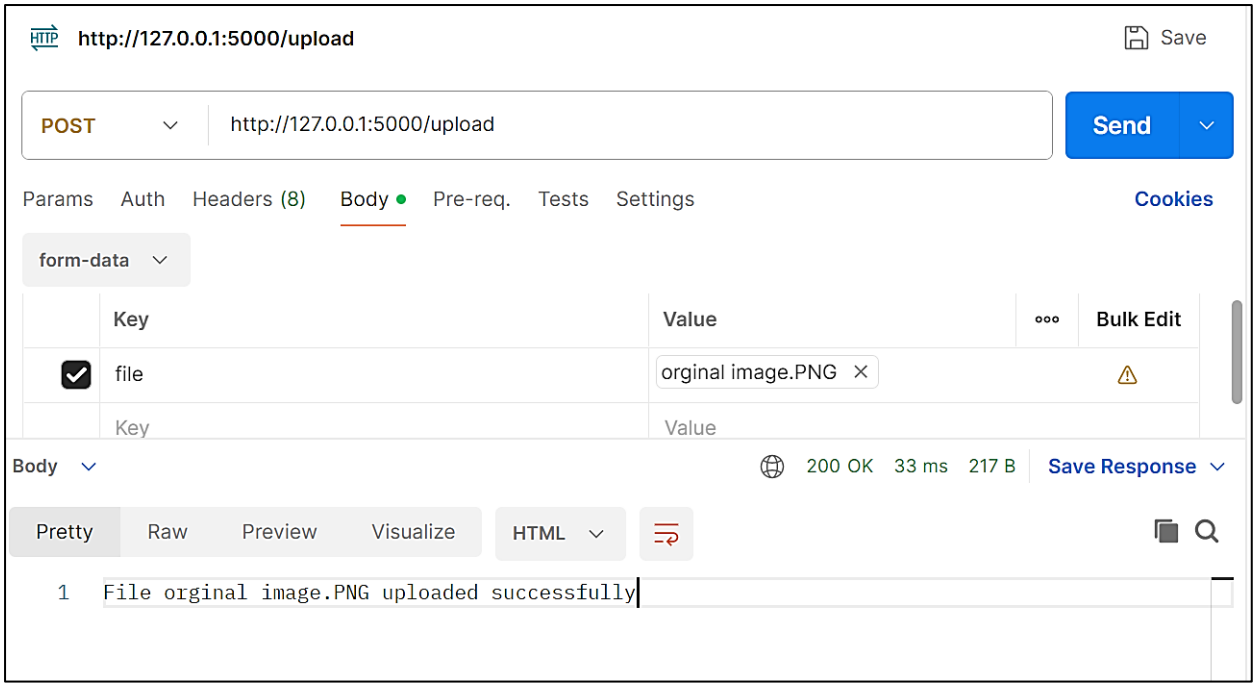
<u>Output:</u>

First we have to run flask for this

Go to New Terminal ->Type Command (**flask run**)

```
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deploymen
t. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

<u>Upload File</u>: Send a POST request to **http://localhost:5000/upload** with a form-data body containing a file field named "**file**".



<u>Download File:</u> Send a GET request to **http://localhost:5000/download/<filename>,** where <filename> is the name of the file you uploaded.The flile will be downloaded.

PRACTICAL - 7

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

Installation Steps:

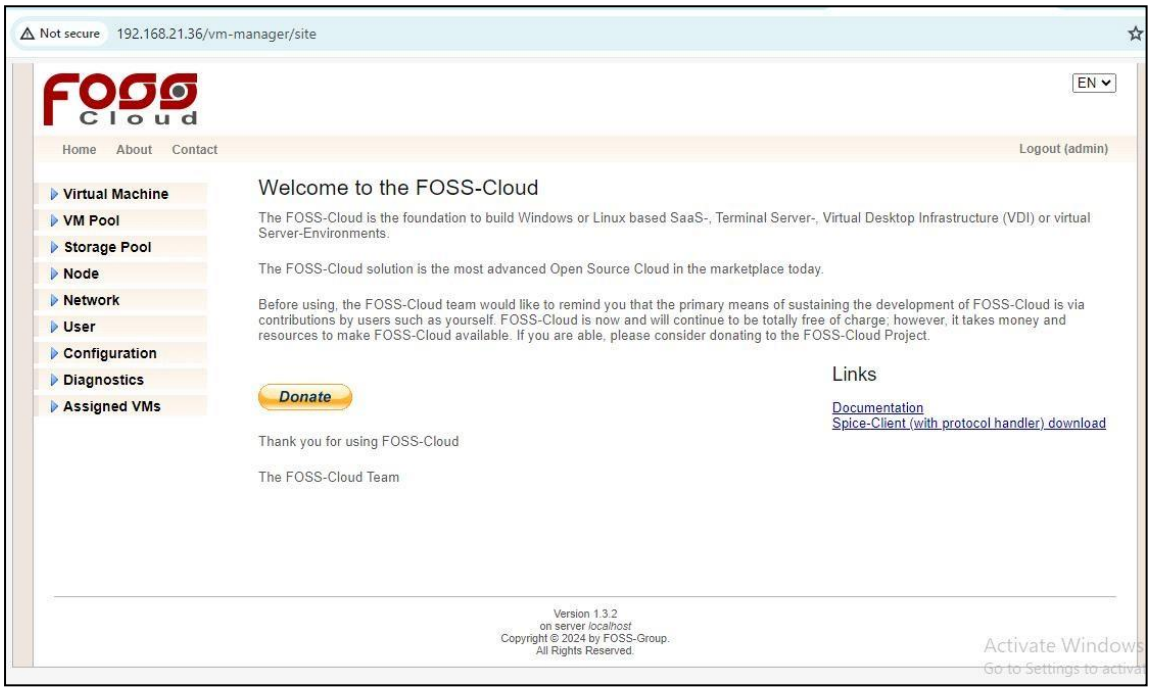
1. Choose your keymap.
2. Confirm that you want to start: yes
3. Choose Demo-System: 1
4. Choose a Block-Device: sda
5. Confirm that you want to continue: yes
6. Confirm that you want to continue: yes
7. Choose the network interface: eth0
8. Choose if you want to use automatic network configuration: Yes
9. Reboot your system: yes
10. Login as root and run "fc-node-configuration -n demo-system --password admin"

(Above mentioned steps aren't mandatory you can start with the installation process of virt-viewer as shown below.)

1. **Install virt-viewer.**
2. **Install spice-client.0.6.3.**

3. Open Browser.

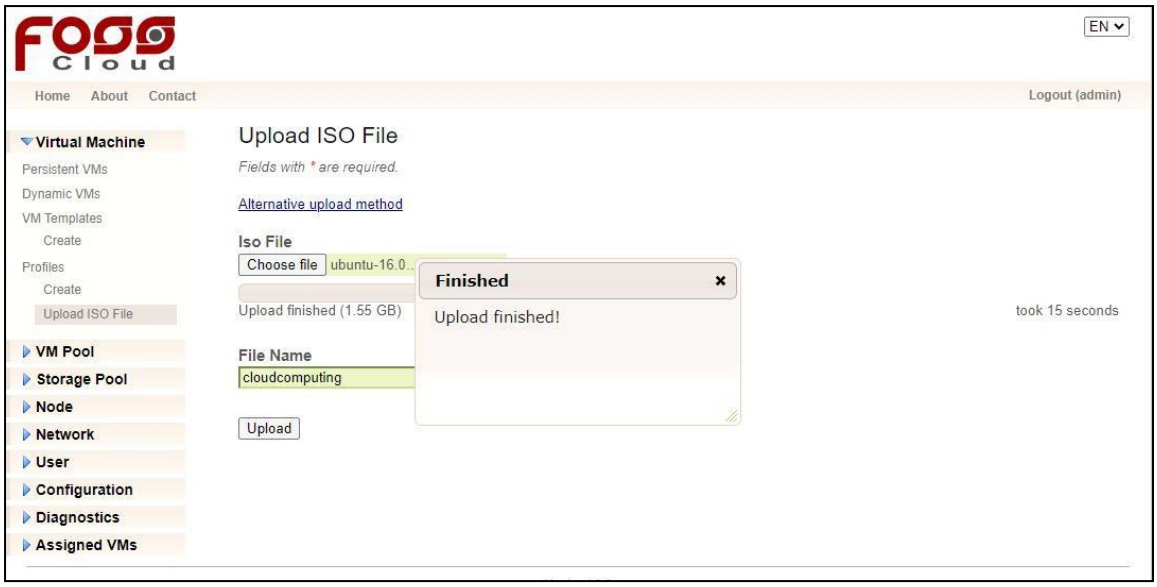
Once you open browser then on the search type**, 192.168.21.36**



Uploading ISO files to Foss cloud

Go to **virtual machine -> Profile->Upload ISO file->file name**

Now choose the **ubuntu-16.04.5-desktop-i386 iso-file** from the FOSS cloud folder. Then, give your desired file name to it and click on the upload button

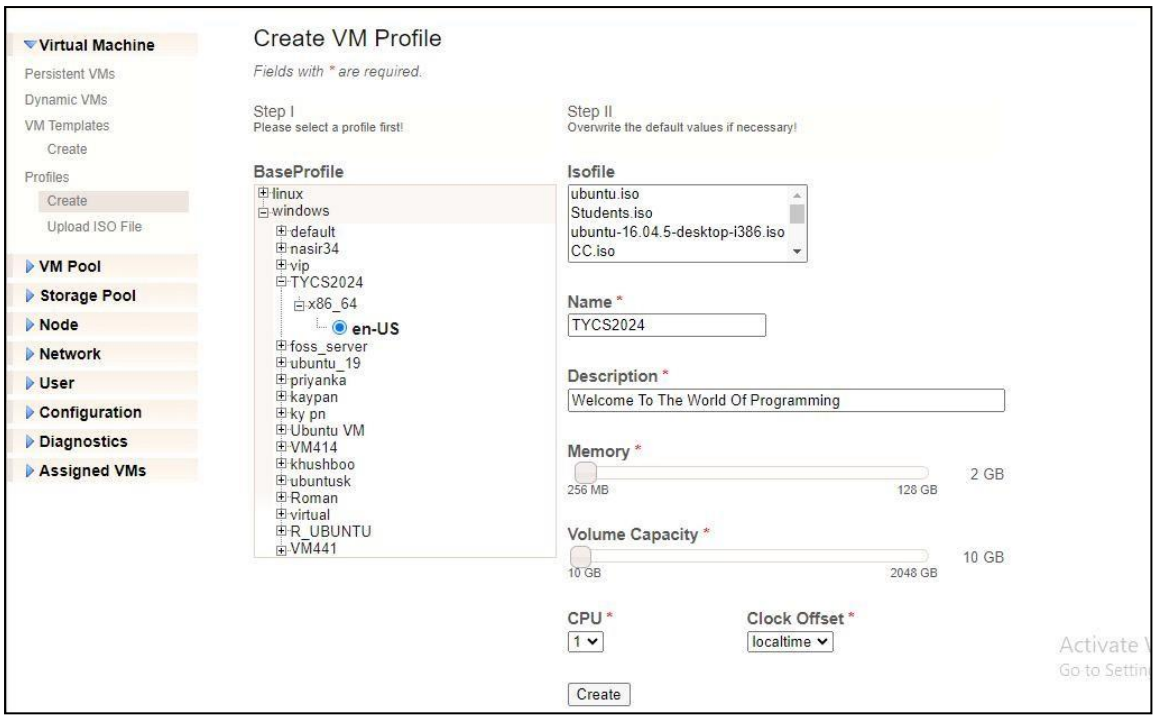


Once done with the upload process go to the create section in profiles.

Creating a Profile

The profile creates the relationship between the ISO file and the FOSS-Cloud.

1. Open Virtual Machines
2. Choose VM-Profiles
3. Choose the right Base Profile
4. Choose the right architecture (Windows only x86_64)
5. Chose the language (it is an information - not keyboard relevant)
6. Choose the ISO file (which you gave in file name tab)
7. Fill out name and description
8. Choose the amount of memory and volume capacity
9. Choose amount of CPU
10. Choose clock offset (normally Windows is "localtime" and Linux is "utc")



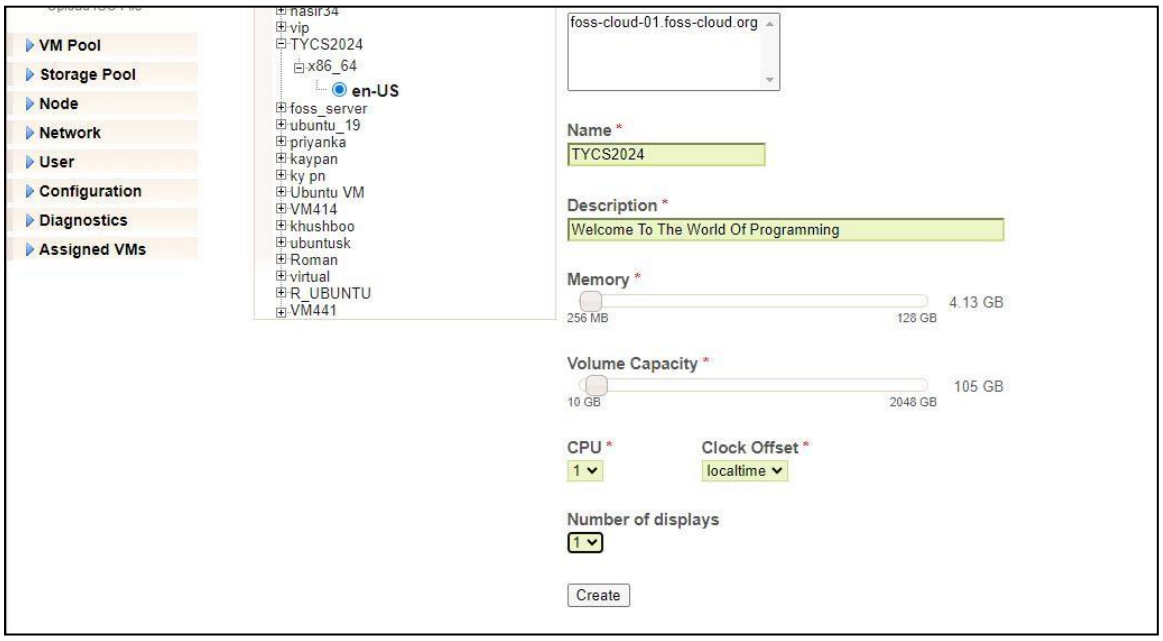
Creating Template

1. Choose the profile you have prepared before.

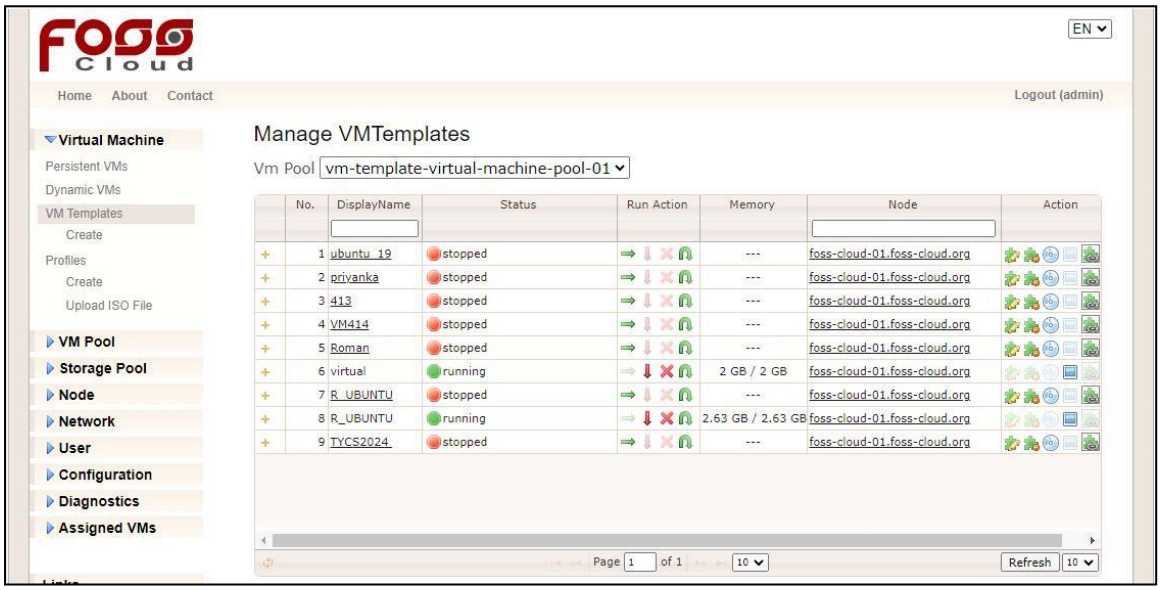
2. Add the VM-pool and one or more nodes, where you will run this VM (when the chosen VM-pool has only one node assigned, you don't have a choice)

3. You can change all the other information you have entered before

Click on "create" and the template is ready for installing the guest operating system.

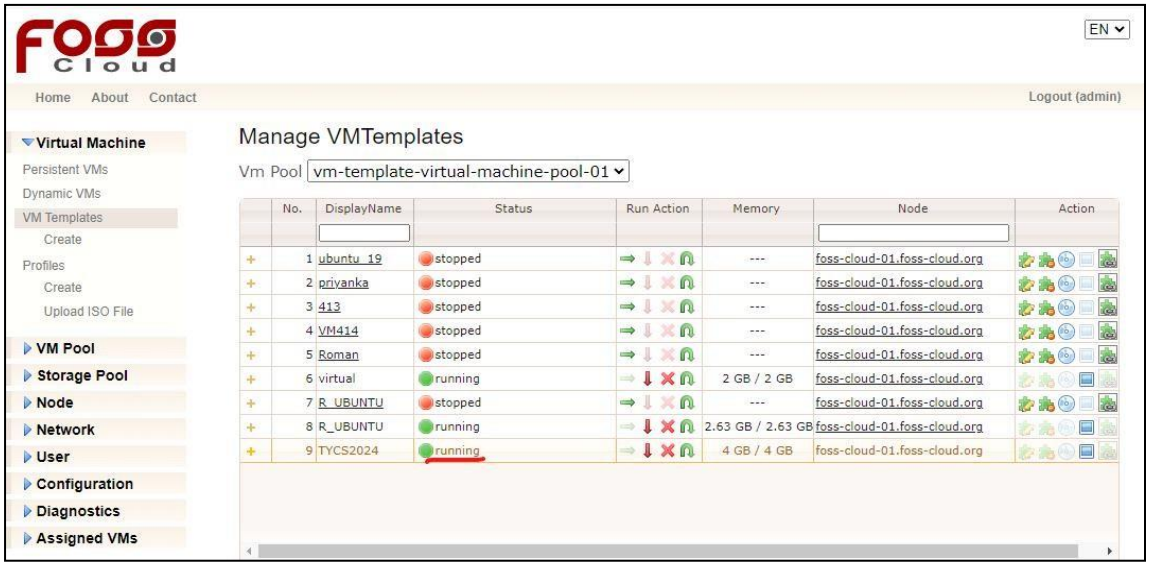


Once you create your virtual machine template you will be able to see your **VM template.**

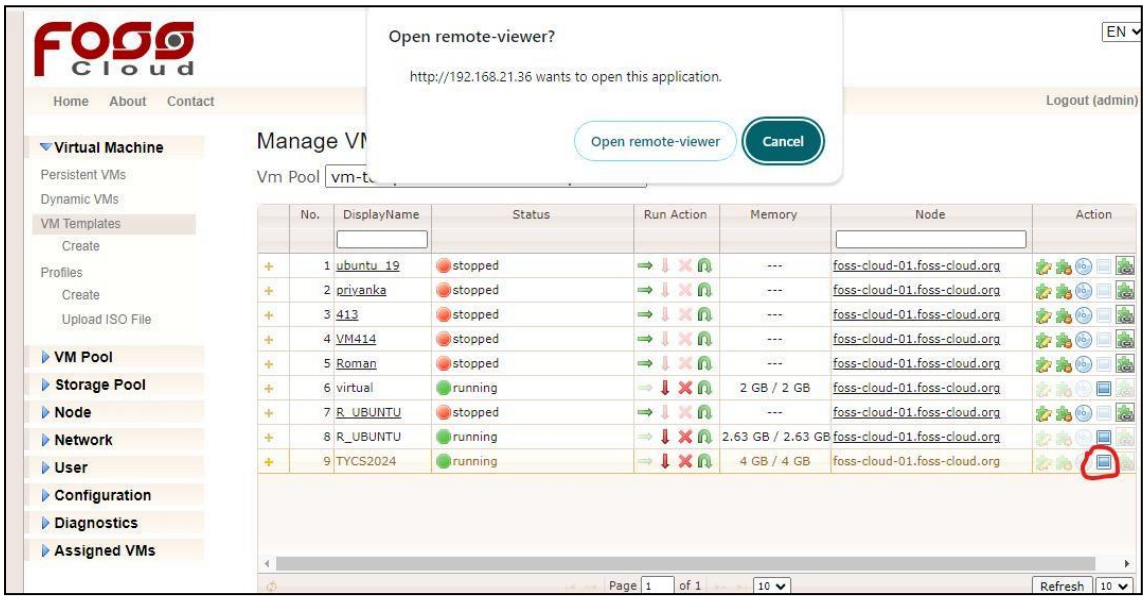


As you can see the last row of Display column our template is created.

To update your template status, click on the green arrow

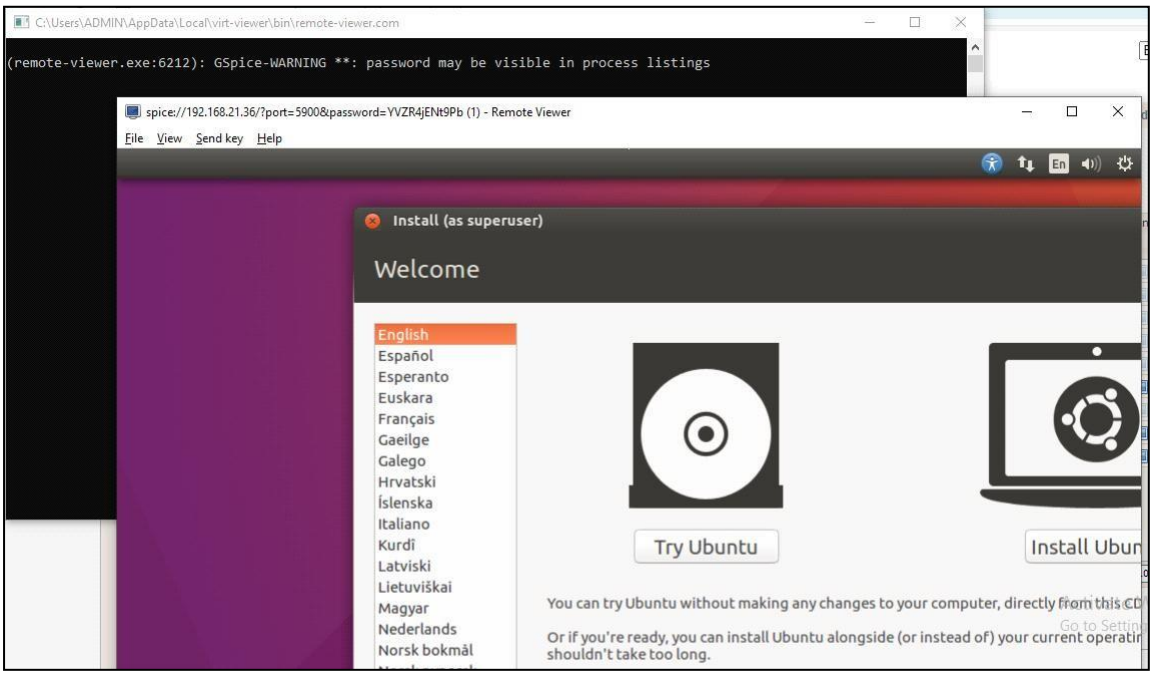


Now click on Use Template in Action column of your corresponding template.



Just after clicking on, it a pop up would appear to Open remote-viewer?

Click on Open-remote- viewer button, you will see the following screen.



PRACTICAL - 8

Aim: Implement FOSS-Cloud Functionality-VSI Platform as a Service (PaaS)

Software development Kits can be made available in the Virtual Machines that can be implemented as Platform as a Service.

Installation of Netbeans, Eclipse, Visual Studio and DBMS can be done in the appropriate Virtual Machines.