

## Practical 1

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

### Software Tools Required:

- **Code Editor:** Visual Studio Code (VS Code)
- **API Testing Software Application:** Postman
- **Framework:** Express.js (web framework)
- **Runtime Environment:** Node.js
- **Programming Languages:** JavaScript (for Node.js and Express.js), Java, C# (for .NET)

### Downloads Required:

- Node.js: [Node.js — Download Node.js®](#)
- JDK: [Java Downloads | Oracle India \(x64 MSI Installer\)](#)
- Dotnet SDK: [Download .NET 9.0 SDK \(v9.0.101\) - Windows x64 Installer](#)
- Postman: [Download Postman | Get Started for Free](#)  
(Create Account/ Log in with Google)

After Downloading Node.js, JDK and Dotnet SDK, Set the Path in Environment Variable in User Variables > Path > Edit > New and Paste the Path for Node.js, JDK and Dotnet SDK

### Demonstration:

---

C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin

---

C:\Program Files\Java\jdk-23\bin

---

C:\Program Files\nodejs\  

---

Click OK to remaining opened Edit environment variable, Environment Variables and System Properties windows.

Now check the versions that shows path is set for Nodejs, Java JDK and Dotnet SDK

```
C:\Users\Kishore>node -v
v22.11.0

C:\Users\Kishore>java --version
java 23.0.1 2024-10-15
Java(TM) SE Runtime Environment (build 23.0.1+11-39)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.1+11-39, mixed mode, sharing)

C:\Users\Kishore>dotnet --version
9.0.101
```

Here all versions are shown so path is set by checking versions for Nodejs, Java JDK and Dotnet SDK.

## 1. Design the Currency Conversion Service:

Create a simple API that takes an amount in Indian Rupees and returns the equivalent amount in US Dollars. You can use a simple formula for conversion, or you can fetch real-time exchange rates from a reliable source.

Example API endpoint:

POST/convert

Request:

```
"amount in rs": 1000
```

Response:

```
{  
  
"amount in usd": 14.5  
  
}
```

## 2. Implement the Service:

You can implement the service using any programming language and framework. For simplicity, let's use Node.js with Express in this example,

### i. Set up a Node.js Project

Create a Project Directory

Example: Directory Path: Documents in C Drive

Create a Folder Name: Cloud\_Computing\_Practical1

C: \Documents\Cloud\_Computing\_Practical1

Select the Directory Path, Cut (Backspace) and Type cmd and hit enter

We will get inside the path now type the following commands

- `mkdir currency-conversion-service` : `mkdir` - Make directory
- `cd currency-conversion-service` : `cd` – change directory
- `npm init -y` : `npm` – node package manager initializes yes
- `npm install express body-parser` : To create simple server & handle incoming requests

## Demonstration:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kishore\Documents\Cloud_Computing_Practical1>mkdir currency-conversion-service

C:\Users\Kishore\Documents\Cloud_Computing_Practical1>cd currency-conversion-service

C:\Users\Kishore\Documents\Cloud_Computing_Practical1\currency-conversion-service>npm init -y
Wrote to C:\Users\Kishore\Documents\Cloud_Computing_Practical1\currency-conversion-service\package.json:

{
  "name": "currency-conversion-service",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

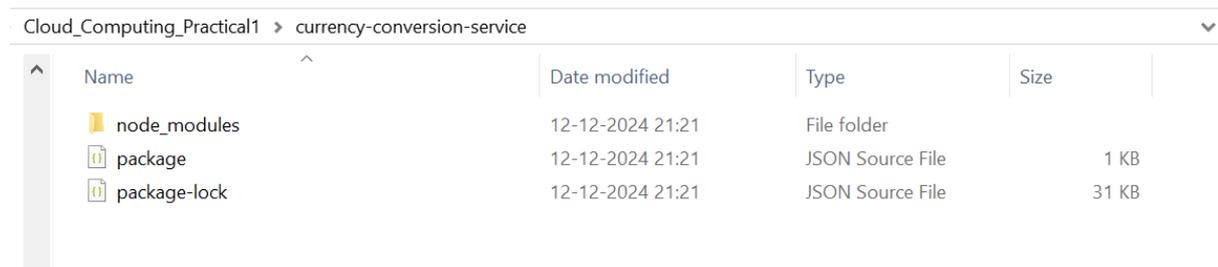
C:\Users\Kishore\Documents\Cloud_Computing_Practical1\currency-conversion-service>npm install express body-parser
added 72 packages, and audited 73 packages in 3s

17 packages are looking for funding
  run `npm fund` for details

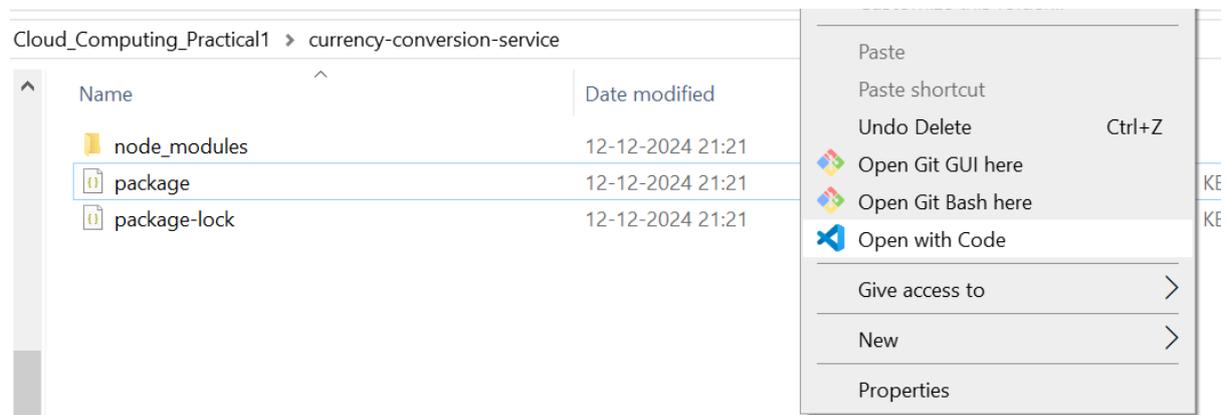
found 0 vulnerabilities

C:\Users\Kishore\Documents\Cloud_Computing_Practical1\currency-conversion-service>
```

Inside currency-conversion-service > this is the folder architecture after initializing npm and express body-parser.



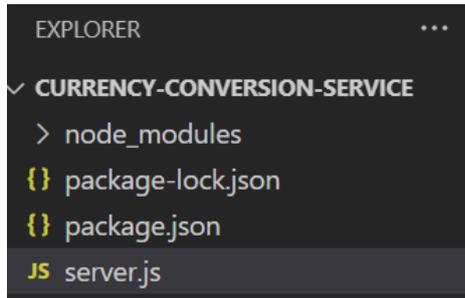
## Now Open with VS Code



## JavaScript (for Node.js and Express.js)

### File Architecture:

Create a file to implement the service with JavaScript Programming language



**Filename:** server.js

### Code:

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const port = 3000;
app.use(bodyParser.json());
app.post('/convert', (req, res) => {
  const amountInRs = req.body.amount_in_rs;
  // Perform the conversion (use a real exchange rate or a fixed rate for simplicity)
  const conversionRate = 0.0145;
  const amountInUsd = amountInRs * conversionRate;
  res.json({ amount_in_usd: amountInUsd });
});
app.listen(port, () => {
  console.log(`Currency conversion service running on http://localhost:${port}`);
});
```

### Output:

Click View > Terminal:

```
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\currency-conversion-service> node server.js
```

Currency conversion service running on <http://localhost:3000>

**The Output is successful indicating the service is running on <http://localhost:3000>**

## Demonstration:

```
JS server.js X
JS server.js > app.listen() callback
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const app = express();
4  const port = 3000;
5  app.use(bodyParser.json());
6  app.post('/convert', (req, res) => {
7      const amountInRs = req.body.amount_in_rs;
8      // Perform the conversion (use a real exchange rate or a fixed rate for simplicity)
9      const conversionRate = 0.0145;
10     const amountInUsd = amountInRs * conversionRate;
11     res.json({ amount_in_usd: amountInUsd });
12 });
13 app.listen(port, () => {
14     console.log(`Currency conversion service running on http://localhost:${port}`);
15 });
16

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\currency-conversion-service> node server.js
Currency conversion service running on http://localhost:3000
█
```

# Postman

## API Testing Software Application to Test the API

### Open Postman

### Send a POST Request:

**URL:** <http://localhost:3000/convert>

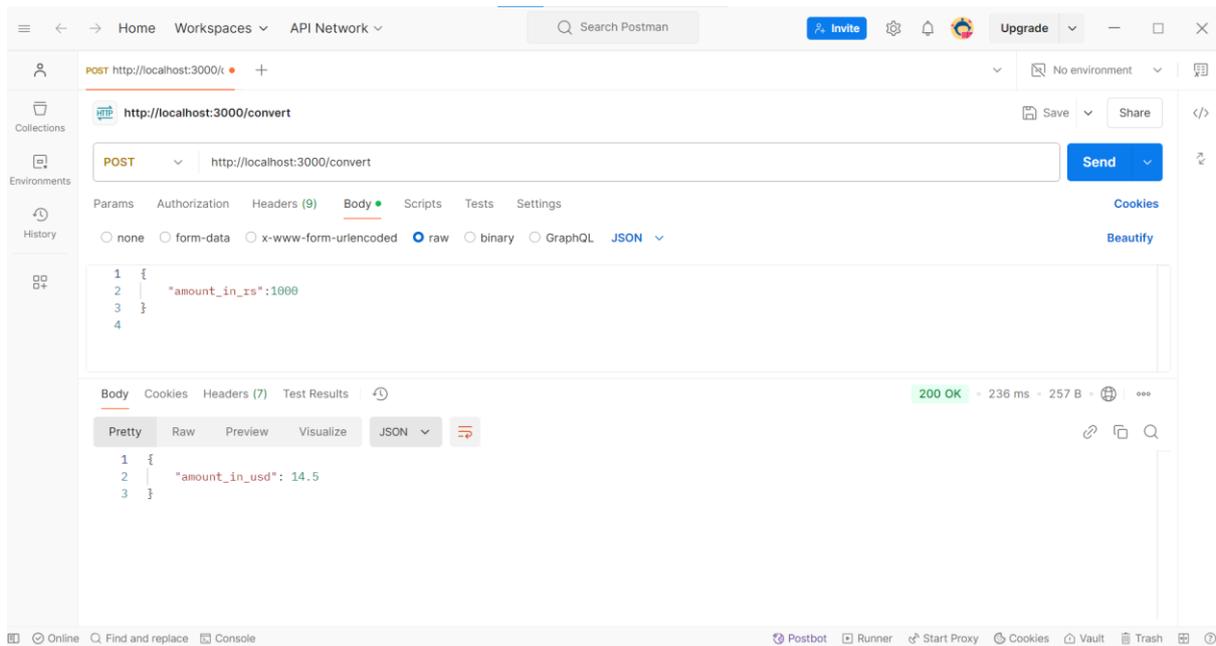
**Body:** Set the body to raw JSON and include:

```
{
  "amount_in_rs": 1000
}
```

### Response:

```
{
  "amount_in_usd": 14.5
}
```

### Demonstration:



Successfully output retrieved in Postman to test API.

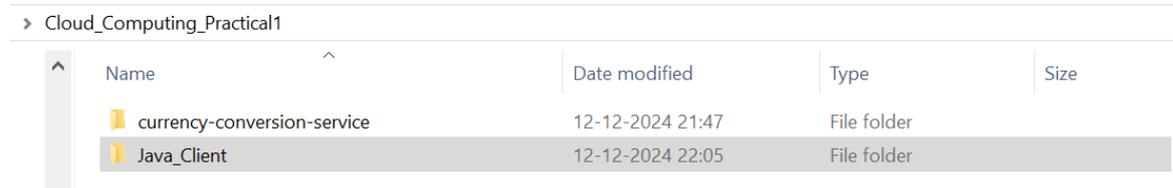
Now, to call from 2 different platforms Java and .NET Programming Language.

## Java Programming Language

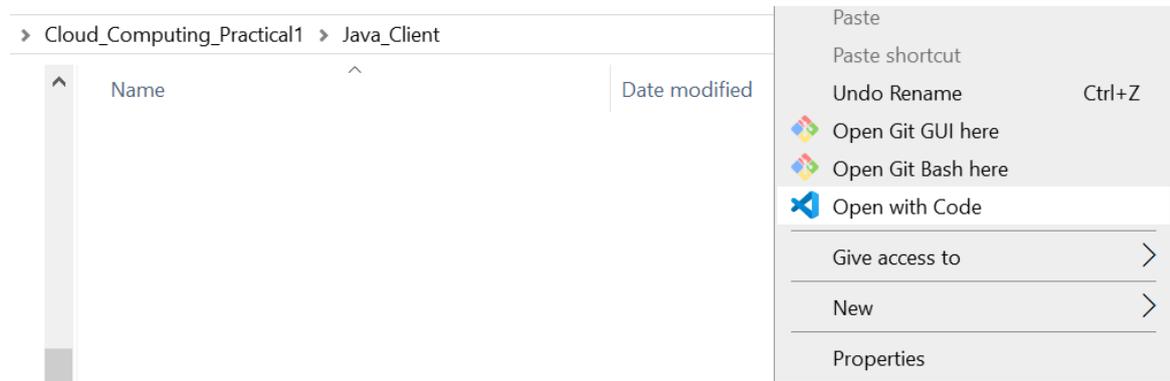
1. Java Client: Implement a Java Client to call the currency conversion service.  
You can use libraries like HttpClient for making HTTP requests.

**Create folder name:** Java\_Client in Cloud\_Computing\_Practical1

### Demonstration:



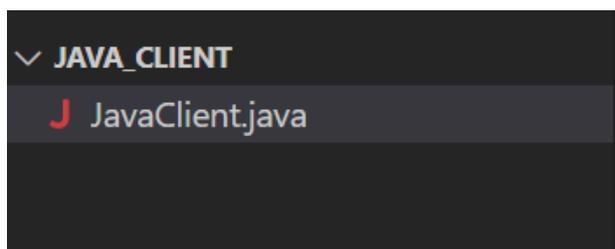
Open Java\_Client and Open with VS Code



**Extension:** Install Java in VS Code



**File Architecture:**



**Filename:** JavaClient.java

**Code:**

```
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.net.http.HttpRequest.BodyPublishers;
import java.net.http.HttpResponse.BodyHandlers;

public class JavaClient {
    public static void main(String[] args) {
        HttpClient client = HttpClient.newHttpClient();
        String uri = "http://localhost:3000/convert";
        String requestBody = "{ \"amount_in_rs\": 1000 }";

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(uri))
            .header("Content-Type", "application/json")
            .POST(BodyPublishers.ofString(requestBody))
            .build();

        client.sendAsync(request, BodyHandlers.ofString())
            .thenApply(HttpResponse::body)
            .thenAccept(System.out::println)
            .join();
    }
}
```

**Output:**

Click View > Terminal:

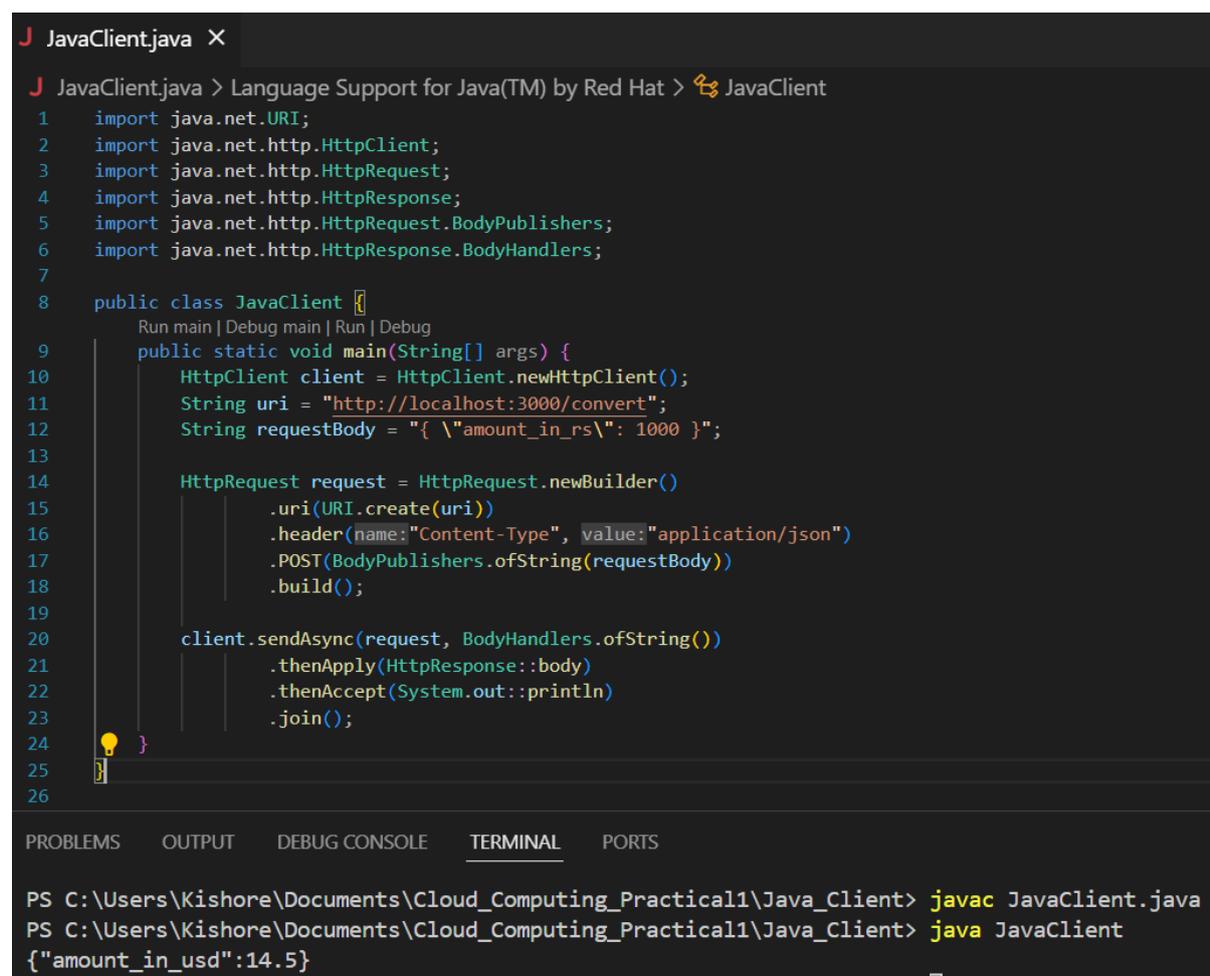
```
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\Java_Client> javac JavaClient.java
```

```
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\Java_Client> java JavaClient
```

```
{"amount_in_usd":14.5 }
```

**The Output is Successful that we called the server.js with making HTTP request**

## Demonstration:



```
J JavaClient.java X
J JavaClient.java > Language Support for Java(TM) by Red Hat > JavaClient
1  import java.net.URI;
2  import java.net.http.HttpClient;
3  import java.net.http.HttpRequest;
4  import java.net.http.HttpResponse;
5  import java.net.http.HttpRequest.BodyPublishers;
6  import java.net.http.HttpResponse.BodyHandlers;
7
8  public class JavaClient {
9      Run main | Debug main | Run | Debug
10     public static void main(String[] args) {
11         HttpClient client = HttpClient.newHttpClient();
12         String uri = "http://localhost:3000/convert";
13         String requestBody = "{ \"amount_in_rs\": 1000 }";
14
15         HttpRequest request = HttpRequest.newBuilder()
16             .uri(URI.create(uri))
17             .header("Content-Type", "application/json")
18             .POST(BodyPublishers.ofString(requestBody))
19             .build();
20
21         client.sendAsync(request, BodyHandlers.ofString())
22             .thenApply(HttpResponse::body)
23             .thenAccept(System.out::println)
24             .join();
25     }
26 }

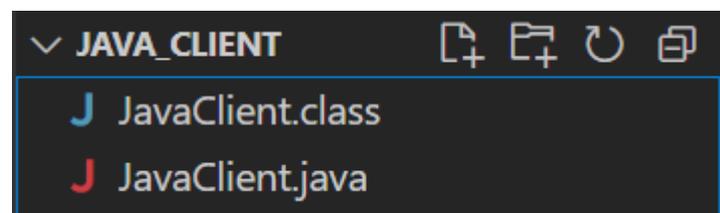
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\Java_Client> javac JavaClient.java
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\Java_Client> java JavaClient
{"amount_in_usd":14.5}
```

The java command is used to run the compiled Java bytecode. It starts the JVM, loads the specified .class file, and then executes the main method of that class.

1. Ensure you have a .class file: This file is generated by compiling a .java file using the javac command.
2. Run the Java program: Use the java command followed by the name of the class (without the .class extension) to execute the program.

Here JavaClient.class file will be created after running the program

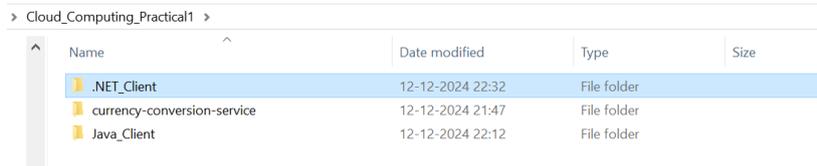


## .NET Programming Language

2. .NET Client: Implement a .NET Client to call the currency conversion service.  
You can use HttpClient in C#.

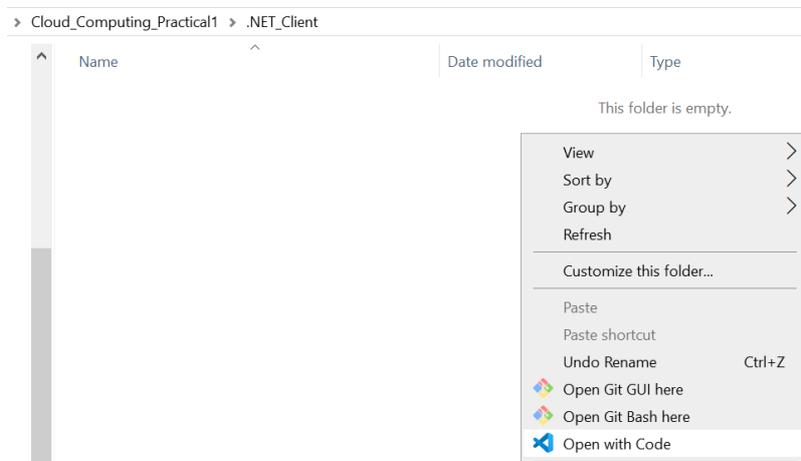
**Create folder name:** .NET\_Client in Cloud\_Computing\_Practical1

### Demonstration:



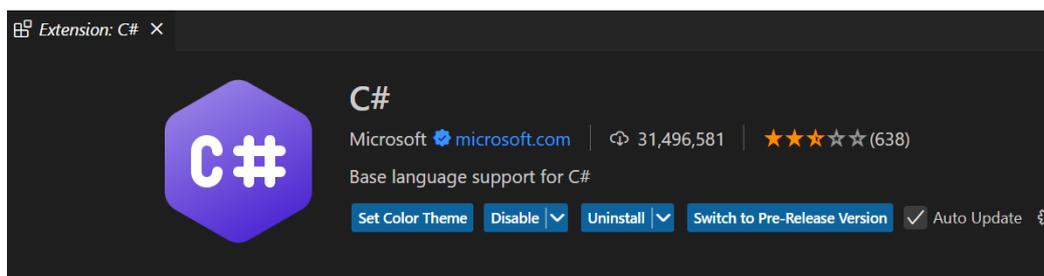
Name	Date modified	Type	Size
.NET_Client	12-12-2024 22:32	File folder	
currency-conversion-service	12-12-2024 21:47	File folder	
Java_Client	12-12-2024 22:12	File folder	

Open .NET\_Client and Open with VS Code



**Extension: 4 Extensions needed to work in C#**

### C#



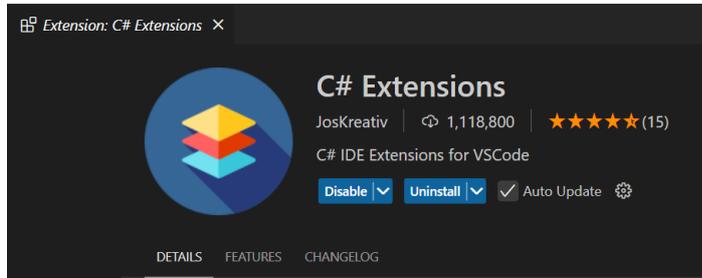
### C# Snippet



## C# Dev Kit



## C# Extensions



## File Architecture:

Click on View > Terminal and Type the following command

- dotnet new console
- dotnet restore

## Demonstration:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

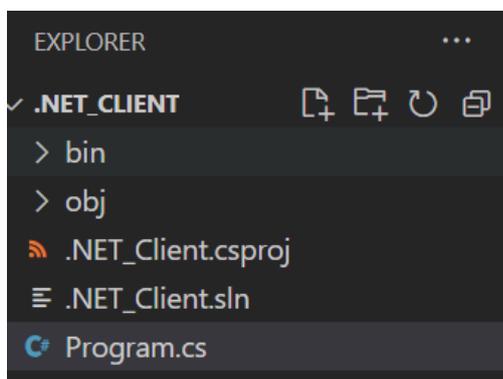
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\.NET_Client> dotnet new console
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring C:\Users\Kishore\Documents\Cloud_Computing_Practical1\.NET_Client\.NET_Client.csproj:
Restore succeeded.

PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\.NET_Client> dotnet restore
Restore complete (0.7s)

Build succeeded in 1.6s
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\.NET_Client> █
```

After entering dotnet new console command the file architecture will generate the .NET Project with Program.cs file shown below.



**Filename:** Program.cs

**Code:**

```
using System;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;

class Program
{
    static async Task Main()
    {
        try
        {
            var client = new HttpClient();
            var apiUrl = "http://localhost:3000/convert";

            // Create the request body with correct JSON format
            var requestBody = new
            {
                amount_in_rs = 1000
            };
            var jsonRequestBody = System.Text.Json.JsonSerializer.Serialize(requestBody);

            // Send an HTTP POST request to the API with the request body
            var response = await client.PostAsync(apiUrl, new StringContent(jsonRequestBody,
            Encoding.UTF8, "application/json"));

            // Check the response status code
            if (response.IsSuccessStatusCode)
            {
                Console.WriteLine("Response Code: " + response.StatusCode);
                Console.WriteLine("Response Body: " + await response.Content.ReadAsStringAsync());
            }
            else
            {
                Console.WriteLine("Error: " + response.StatusCode);
                Console.WriteLine("Details: " + await response.Content.ReadAsStringAsync());
            }
        }
        catch (HttpRequestException e)
        {
            Console.WriteLine("An error occurred: " + e.Message);
        }
    }
}
```

## Output:

Click View > Terminal:

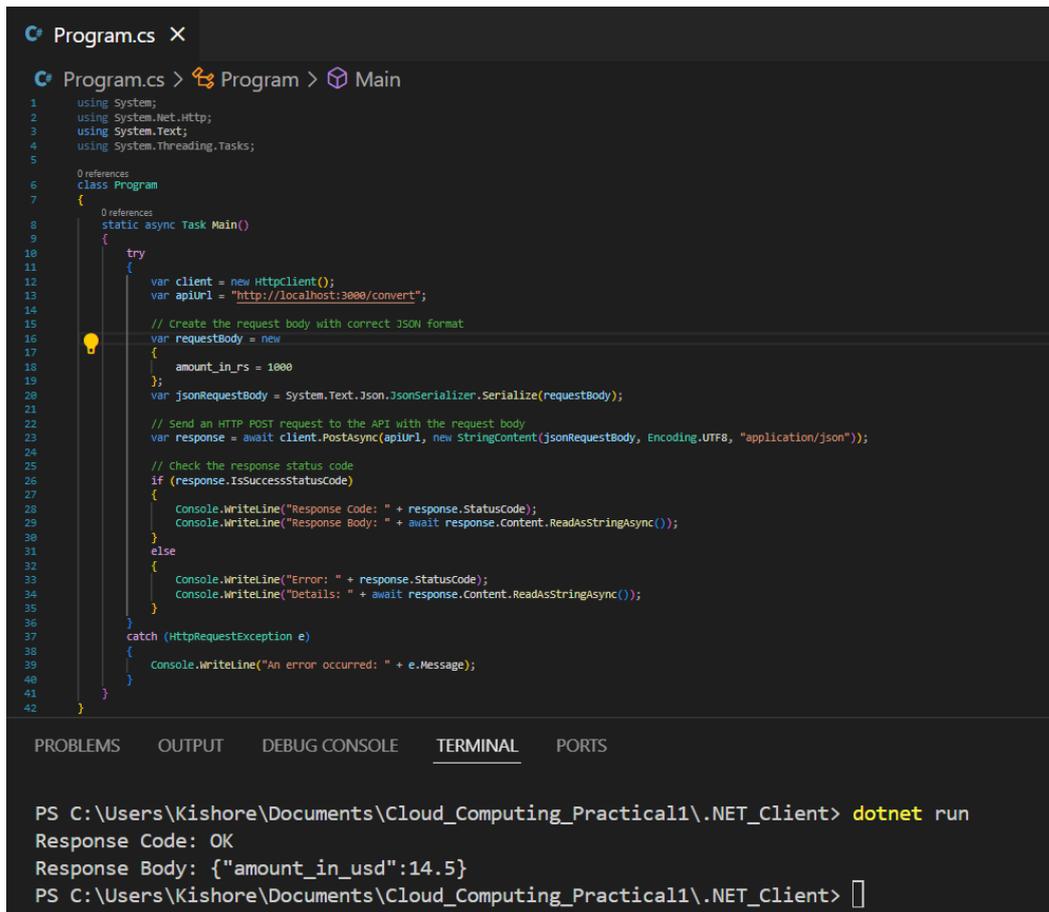
```
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\NET_Client> dotnet run
```

```
Response Code: OK
```

```
Response Body: {"amount_in_usd":14.5}
```

The Output is Successful that we called the server.js with making HTTP request

## Demonstration:



The screenshot shows a Visual Studio Code editor window with a C# file named Program.cs. The code defines a class Program with a static async Task Main() method. Inside Main(), an HttpClient is created, and a POST request is sent to http://localhost:3000/convert with a JSON body {"amount\_in\_rs": 1000}. The response is checked for success, and the status code and body are printed to the console. The terminal output shows the command dotnet run and the resulting response: Response Code: OK and Response Body: {"amount\_in\_usd":14.5}.

```
Program.cs X
Program.cs > Program > Main
1  using System;
2  using System.Net.Http;
3  using System.Text;
4  using System.Threading.Tasks;
5
6  0 references
7  class Program
8  {
9      0 references
10     static async Task Main()
11     {
12         try
13         {
14             var client = new HttpClient();
15             var apiUrl = "http://localhost:3000/convert";
16
17             // Create the request body with correct JSON format
18             var requestBody = new
19             {
20                 amount_in_rs = 1000
21             };
22             var jsonRequestBody = System.Text.Json.JsonSerializer.Serialize(requestBody);
23
24             // Send an HTTP POST request to the API with the request body
25             var response = await client.PostAsync(apiUrl, new StringContent(jsonRequestBody, Encoding.UTF8, "application/json"));
26
27             // Check the response status code
28             if (response.IsSuccessStatusCode)
29             {
30                 Console.WriteLine("Response Code: " + response.StatusCode);
31                 Console.WriteLine("Response Body: " + await response.Content.ReadAsStringAsync());
32             }
33             else
34             {
35                 Console.WriteLine("Error: " + response.StatusCode);
36                 Console.WriteLine("Details: " + await response.Content.ReadAsStringAsync());
37             }
38         }
39         catch (HttpRequestException e)
40         {
41             Console.WriteLine("An error occurred: " + e.Message);
42         }
43     }
44 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\NET_Client> dotnet run
Response Code: OK
Response Body: {"amount_in_usd":14.5}
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\NET_Client> █
```

**Conclusion:** The Practical worked successfully after creating currency conversion in postman with javascript and calling with Java and .NET by HTTP Requests.

-----

## Practical 2

Create a Simple SOAP service.

### Software Tools Required:

- **Code Editor:** Eclipse IDE
- **Build Automation Tool:** Apache Maven
- **SOAP Testing Tool:** SOAPUI
- **Framework:** Apache CXF
- **Programming Language:** Java

### Downloads Required:

- Apache Maven: [Download Apache Maven – Maven](#)
- JDK: [Java Downloads | Oracle India](#) (x64 MSI Installer)
- Eclipse IDE: [Eclipse downloads - Select a mirror | The Eclipse Foundation](#)
- SOAP Testing Tool: [Download REST & SOAP Automated API Testing Tool | Open Source | SoapUI](#) (SoapUI Open Source)

After Downloading JDK and Maven, Set the Path in Environment Variable in User Variables > Path > Edit > New and Paste the Path for JDK and Maven and check the versions for both in command prompt.

### Demonstration:

Path Set

C:\Program Files\Java\jdk-23\bin

D:\apache-maven-3.9.9\bin

Version checked in command prompt

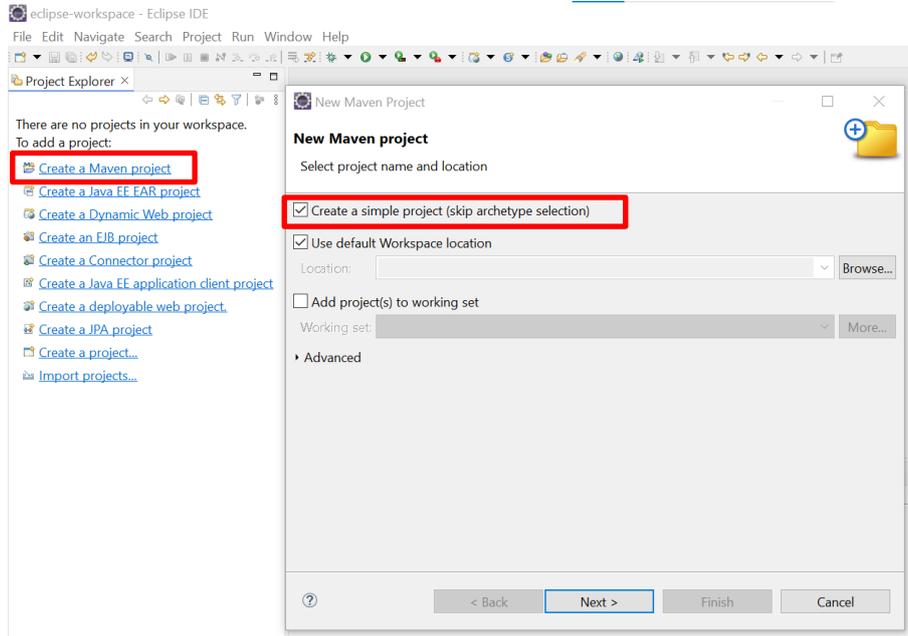
```
C:\Users\Kishore>mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfc97d260186937)
Maven home: D:\apache-maven-3.9.9
Java version: 23.0.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-23
Default locale: en_IN, platform encoding: UTF-8
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

C:\Users\Kishore>java --version
java 23.0.1 2024-10-15
Java(TM) SE Runtime Environment (build 23.0.1+11-39)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.1+11-39, mixed mode, sharing)
```

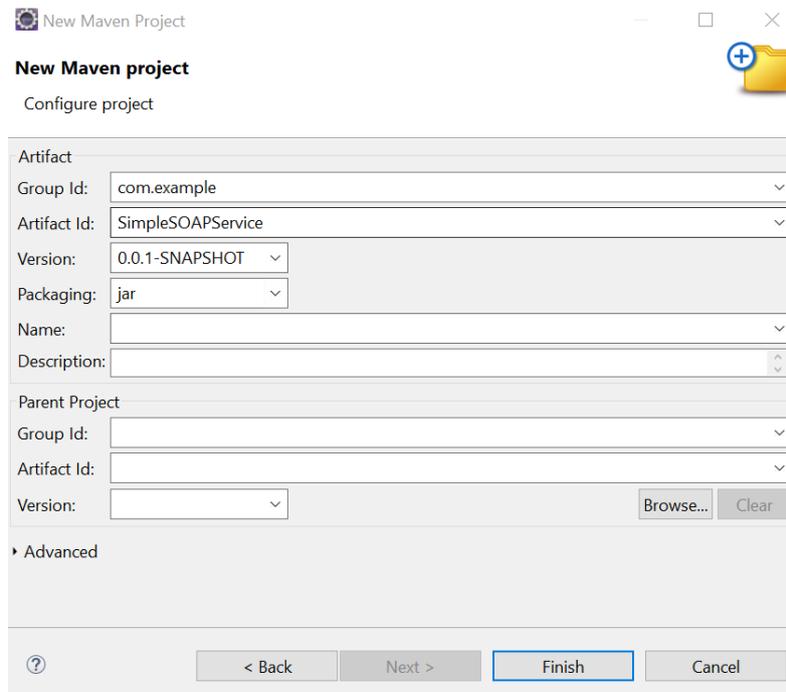
# Java Programming Language

**Project Name:** SimpleSOAPService

**Step 1:** Open Eclipse IDE, Create a Maven Project, Check – Create a simple project (skip archetype selection) and Click Next.



**Step 2:** In New Maven Project window add the following details – Group id: com.example, Artifact Id: SimpleSOAPService and Version: 0.0.1- SNAPSHOT and click Finish



**Step 3:** After creating the Maven project, update the pom.xml to include the necessary dependencies for Apache CXF and JAX-WS. Here's the full pom.xml file.

### **pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>SimpleSOAPService</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <!-- Apache CXF for JAX-WS -->
    <dependency>
      <groupId>org.apache.cxf</groupId>
      <artifactId>cxf-rt-frontend-jaxws</artifactId>
      <version>3.4.5</version>
    </dependency>

    <!-- Apache CXF HTTP Transport -->
    <dependency>
      <groupId>org.apache.cxf</groupId>
      <artifactId>cxf-rt-transport-http</artifactId>
      <version>3.4.5</version>
    </dependency>

    <!-- Apache CXF HTTP Jetty Transport (for embedded HTTP server) -->
    <dependency>
      <groupId>org.apache.cxf</groupId>
      <artifactId>cxf-rt-transport-http-jetty</artifactId>
      <version>3.4.5</version>
    </dependency>

    <!-- Jakarta XML WS (JAX-WS API) -->
    <dependency>
      <groupId>jakarta.xml.ws</groupId>
      <artifactId>jakarta.xml.ws-api</artifactId>
      <version>2.3.3</version>
    </dependency>

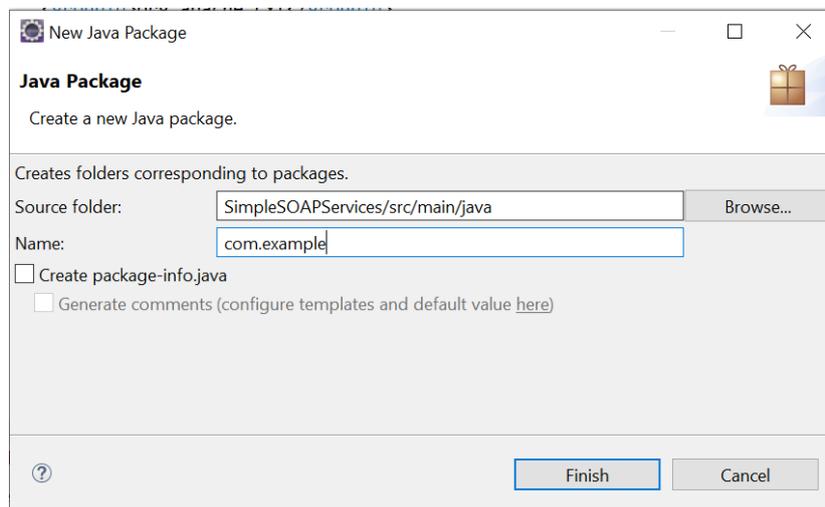
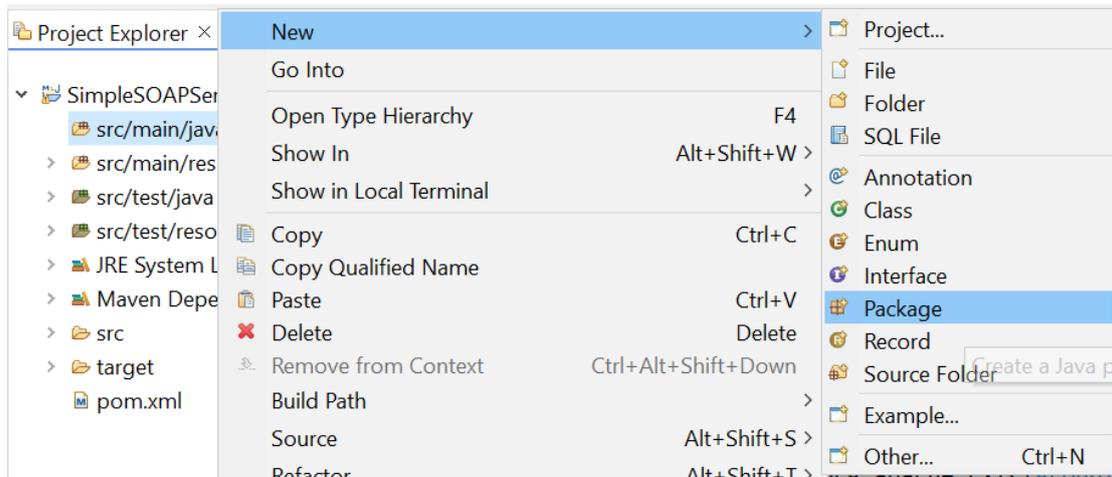
    <!-- Java Annotations -->
    <dependency>
      <groupId>javax.annotation</groupId>
      <artifactId>javax.annotation-api</artifactId>
      <version>1.3.2</version>
    </dependency>
  </dependencies>
</project>
```

After adding dependencies Save it by Ctrl+S

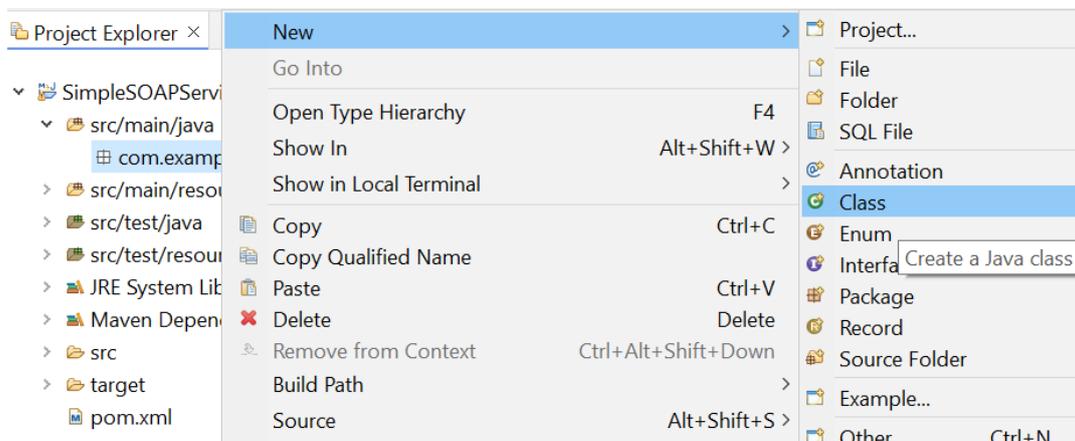
#### Step 4: Create the Service Interface

Now, create a simple SOAP service interface to define the operations.

1. **Right-click** on src/main/java → **New** → **Package** and name it com.example



1. **Right-click** on the com.example package → **New** → **Class** and name it CalculatorService.java.



**Java Class**  
Create a new Java class.

Source folder: SimpleSOAPServices/src/main/java Browse...

Package: com.example Browse...

Enclosing type: Browse...

Name: CalculatorService

Modifiers:  public  package  private  protected  
 abstract  final  static

Superclass: java.lang.Object Browse...

Interfaces: Add...  
Remove

Which method stubs would you like to create?  
 public static void main(String[] args)  
 Constructors from superclass  
 Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))  
 Generate comments

? Finish Cancel

**File Name:** CalculatorService.java

**Code:**

```
package com.example;
```

```
import javax.jws.WebMethod;
```

```
import javax.jws.WebService;
```

```
@WebService
```

```
public interface CalculatorService {
    @WebMethod
    int add(int num1, int num2);
```

```
    @WebMethod
    int subtract(int num1, int num2);
```

```
}
```

**Save it**

### Step 5: Implement the Service

Now, create a class that implements the CalculatorService interface.

1. **Right-click** on com.example → **New** → **Class** and name it CalculatorServiceImpl.java.
2. Add the following code to CalculatorServiceImpl.java:

**File Name:** CalculatorServiceImpl.java

#### Code:

```
package com.example;
```

```
import javax.ws.WebService;
```

```
@WebService(endpointInterface = "com.example.CalculatorService")
```

```
public class CalculatorServiceImpl implements CalculatorService {
```

```
    @Override
```

```
    public int add(int num1, int num2) {
```

```
        return num1 + num2;
```

```
    }
```

```
    @Override
```

```
    public int subtract(int num1, int num2) {
```

```
        return num1 - num2;
```

```
    }
```

```
}
```

**Save It**

### Step 6: Create the SOAP Server

Next, create a class to publish the SOAP service.

1. **Right-click** on com.example → **New** → **Class** and name it SOAPServer.java.
2. Add the following code to SOAPServer.java:

**File Name:** SOAPServer.java

**Code:** package com.example;

```
import javax.xml.ws.Endpoint;
```

```
public class SOAPServer {
```

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

```
        // Create an instance of the service implementation
```

```
        CalculatorServiceImpl implementor = new CalculatorServiceImpl();
```

```
        // Publish the service
```

```
        String address = "http://localhost:8080/CalculatorService";
```

```
        Endpoint.publish(address, implementor);
```

```
        System.out.println("SOAP Service started at " + address);
```

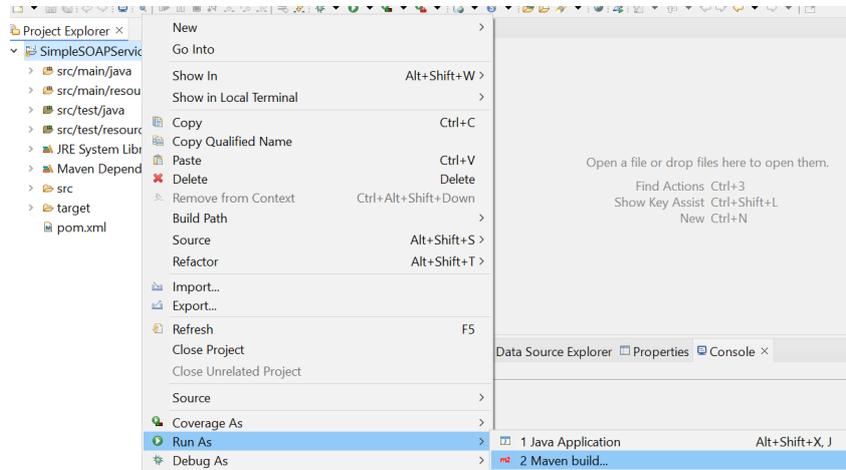
```
    }
```

```
}
```

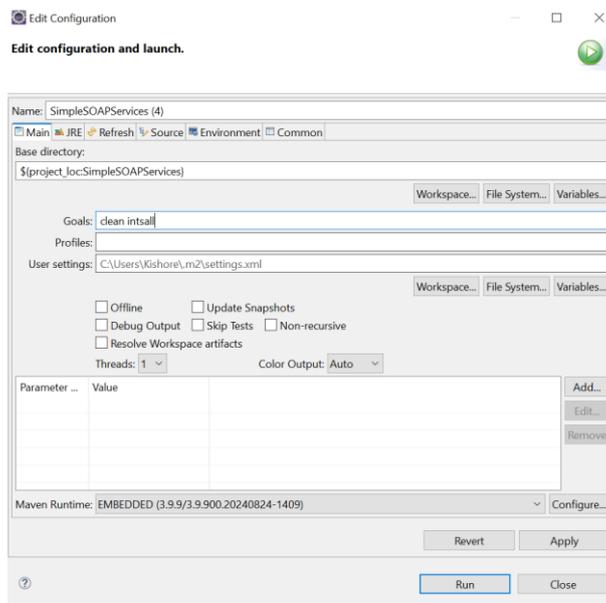
**Save It**

## Output:

**mvn clean install:** The clean install command is commonly used in Maven to build and install a project. The purpose of running clean is to remove any previously compiled code or artifacts, ensuring that the next build starts fresh without using any old files that might cause conflicts or errors.



## In Goal type clean install and run



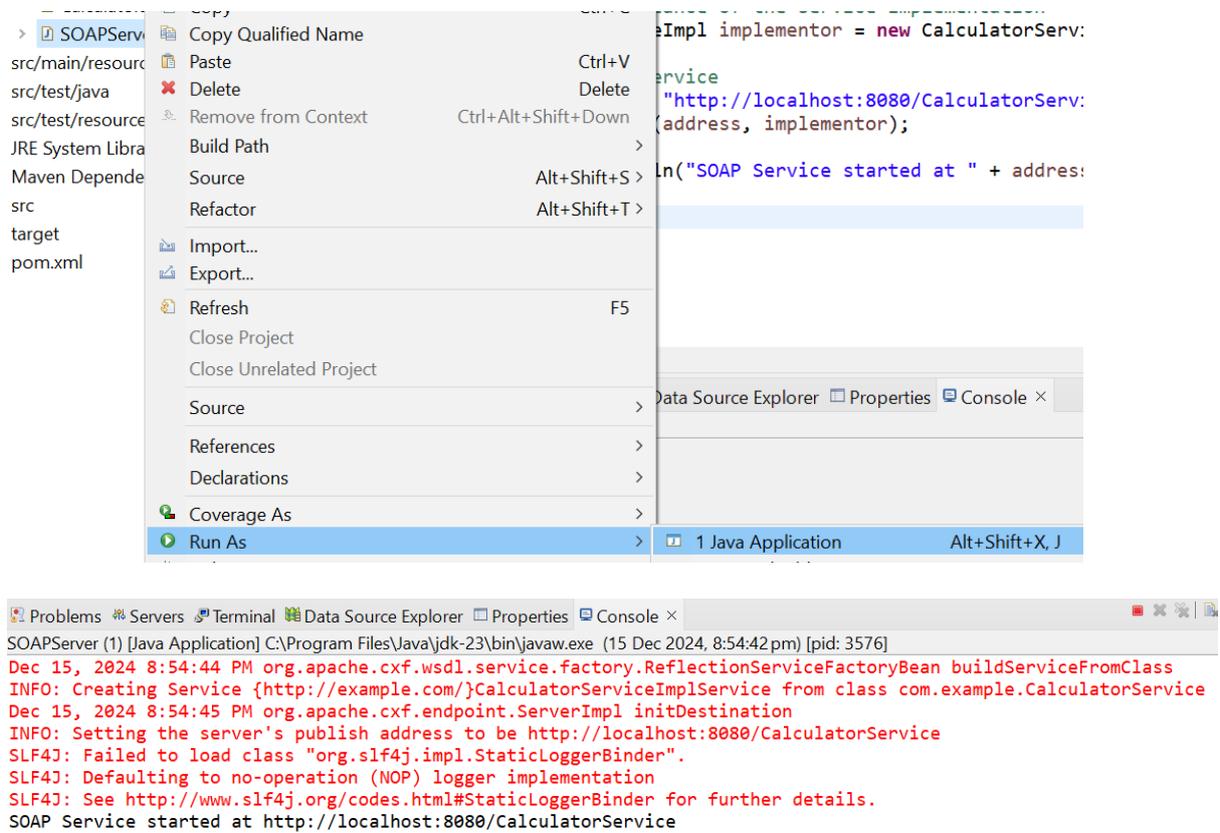
```
<terminated> SimpleSOAPServices (5) [Maven Build] C:\Program Files\Java\jdk-23\bin\javaw.exe (15 Dec 2024, 9:44:13 pm - 9:44:19 pm) [pid: 11976]
[INFO] --- surefire:3.2.5:test (default-test) @ SimpleSOAPServices ---
[INFO]
[INFO] --- jar:3.4.1:jar (default-jar) @ SimpleSOAPServices ---
[INFO] Building jar: C:\Users\Kishore\eclipse-workspace\SimpleSOAPServices\target\SimpleSOAPServices-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- install:3.1.2:install (default-install) @ SimpleSOAPServices ---
[INFO] Installing C:\Users\Kishore\eclipse-workspace\SimpleSOAPServices\pom.xml to C:\Users\Kishore\m2\repository\com\example\SimpleSOAF
[INFO] Installing C:\Users\Kishore\eclipse-workspace\SimpleSOAPServices\target\SimpleSOAPServices-0.0.1-SNAPSHOT.jar to C:\Users\Kishore\
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 3.921 s
[INFO] Finished at: 2024-12-15T21:44:19+05:30
[INFO]
[INFO]
```

Activate Windows  
Go to Settings to activate Windows.

## Run the Application

1. Right-click on SOAPServer.java → Run As → Java Application.
2. You should see the following output:

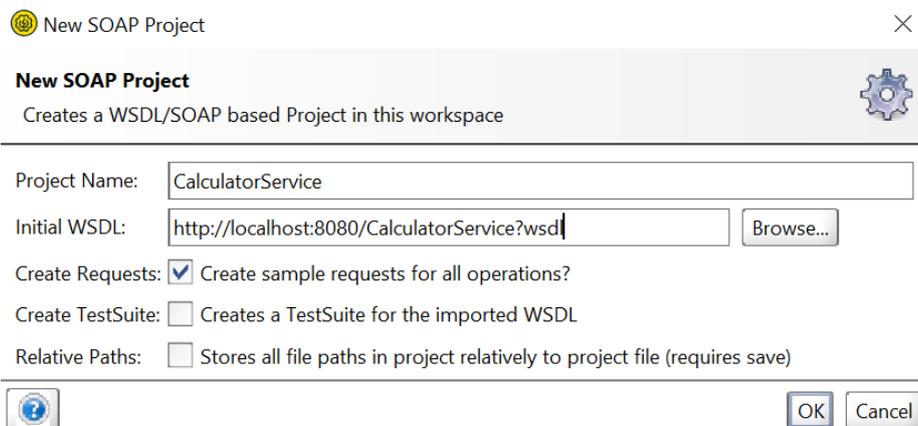
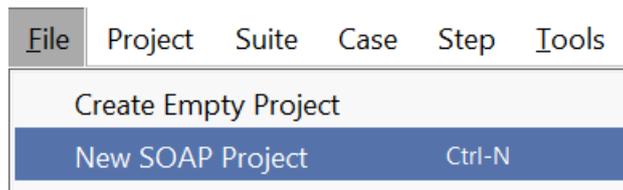
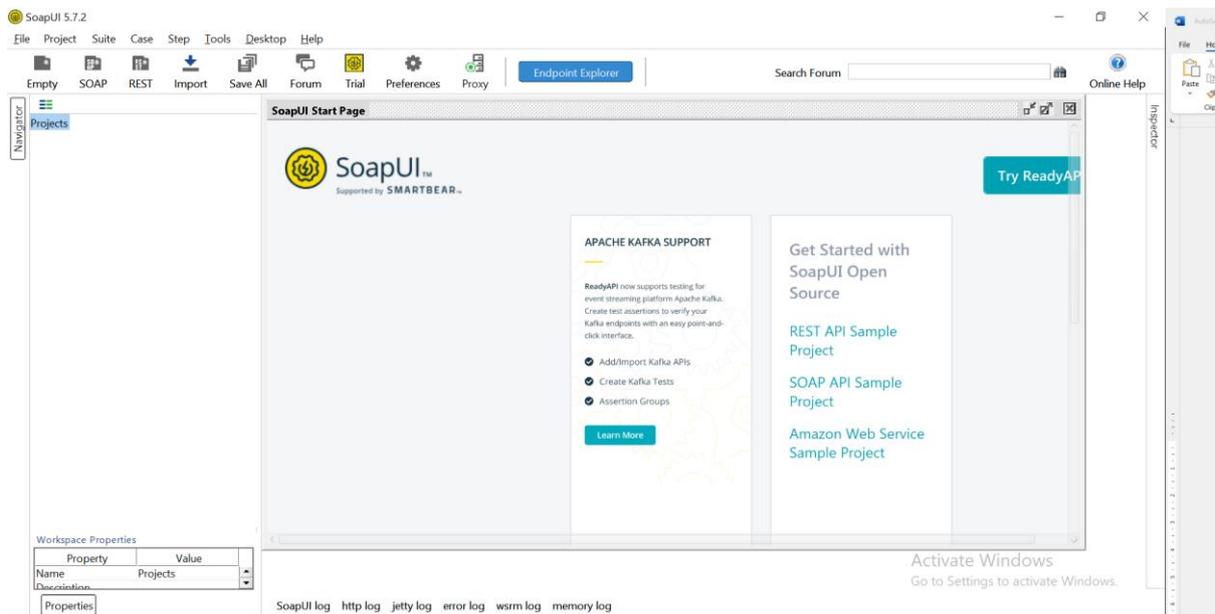
SOAP Service started at <http://localhost:8080/CalculatorService>



## Step 7. Test the SOAP Service Using SOAPUI

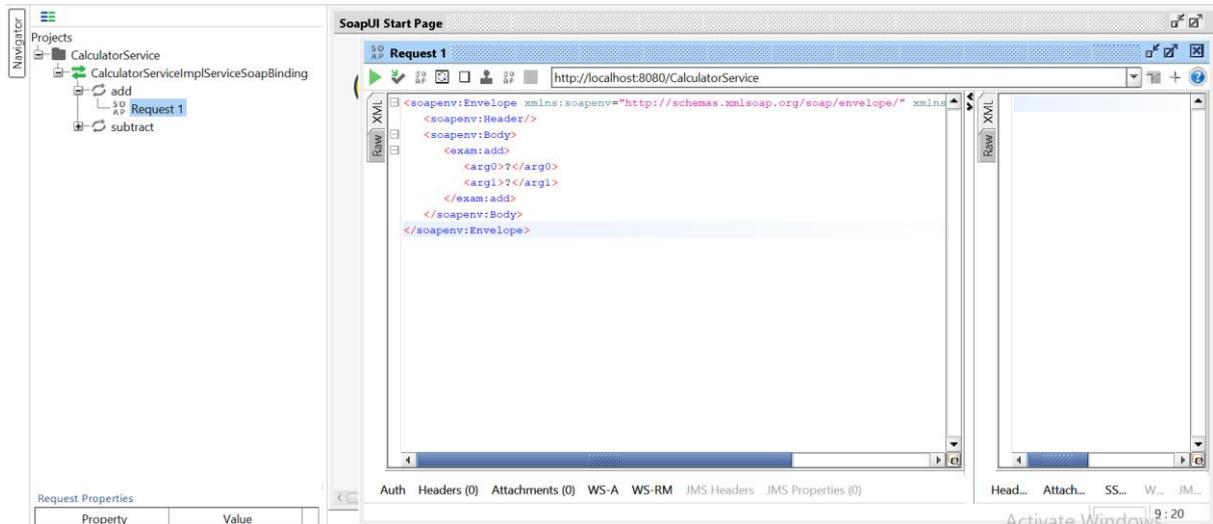
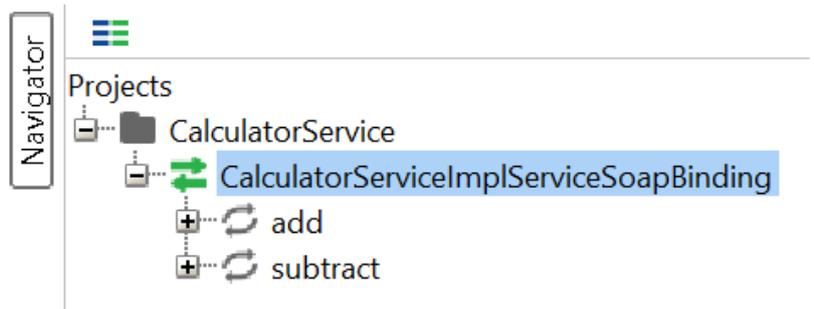
1. Open SOAPUI and create a new project:
  1. Click File → New SOAP Project.
  2. In the Project Name field, enter CalculatorService.
  3. In the Initial WSDL field, enter:  
**http://localhost:8080/CalculatorService?wsdl**

### SOAPUI:



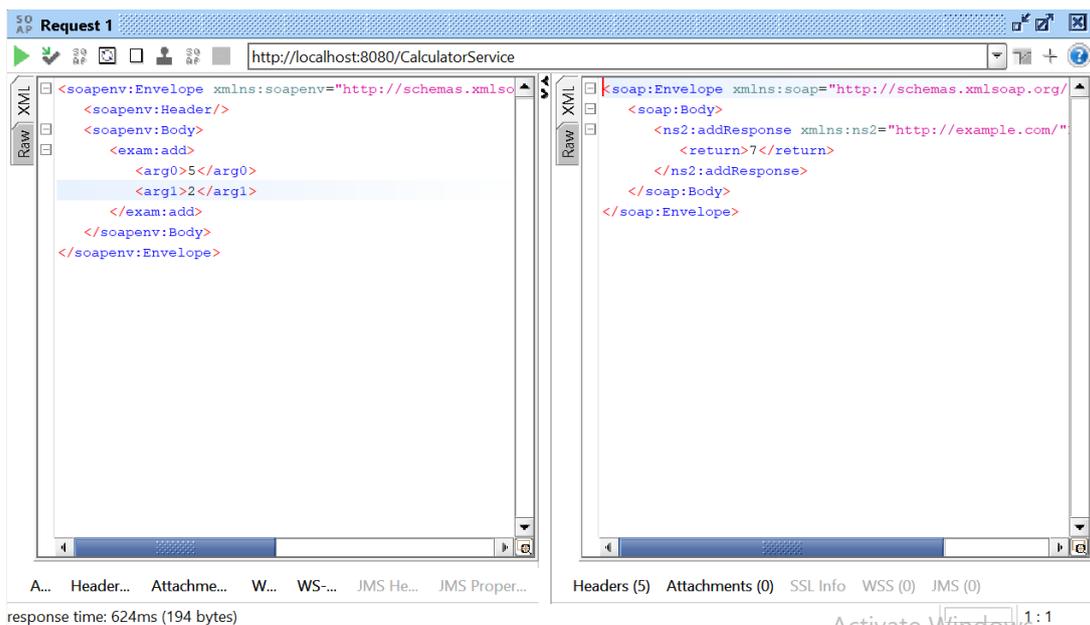
Click OK

SOAPUI will import the WSDL and display the operations (add and subtract). You can now test the SOAP service by right-clicking on the operations and sending requests.



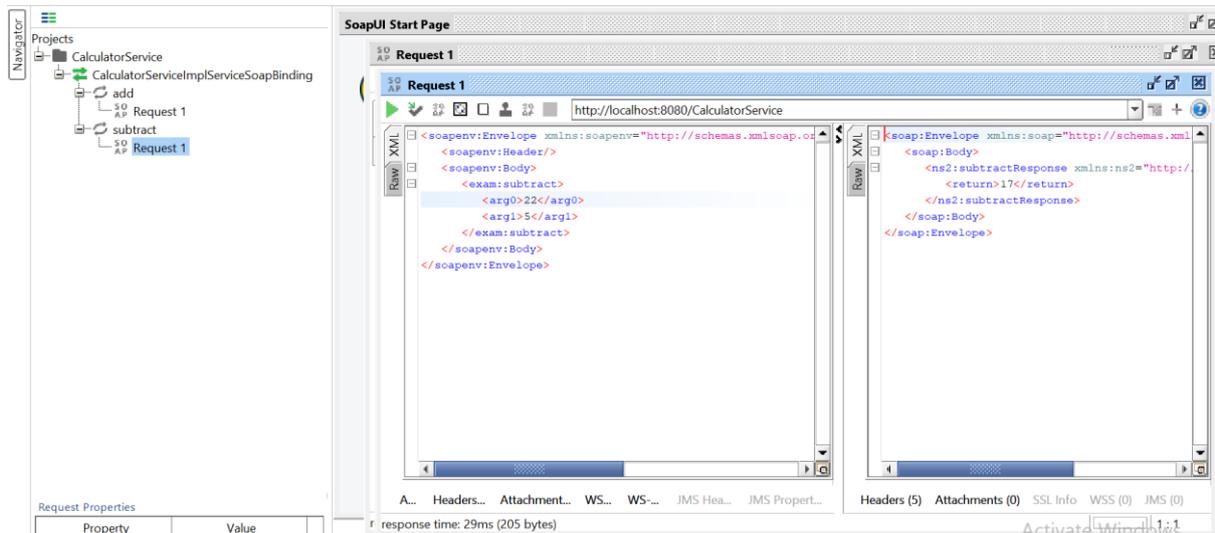
### add > Request 1

Here, in add we got the request and edited 5 and 2 in argument tags and output displayed in xml.



## subtract > Request 1

Here, in subtract we got the request and edited 22 and 5 in argument tags and output displayed in xml.



**Conclusion:** Why Use SoapUI to Create SOAP Services and Get Output in XML for Add and Subtract Operations

SoapUI is a popular tool for testing and interacting with SOAP (Simple Object Access Protocol) web services. It is used to create and test SOAP services, allowing developers and testers to ensure that their web services function correctly and handle various input scenarios. Logic to Work with SOAP Services in SoapUI:

Creating SOAP Service:

- WSDL (Web Services Description Language): SoapUI uses the WSDL file to generate the request and response structure for SOAP services. The WSDL defines the operations, input/output messages, and service endpoint.
- SOAP Request and Response: In SoapUI, you can create SOAP requests based on the operations defined in the WSDL. For operations like "Add" and "Subtract," the SOAP request will include the parameters (e.g., two numbers) to be processed by the web service.

-----

## Practical 3

Create a Simple REST Service

### Software Tools Required:

- **Code Editor:** Visual Studio Code (VS Code)
- **API Testing Software Application:** Postman
- **Framework:** Flask (micro web framework)

### Downloads Required:

- Python: [Download Python | Python.org](https://www.python.org/downloads/)
- Postman: [Download Postman | Get Started for Free](https://www.postman.com/downloads/)  
(Create Account/ Log in with Google)

After Downloading Python, Set the Path in Environment Variable in User Variables > Path > Edit > New and Paste the Path for Python.

New version is 3.13, Here the version is 3.12 so following the same for new version.

**Till Python312:** Copy Directory Path

Start > Python 3.13.1 (64 bit) > Open File Location > Python 3.13.1 (64 bit)

**Till Python312\Scripts:** Copy the Directory Path

Start > Python 3.13.1 (64 bit) > Open File Location > Python 3.13.1 (64 bit) > Open File Location > Scripts

### Demonstration:

```
C:\Users\Kishore\AppData\Local\Programs\Python\Python312\Scripts\  
C:\Users\Kishore\AppData\Local\Programs\Python\Python312\  
C:\Users\Kishore\AppData\Local\Programs\Python\Python312\Scripts\
```

### Check Version:

```
C:\Users\Kishore>python --version  
Python 3.12.6  
  
C:\Users\Kishore>_
```

# Python Programming Language

**Step 1:** Install Flask: Make sure you have Python installed on your system. Open cmd and type the following command

**pip install Flask**

**Demonstration:**

```
C:\Users\Kishore>pip install Flask
Requirement already satisfied: Flask in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (3.0.3)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (from Flask) (3.0.3)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (from Flask) (3.1.4)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (from Flask) (2.2.0)
Requirement already satisfied: click>=8.1.3 in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (from Flask) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (from Flask) (1.8.2)
Requirement already satisfied: colorama in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (from click>=8.1.3->Flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (from Jinja2>=3.1.2->Flask) (2.1.5)

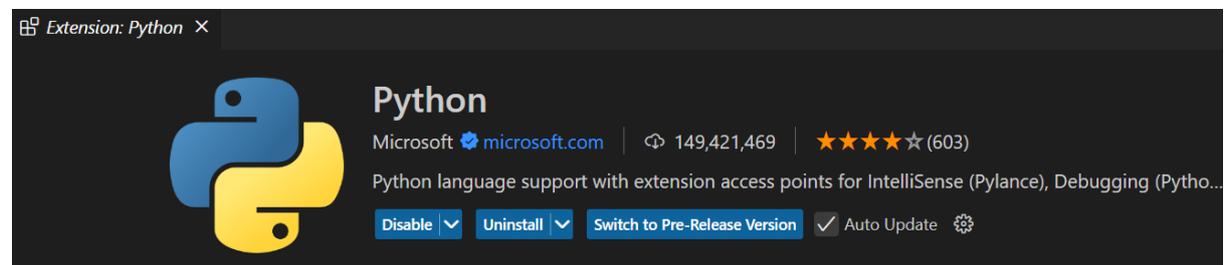
[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

**Step 2:** Create a folder name SimpleRESTService, Open with VS Code and create python file.

Check and Install Extensions for Python

**Extensions:**

**Python**



**Python Debugger**



**Filename:** app.py

**Code:**

```
from flask import Flask, jsonify, request

app = Flask(__name__)

# Sample data
data = [
    {'id': 1, 'name': 'Item 1'},
    {'id': 2, 'name': 'Item 2'},
]

# Endpoint to get all items
@app.route('/items', methods=['GET'])
def get_items():
    return jsonify({'items': data})

# Endpoint to get a specific item by ID
@app.route('/items/<int:item_id>', methods=['GET'])
def get_item(item_id):
    item = next((item for item in data if item['id'] == item_id), None)
    if item:
        return jsonify({'item': item})
    else:
        return jsonify({'message': 'Item not found'}), 404
```

```
# Endpoint to add a new item
@app.route('/items', methods=['POST'])
def add_item():
    new_item = {'id': len(data) + 1, 'name': request.json['name']}
    data.append(new_item)
    return jsonify({'message': 'Item added successfully', 'item': new_item}), 201

# Run the application
if __name__ == '__main__':
    app.run(debug=True)
```

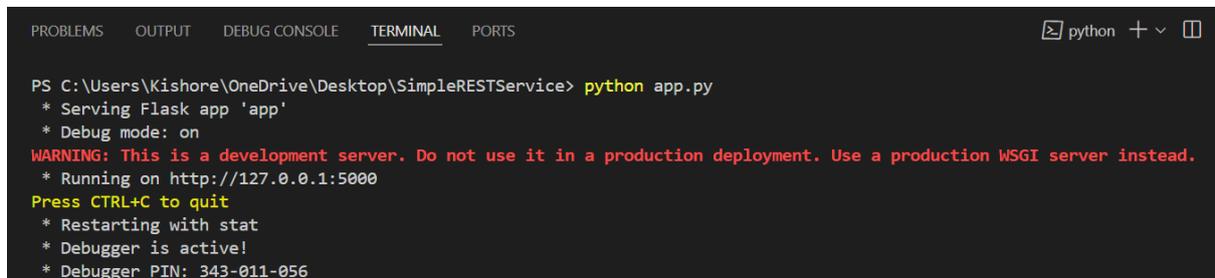
### Output:

View>Terminal

Type

Path> **python app.py**

### Demonstration:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python + v □
PS C:\Users\Kishore\OneDrive\Desktop\SimpleRESTService> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 343-011-056
```

# Postman

## API Testing Software Application to Test the API

### Open Postman

# Endpoint to get all items

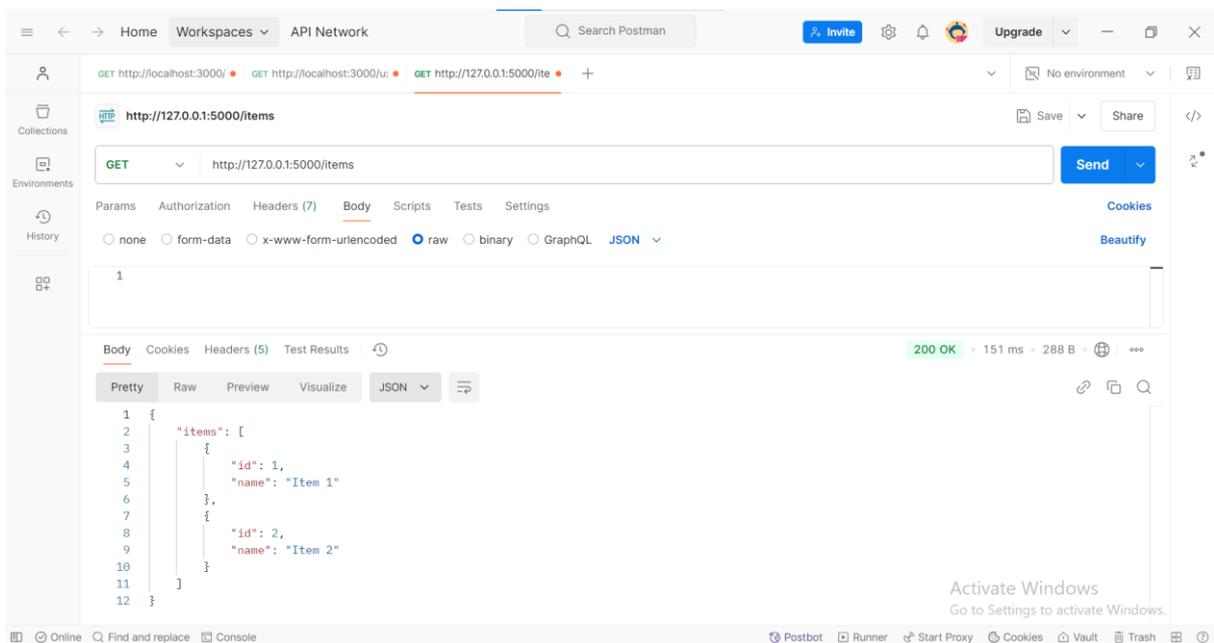
Send a GET Request:

URL: <http://127.0.0.1:5000/items>

Response:

```
{
  "items": [
    {
      "id": 1,
      "name": "Item 1"
    },
    {
      "id": 2,
      "name": "Item 2"
    }
  ]
}
```

Demonstration:



The screenshot displays the Postman interface for a REST client. The top navigation bar shows 'Home', 'Workspaces', and 'API Network'. The main workspace is set to 'http://127.0.0.1:5000/items'. The request method is 'GET' and the body is set to 'raw'. The response status is '200 OK' with a response time of '151 ms' and a size of '288 B'. The response body is displayed in 'Pretty' format, showing a JSON array of two items: 'Item 1' (id: 1) and 'Item 2' (id: 2). The interface includes various tabs like 'Body', 'Cookies', 'Headers', and 'Test Results', and a sidebar with 'Collections', 'Environments', and 'History'.

## # Endpoint to get a specific item by ID

### Send a GET Request:

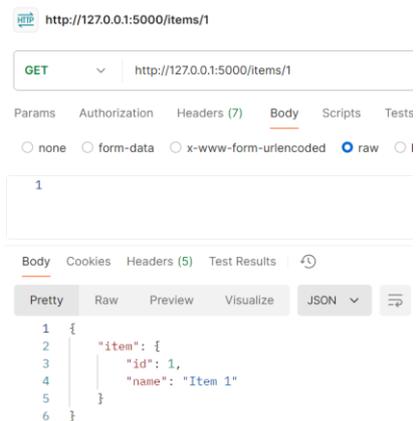
URL: <http://127.0.0.1:5000/items/1>

### Response:

```
{
  "item": {
    "id": 1,
    "name": "Item 1"
  }
}
```

Provides specific id from Sample Data available i.e., URL: <http://127.0.0.1:5000/items/1>

### Demonstration:



http://127.0.0.1:5000/items/1

GET http://127.0.0.1:5000/items/1

Params Authorization Headers (7) Body Scripts Tests

none  form-data  x-www-form-urlencoded  raw

1

Body Cookies Headers (5) Test Results

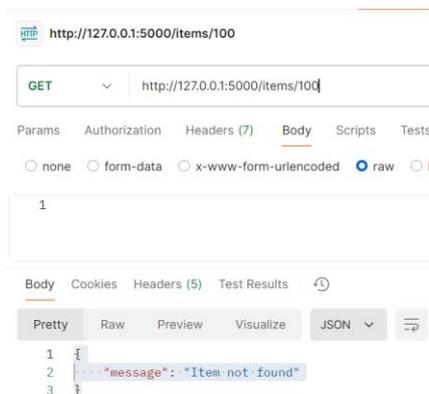
Pretty Raw Preview Visualize JSON

```
1 {
2   "item": {
3     "id": 1,
4     "name": "Item 1"
5   }
6 }
```

### Throws Error when not item not found from Sample Data. Example

URL: <http://127.0.0.1:5000/items/100>

### Demonstration:



http://127.0.0.1:5000/items/100

GET http://127.0.0.1:5000/items/100

Params Authorization Headers (7) Body Scripts Tests

none  form-data  x-www-form-urlencoded  raw

1

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Item not found"
3 }
```

## # Endpoint to add a new item

### Send a POST Request:

**URL:** <http://127.0.0.1:5000/items>

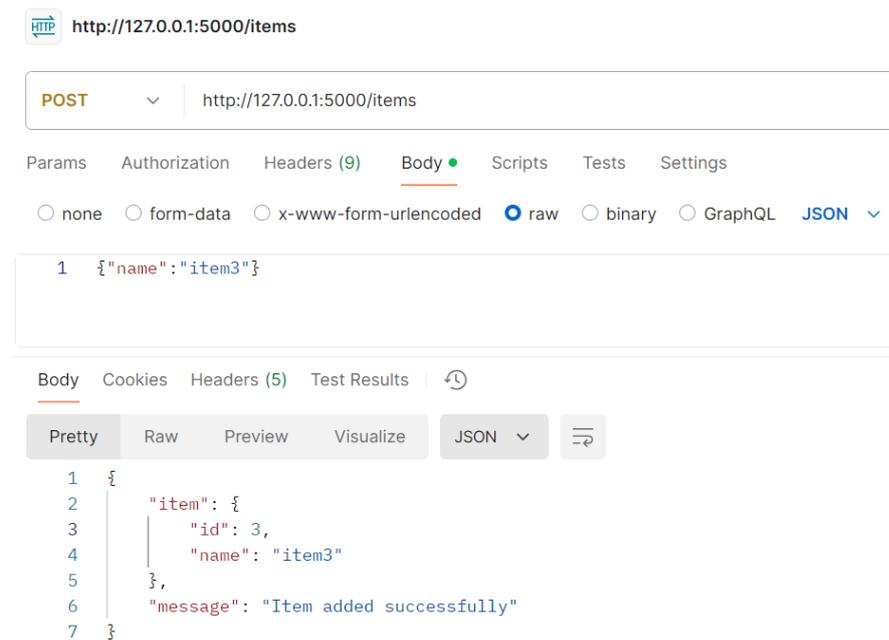
**Body:** Set the body to raw JSON and include:

```
{"name": "item3"}
```

### Response:

```
{  
  "item": {  
    "id": 3,  
    "name": "item3"  
  },  
  "message": "Item added successfully"  
}
```

### Demonstration:



The screenshot displays a REST client interface for the endpoint `http://127.0.0.1:5000/items`. The request method is set to `POST`. The request body is configured as `raw` JSON, containing the payload `{"name": "item3"}`. The response is shown in a `JSON` format, displaying the following structure:

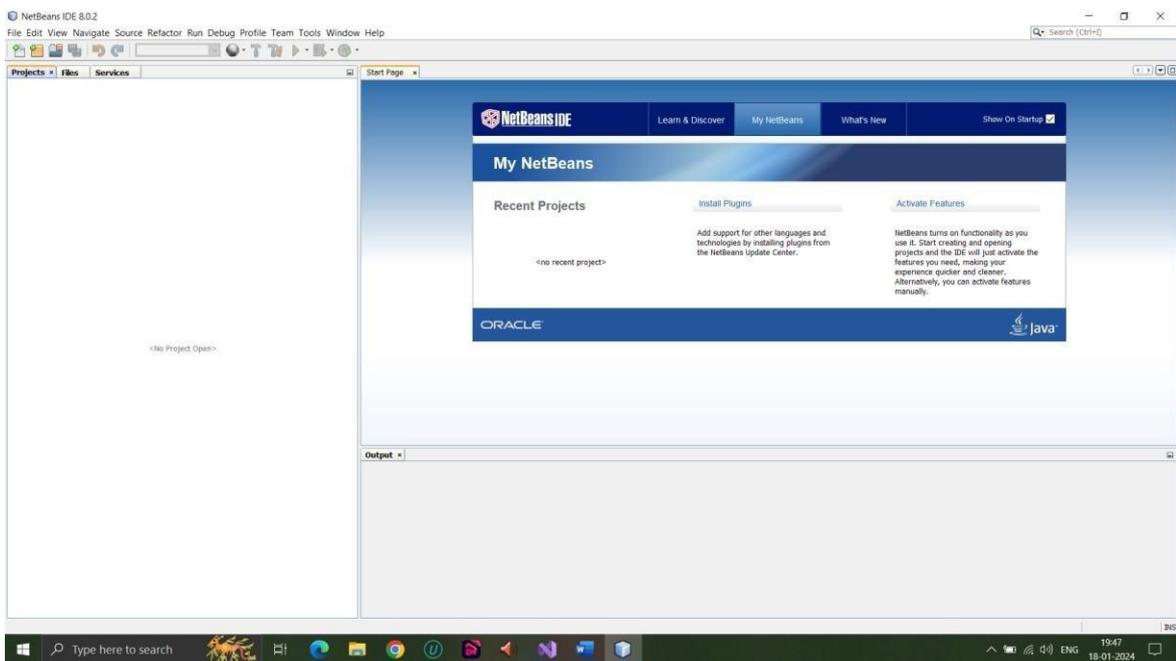
```
1 {  
2   "item": {  
3     "id": 3,  
4     "name": "item3"  
5   },  
6   "message": "Item added successfully"  
7 }
```

# Practical-1

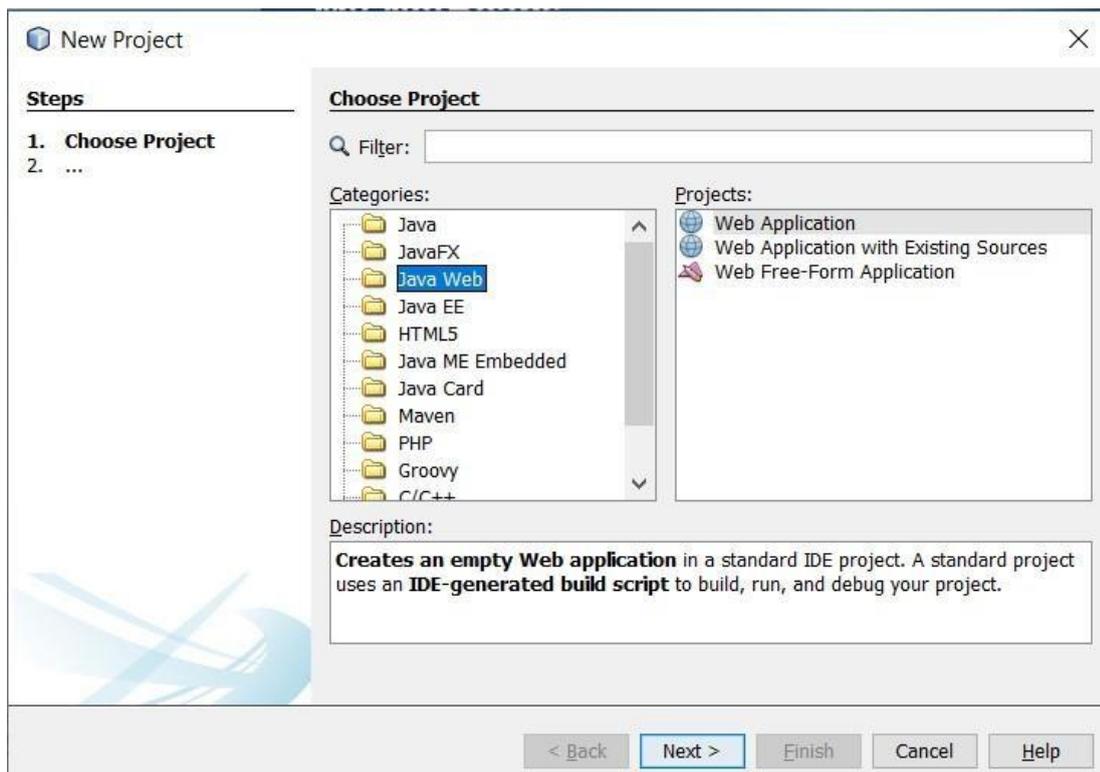
Create a Simple SOAP service.

## Steps:

1] Open the NetBeans , and you will get the following screen. Close the start page.



2] Now click on the file tab and click on new project you will get the following screen :



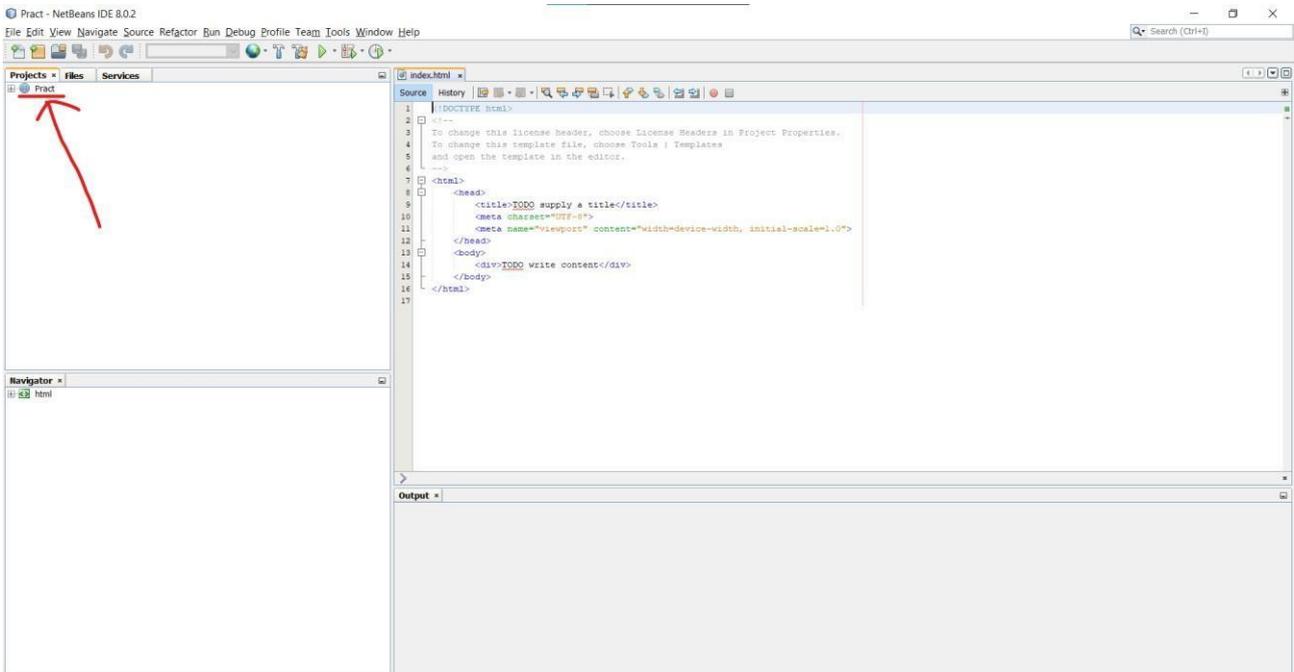
3] In Categories select Java Web and in Projects , Select Web Application .After selecting click onnext . You will get the following window:

The screenshot shows the 'New Web Application' dialog box with the 'Name and Location' step selected. The 'Steps' list on the left includes: 1. Choose Project, 2. Name and Location (highlighted), 3. Server and Settings, and 4. Frameworks. The 'Name and Location' section contains the following fields: 'Project Name' with the value 'Pract1', 'Project Location' with the value 'C:\Users\LENOVO\Documents\NetBeansProjects' and a 'Browse...' button, and 'Project Folder' with the value 'C:\Users\LENOVO\Documents\NetBeansProjects\Pract1'. Below these is a checkbox for 'Use Dedicated Folder for Storing Libraries' which is unchecked, followed by a 'Libraries Folder' field and a 'Browse...' button. A note states: 'Different users and projects can share the same compilation libraries (see Help for details)'. At the bottom, a red error message reads: 'Project Folder already exists and is not empty.' The bottom navigation bar contains buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

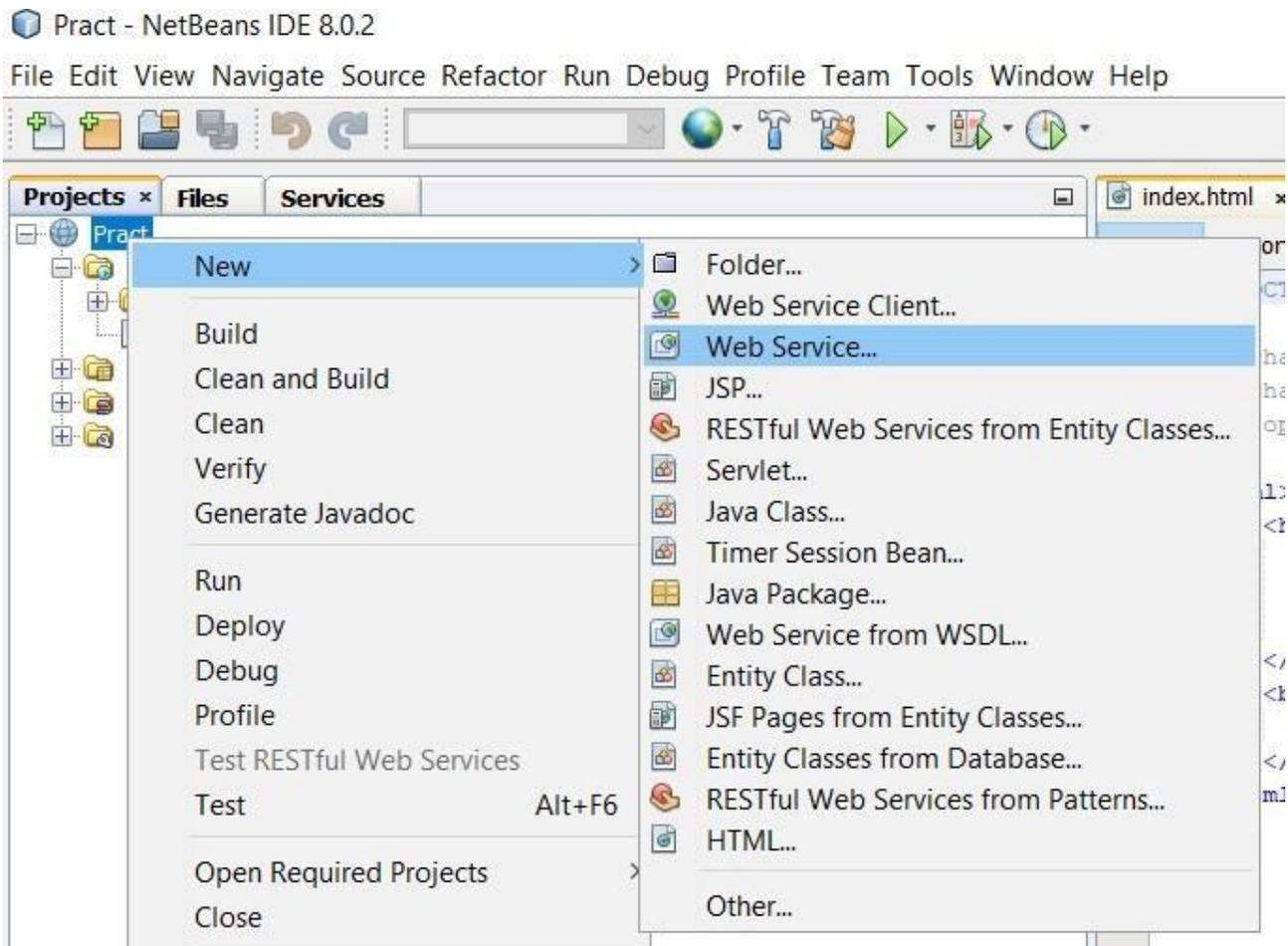
4] Now Give name to the Project Name: , and click on next . you will get the following windowthen . again click on finish:

The screenshot shows the 'New Web Application' dialog box with the 'Server and Settings' step selected. The 'Steps' list on the left includes: 1. Choose Project, 2. Name and Location, 3. Server and Settings (highlighted), and 4. Frameworks. The 'Server and Settings' section contains the following fields: 'Add to Enterprise Application' with a dropdown menu set to '<None>', 'Server' with a dropdown menu set to 'GlassFish Server 4.1' and an 'Add...' button, 'Java EE Version' with a dropdown menu set to 'Java EE 7 Web', and 'Context Path' with the value '/Pract'. The bottom navigation bar contains buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

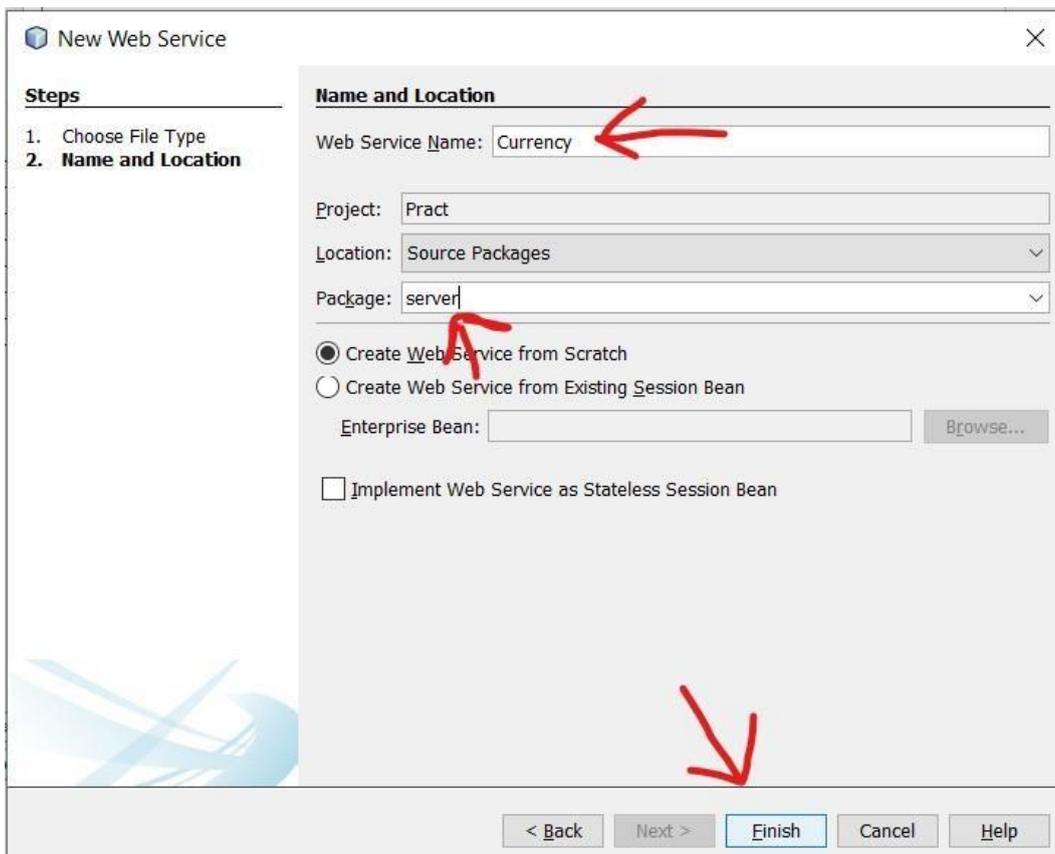
5] You will get the following screen now carefully see in projects section your recently created project appears double click to expand it:



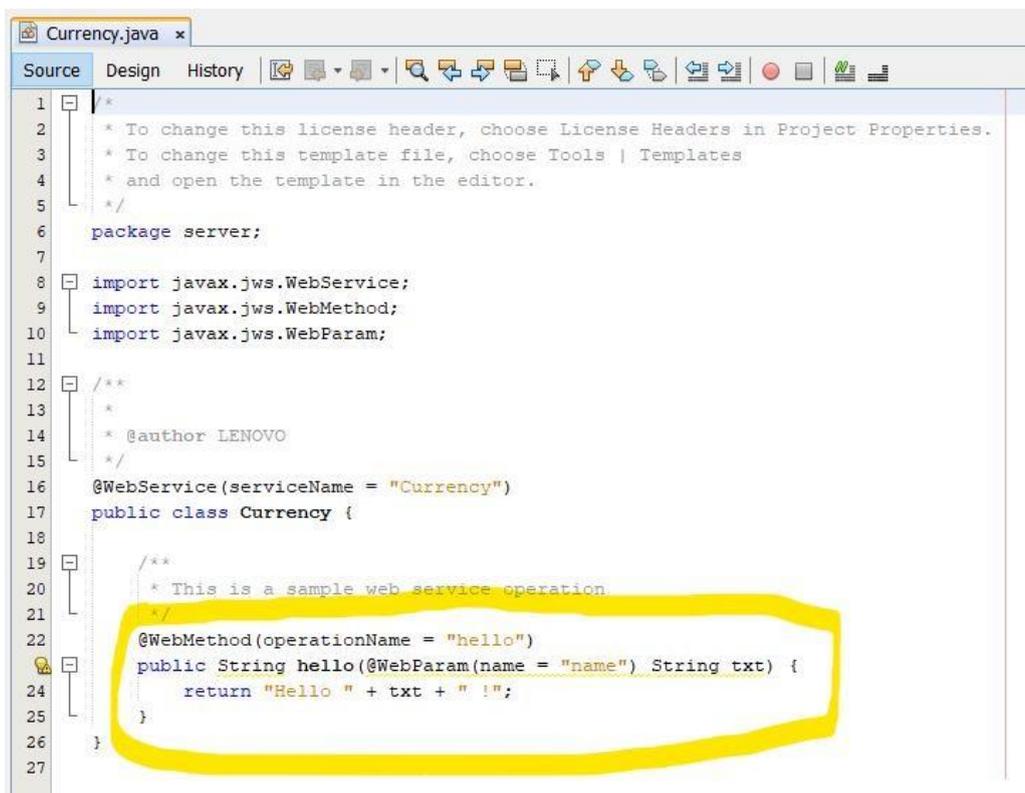
6] Now Right click on the project and select new and then select Web Service ,As shown Below:



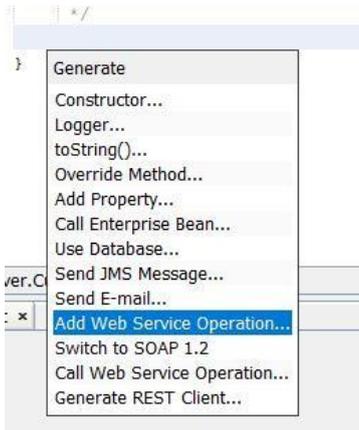
7] after clicking Web Service following window should appear , now give name to web service and give package as "server". As shown in the image:



After clicking finish you should get the following window erase the mentioned code :

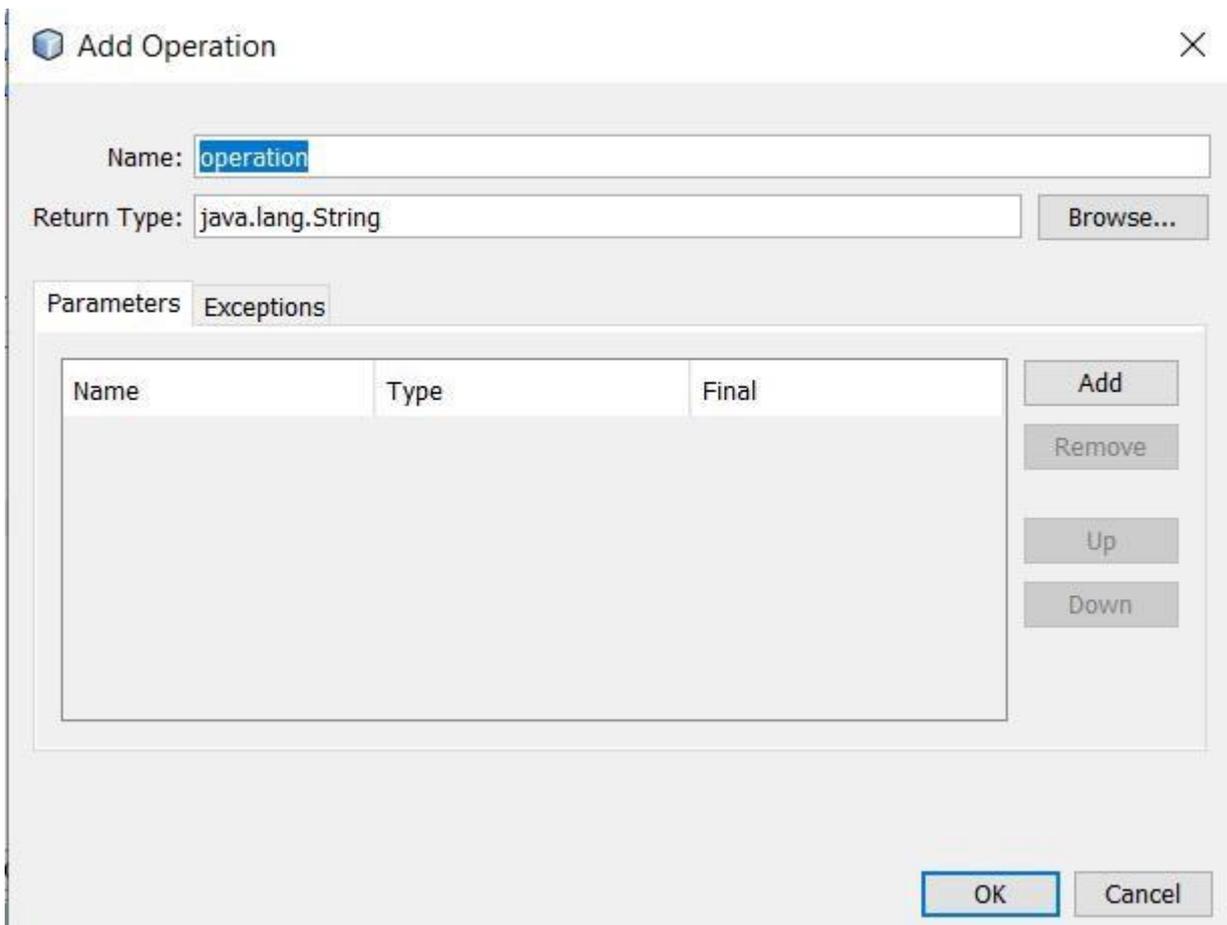


8] Now right click any where and click on insert code and select Add Web Service Operation :

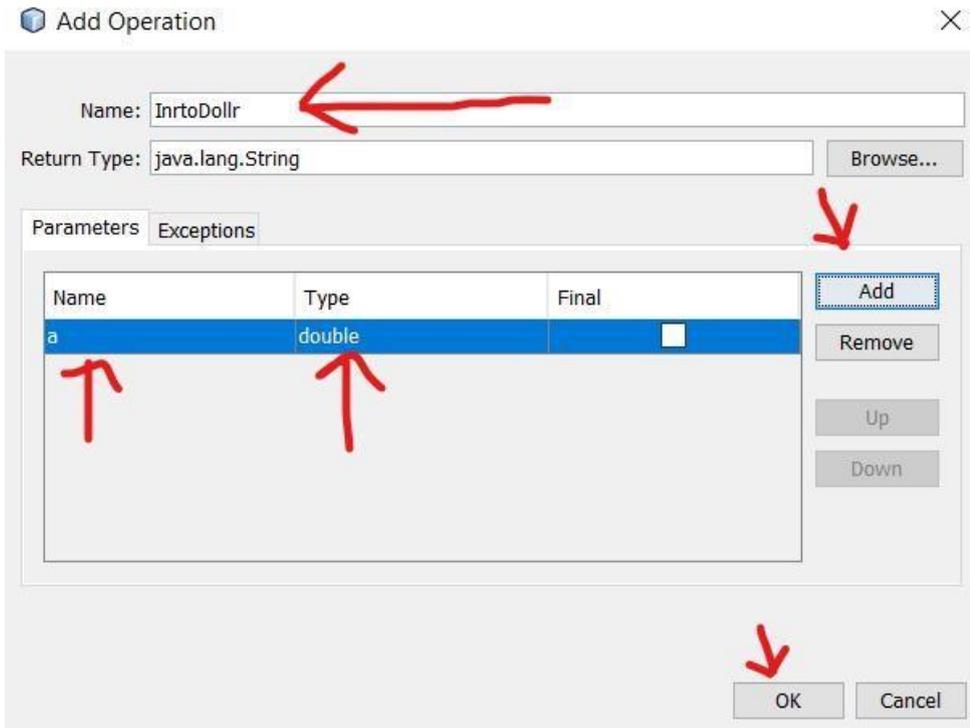


9] the following window would appear :

Just give name to the method or operation and click on add button to add parameters to the method as here we are converting dollar to rupees we should need only one parameter .



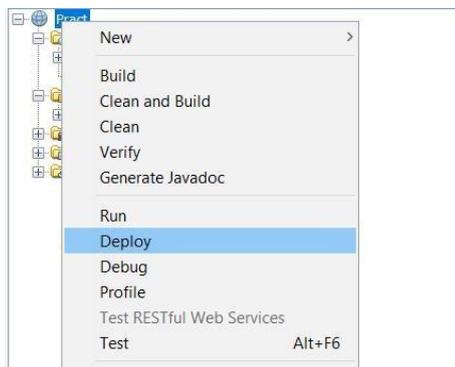
10] give the name to the variable select data type as double and click on ok as shown below:



11] After clicking ok code will autogenerate , make changes In that code as mentioned below:

```
@WebMethod(operationName = "InrtoDollar")
public String InrtoDollar(@WebParam(name = "a") double a) {
    //TODO write your implementation code here:
    return "The Indian rupees "+a+" in Dollars is "+(a/83.17);
}
```

12] now our web service is ready now right click on project and click on deploy :



13] now right click on webservice and click on test web service you will get the following output:

CurrencyConverter Web Service

localhost:8080/practical1/CurrencyConverter?Tester

# CurrencyConverter Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

---

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

**Methods :**

public abstract java.lang.String server.CurrencyConverter.inrtoDollar(double)

inrtoDollar (  )

Method invocation trace

localhost:8080/practical1/CurrencyConverter?Tester

## inrtoDollar Method invocation

---

**Method parameter(s)**

Type	Value
double	1233

---

**Method returned**

java.lang.String : "The Indian rupees 1233.0 in Dollars is 14.825057111939401"

---

**SOAP Request**

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:InrtoDollar xmlns:ns2="http://server/">
      <a>1233.0</a>
    </ns2:InrtoDollar>
  </S:Body>
</S:Envelope>
```

---

**SOAP Response**

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:InrtoDollarResponse xmlns:ns2="http://server/">
      <return>The Indian rupees 1233.0 in Dollars is 14.825057111939401</return>
    </ns2:InrtoDollarResponse>
  </S:Body>
</S:Envelope>
```

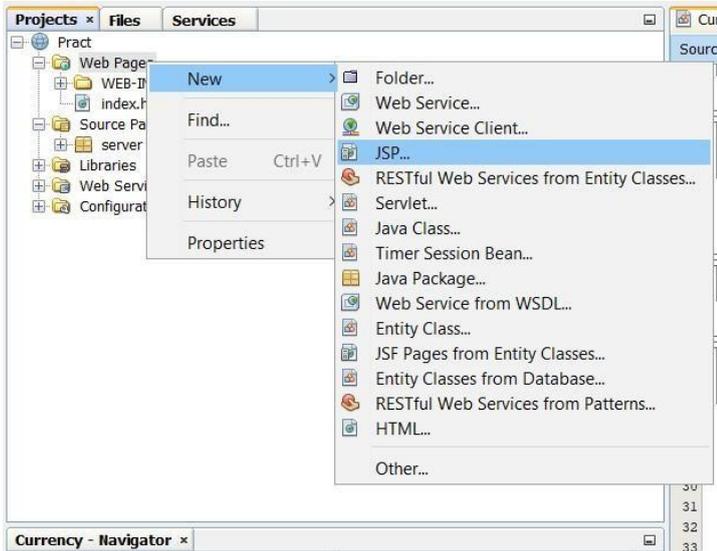
So this is how we created our web service and deployed it.

---

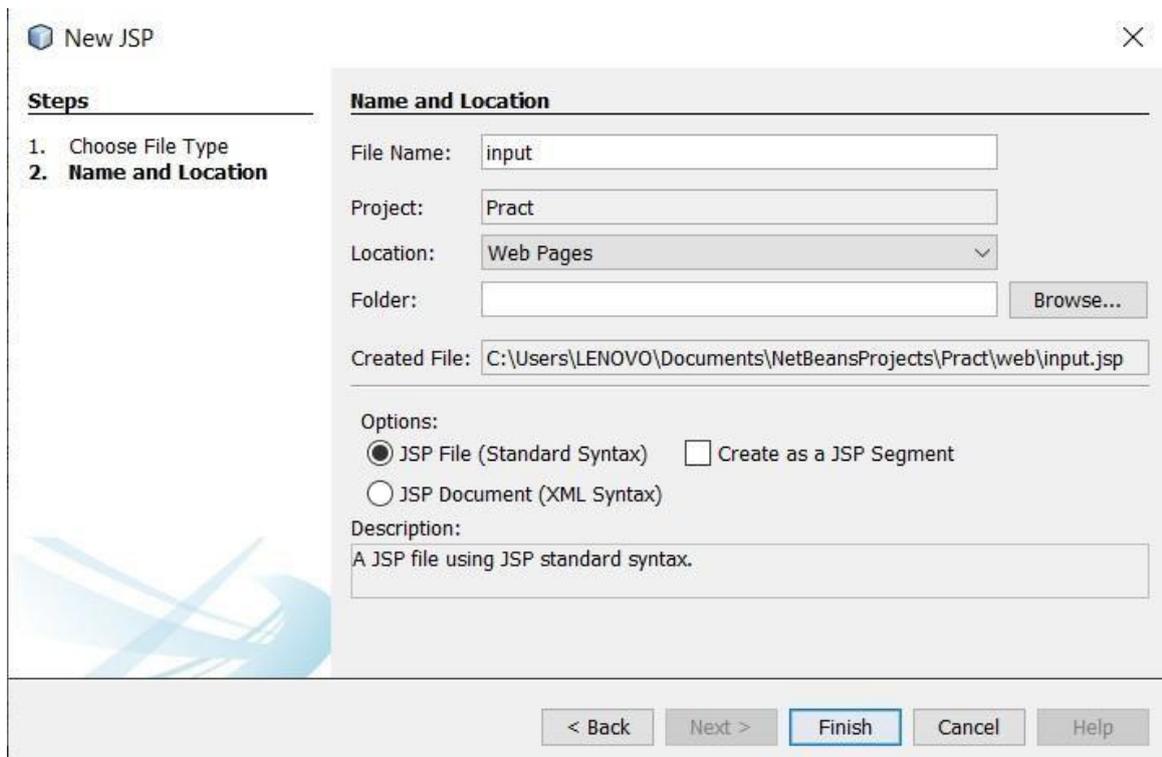
## Creating a java Client using jsp

---

14] now our web service is successfully deployed. Right click on web pages and select new and select jsp as shown below:



15] give name and click on finish as shown below do it 2 times on for input and one for output:

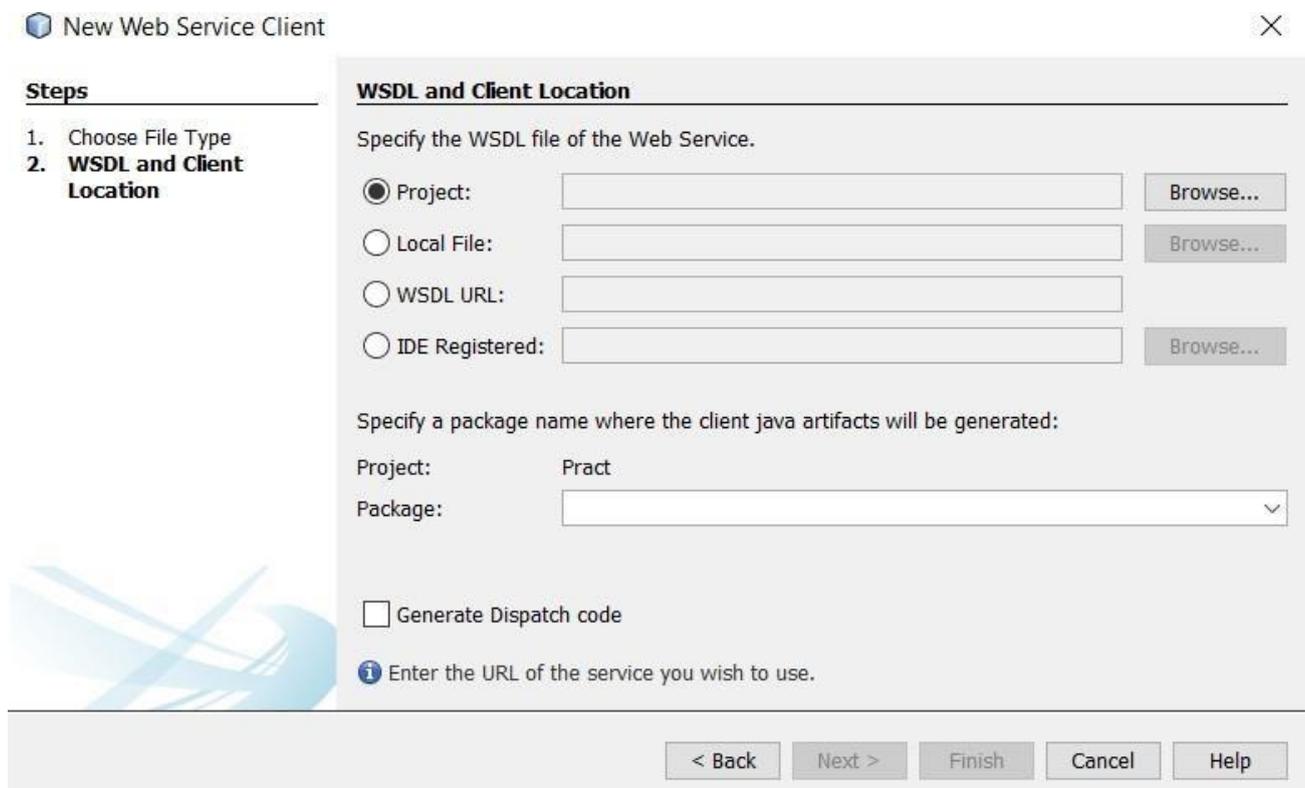


16] In input.jsp create a form for taking user input as shown below:

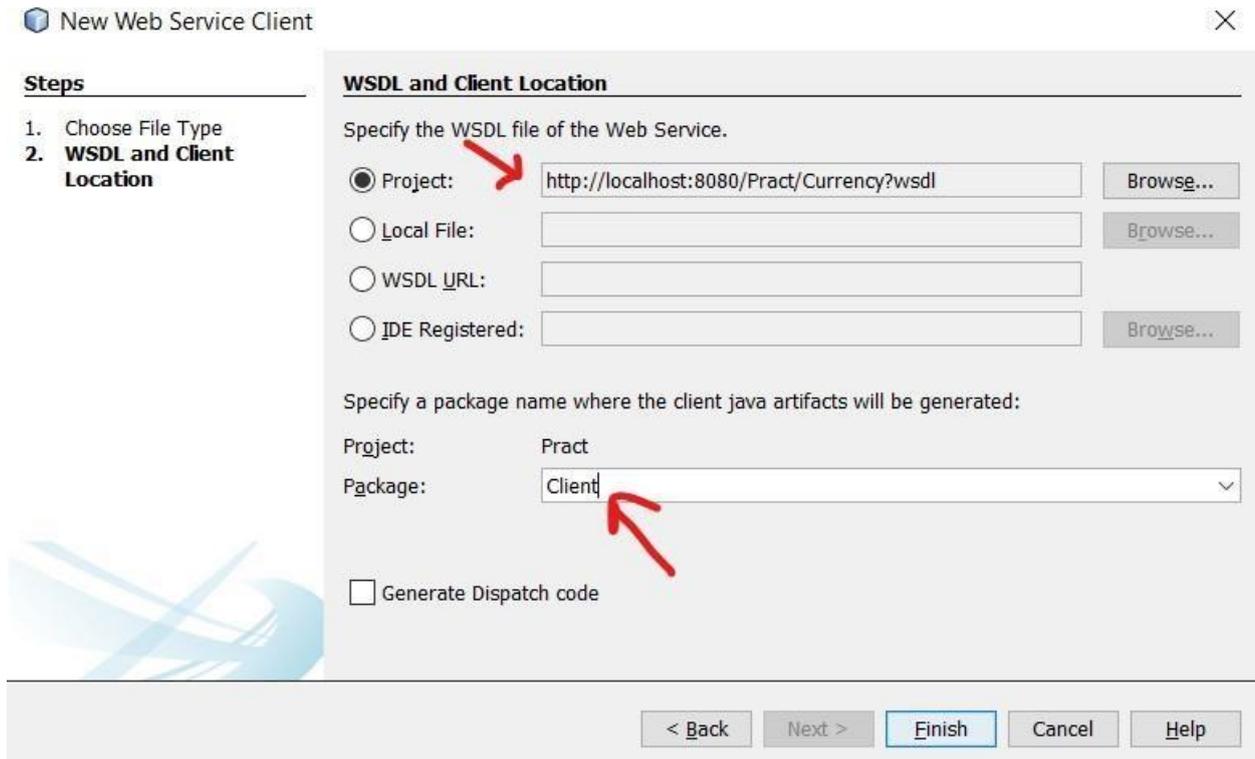
```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <form action="output.jsp">
      <pre>
        Enter the currency in rupees : <input type="text" name="t1">
        <input type="submit"> <input type="reset">
      </pre>
    </form>
  </body>
</html>
```

In this code set action = the jsp file where u want output and give name to textbox input.

17] now we have 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:



At this step click on browse and select your project and click ok after clicking ok you will get wsdl url as shown below:



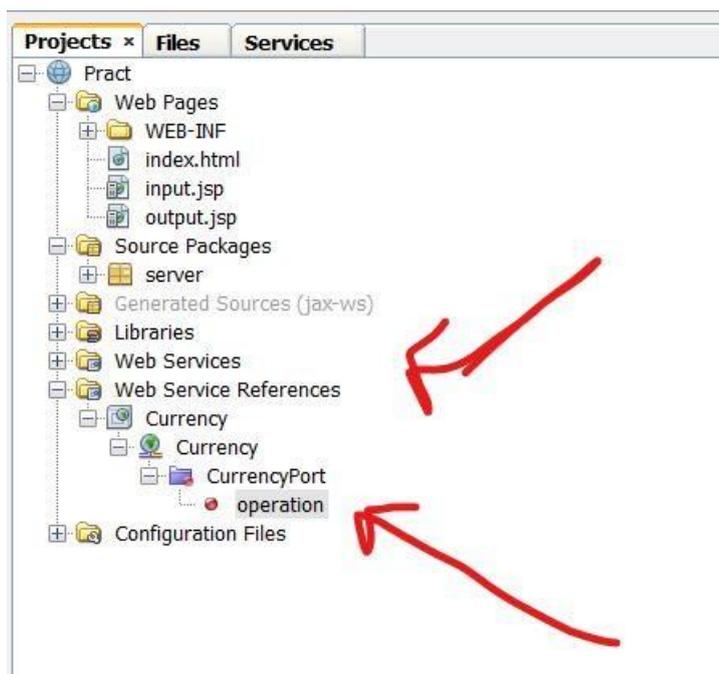
**At this stage copy the url and paste it in notepad for future.**

**Give package name as “Client”.**

**Now click on finish.**

18] now in project section you can see a new folder is been created named “web servicereference”.

double click to expand it you should get the following :



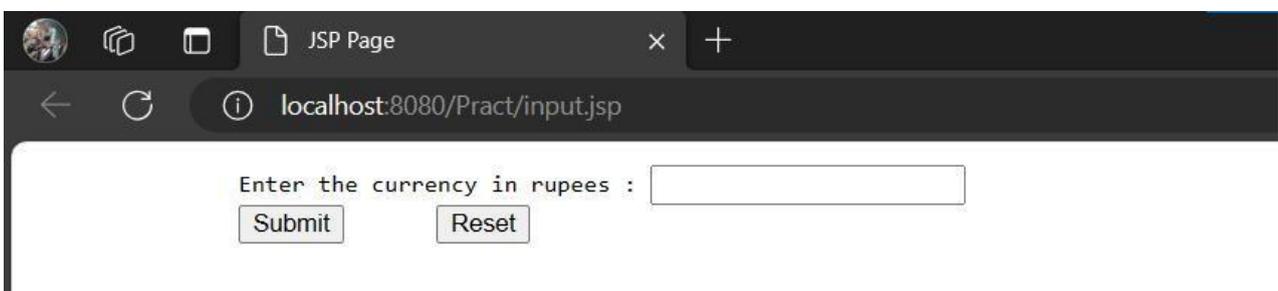
Hold the operation or your operation name and drag it into the output.jsp in body tag as shown : You will get the following auto generated code:

```
<body>
  <!-- start web service invocation --><hr/>
  <%
  try {
    Client.Currency_Service service = new Client.Currency_Service();
    Client.Currency port = service.getCurrencyPort();
    // TODO initialize WS operation arguments here
    double a = 0.0d;
    // TODO process result here
    java.lang.String result = port.operation(a);
    out.println("Result = "+result);
  } catch (Exception ex) {
    // TODO handle custom exceptions here
  }
  %>
  <!-- end web service invocation --><hr/>
</body>
```

19] now make changes as shown below in the code:

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

20] now Deploy our project again and right click on input.jsp and click run file you will redirect to a browser page as shown :



I

Enter any numerical value in text box and click on submit you will get the following output:



---

## Python client:

---

1] for consuming java web services in python we should repeat the above steps till step no.19 .

**Note: for getting output the project or web service should be deployed .**

After doing all of the steps write the following code in python idle:

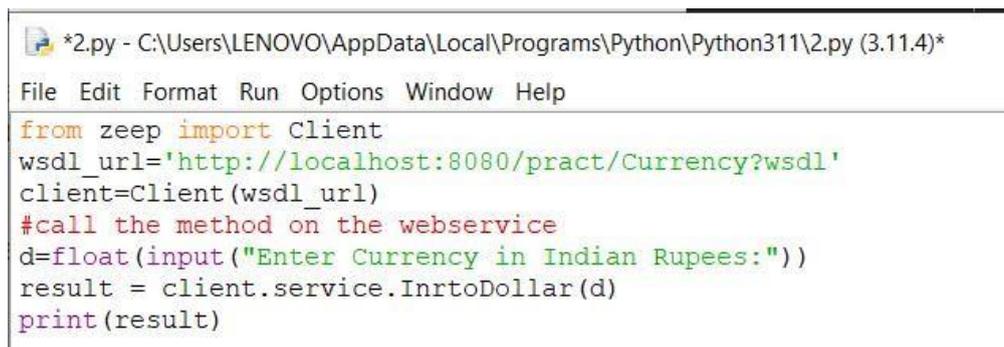
### Code :

```
from zeep import Client
wsdl_url='http://localhost:8080/pract/Currency?wsdl'
client=Client(wsdl_url)
#call the method on the webservice
d=float(input("Enter Currency in Indian Rupees:"))
result = client.service.InrtoDollar(d)
```

print(result)2] replace wsdl\_url with the url which was copied at the time of web service client creation .

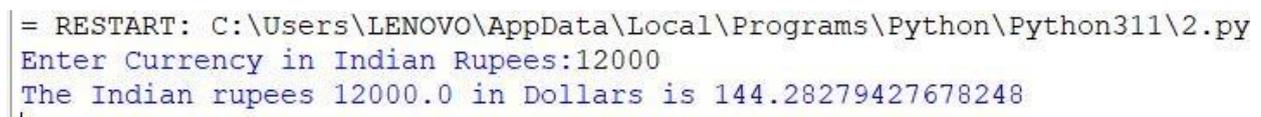
In step no.17 . and replace the InrtoDollar with your method or operation name. and pass parameter to it

3] final code should look like following:



```
*2.py - C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\2.py (3.11.4)*
File Edit Format Run Options Window Help
from zeep import Client
wsdl_url='http://localhost:8080/pract/Currency?wsdl'
client=Client(wsdl_url)
#call the method on the webservice
d=float(input("Enter Currency in Indian Rupees:"))
result = client.service.InrtoDollar(d)
print(result)
```

4] after running the above code you should get the following output:



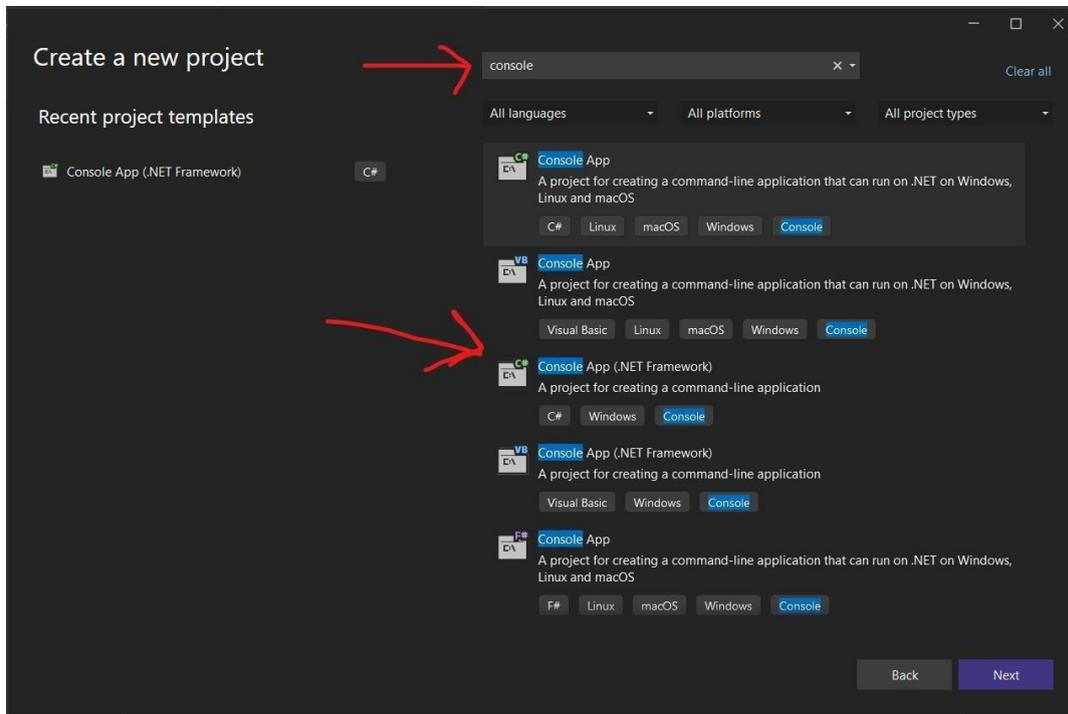
```
= RESTART: C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\2.py
Enter Currency in Indian Rupees:12000
The Indian rupees 12000.0 in Dollars is 144.28279427678248
|
```

---

## .Net Client

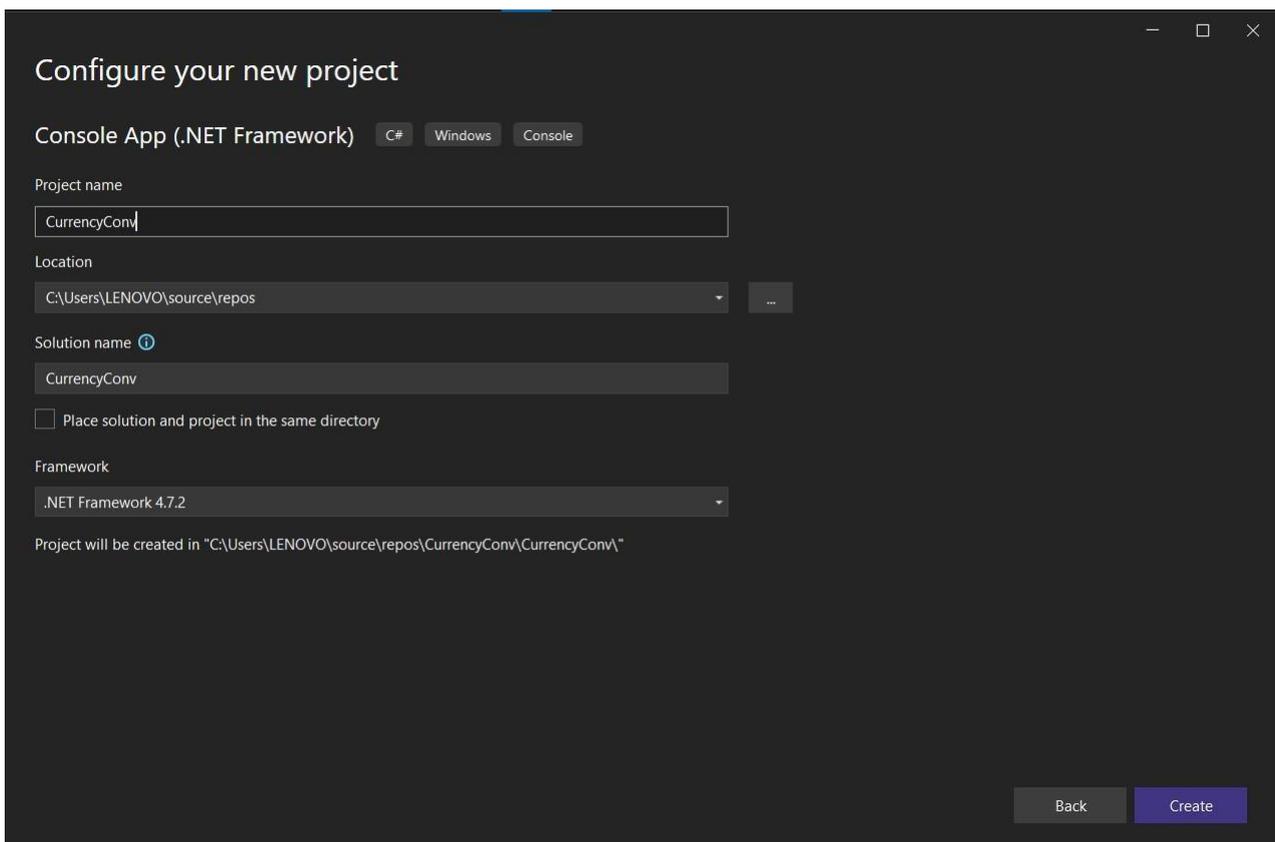
---

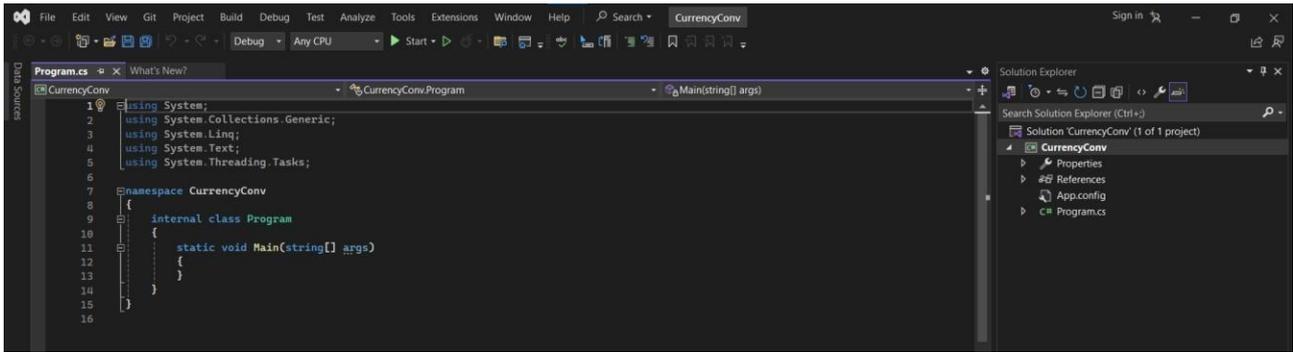
1] open visual studio and select create a new project the following window will appear:



Here in search bar type “console c#” and select console app for .Net frame work as shown in above image.

2] Give name to the project and press create button as shown in the image:

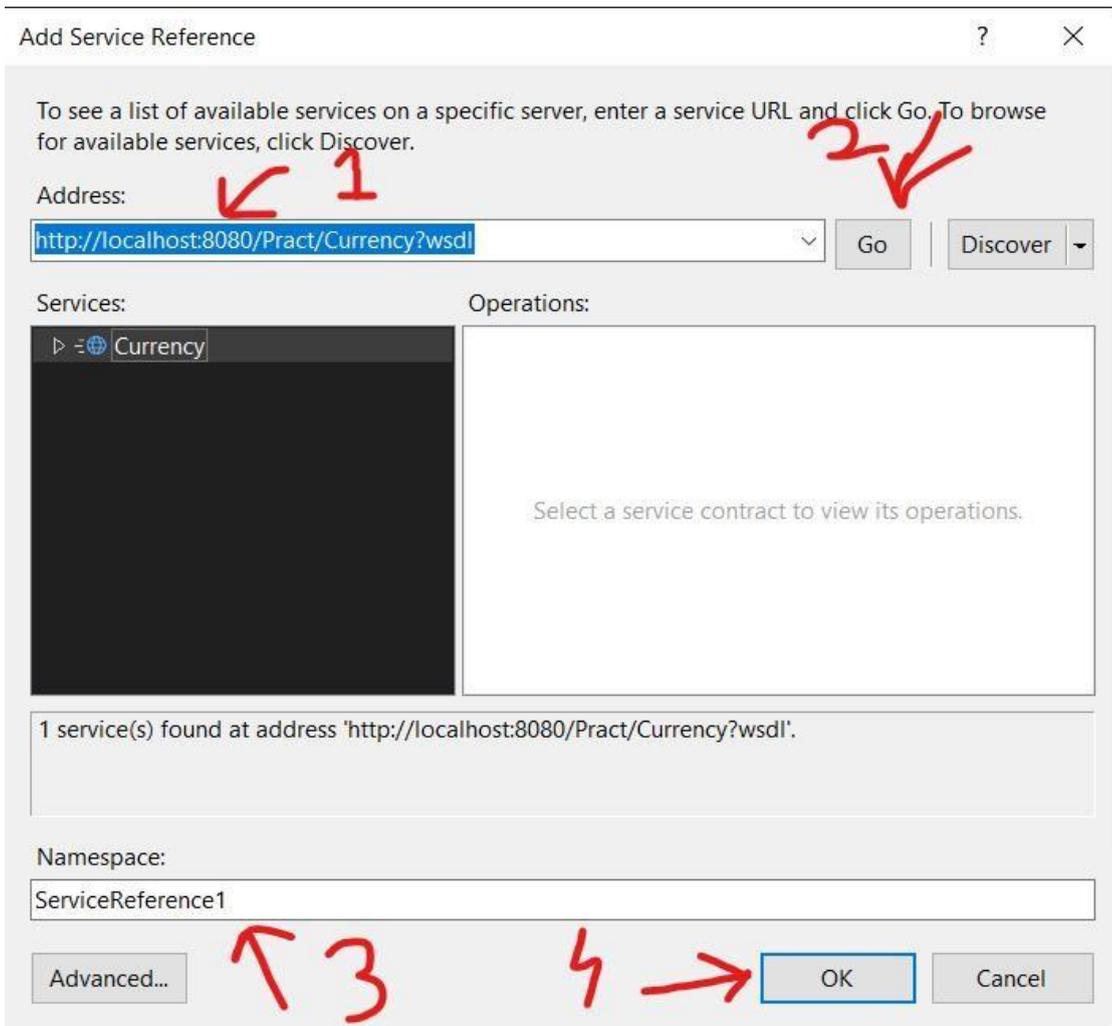




This window should appear after clicking create button now In right side there is solution Explorer.

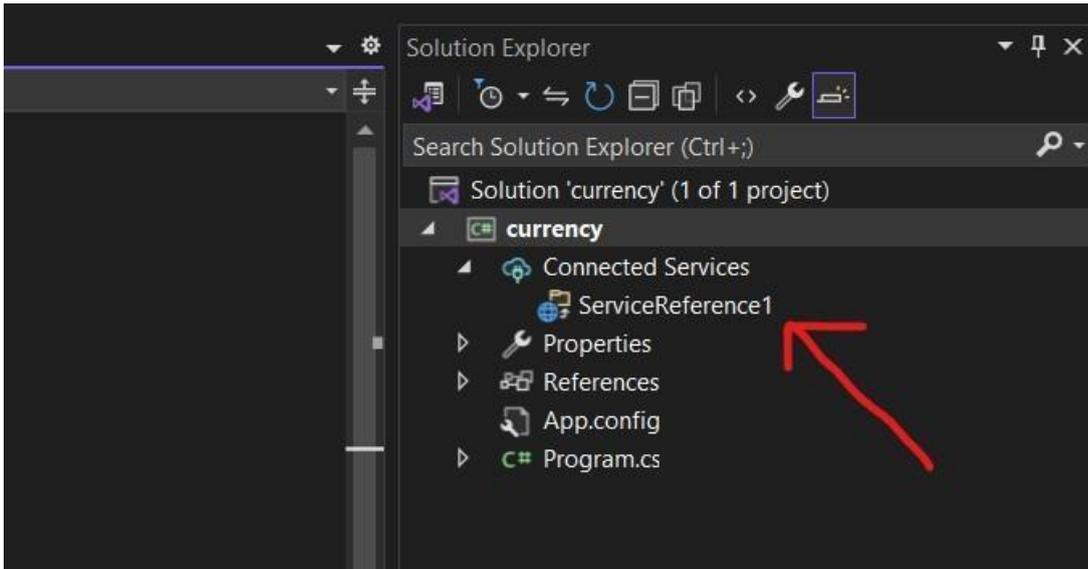
In that right click on your project name and click on add > Service Reference . You will get the following window :

***Note: before executing the code make sure that the web service is deployed.***



Here in address bar paste the wsdl url the same which is used I python code. And press go button . agter set the namespace In my case I am letting is as it is then press ok button.

In following image u can check in solution Explorer service refrence iscreated.:



3] now in code write the following line as shown in image:

```
CurrencyConv CurrencyConv.Program
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using CurrencyConv.ServiceReference1;
7
8  namespace CurrencyConv
9  {
10     internal class Program
11     {
12         static void Main(string[] args)
13         {
14         }
15     }
16 }
17
```

Here type:

Using your\_projectname . your\_serviceReference\_name ;

4] now write the code as shown in the following image:

```

namespace CurrencyConv
{
    internal class Program
    {
        static void Main(string[] args)
        {
            CurrencyClient client = new CurrencyClient();
            Console.WriteLine("Enter the Currency in Indian Rupees : ");
            double d = double.Parse(Console.ReadLine());
            Console.WriteLine(client.InrtoDollar(d));
            Console.WriteLine("Enter any key to Exit...");
            Console.ReadKey();
            Console.ReadLine();
        }
    }
}

```

**Note:**

Here **CurrencyClient** is my webservice class name and **operation()** is my method name as show you can also check your own:



5] when you run the above code you get the following output:

```

C:\Windows\system32\cmd.exe
CurrencyConv.Progra
Enter the Currency in indian rupees:
1000
The Indian rupees 100.0 in Dollars is 1.2023566189731874
Enter any key to Exit...

```

## Practical-2

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

### Steps:

**Note : for all these steps in brief refer practical no.1 document**

- 1] create a new project and create a new webservice
- 2] now right click on the code and click on insert code > add new web service operation
- 3] after give name to operation **FtoC** and add one parameter **a** of double data type and click ok the code will auto generate .
- 4] now do the following changes in that auto generated code:

```
*/
@WebMethod(operationName = "FtoC")
public String FtoC(@WebParam(name = "a") double a) {
    //TODO write your implementation code here:
    return "The Fahrenheit Temperature "+a+" in Celsius is "+((a-32)*5/9);
}
```

Now deploy the model.

- 5] After deploying the model just right click on your web service and click on test web service . and you will get redirected to a browser page where you can check that your web service is working or not.

---

Creating java Client using jsp

---

- 5] now right click on web pages > new > jsp . and give name as input jsp.  
Again right click on web pages > new > jsp . and give name as output jsp.
- 6] In input.jsp write the following code :

4

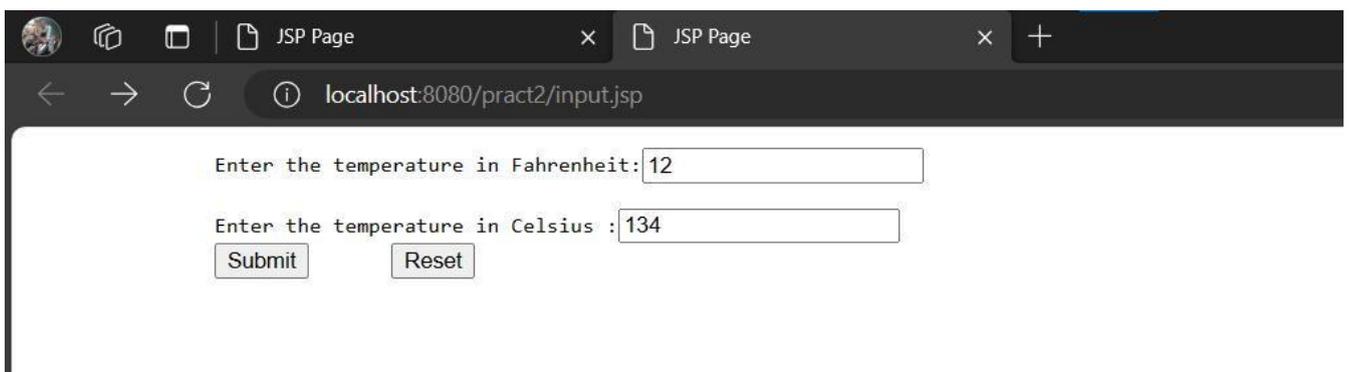
```
<body>
  <form action="output.jsp">
    <pre>
      Enter the temperature in Fahrenheit:<input type="text" name="t1" required>

      Enter the temperature in Celsius :<input type="text" name="t2" required>
      <input type="submit">      <input type="reset">
    </pre>
  </form>
</body>
```

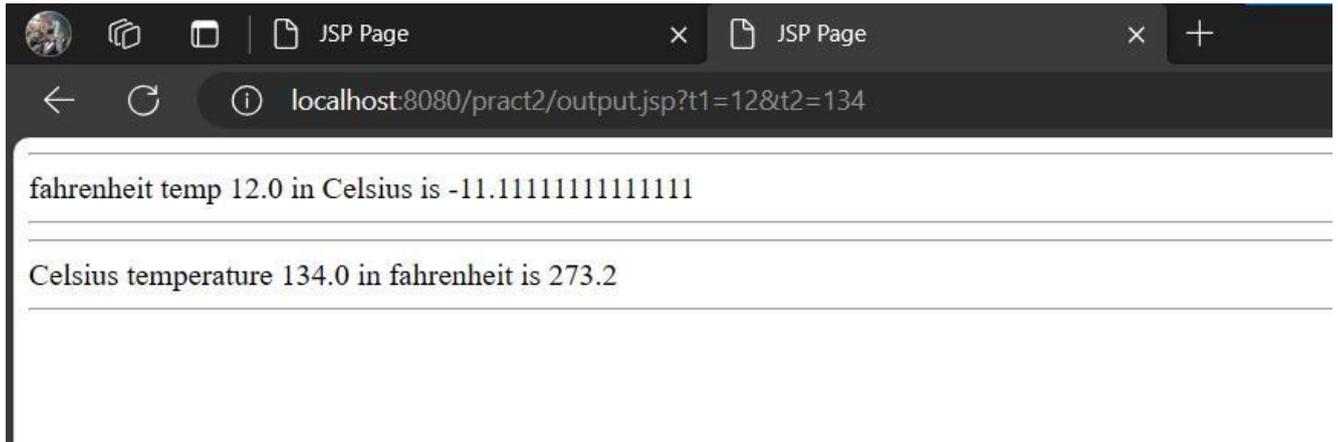
7] create a Web service client and give package name as "Client". After creating web service client go to web service references and drag and drop the operations in output.jsp in body tag as shown below:

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

8] now deploy the model again and run the input.jsp file you will get redirected to the browser page as shown:



After submitting you will get the following output:



---

## Python Client

---

1] Deploy the web service and write the following code :

Code :

```
from zeep import Client
wsdl_url='http://localhost:8080/Practical_2/TempConverter?wsdl'
client=Client(wsdl_url)
#call the method on the webservice
d=float(input("Enter temperaure in Fahrenheit :"))
result = client.service.FtoC(d)
print(result)
```

2] *replace your\_wsdl\_url with the url which was copied at the time of web service client creation .*

**In step no.16 . and replace the operation\_name with your method or operation name. and pass parameter to it**

3] *final code should look like following:*

```
*2.py - C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\2.py (3.11.4)*
File Edit Format Run Options Window Help
from zeep import Client
wsdl_url='http://localhost:8080/Practical_2/TempConverter?wsdl'
client=Client(wsdl_url)
#call the method on the webservice
d=float(input("Enter temperaure in Fahrenheit :"))
result = client.service.FtoC(d)
print(result)
```

4] after running the above code you should get the following output:

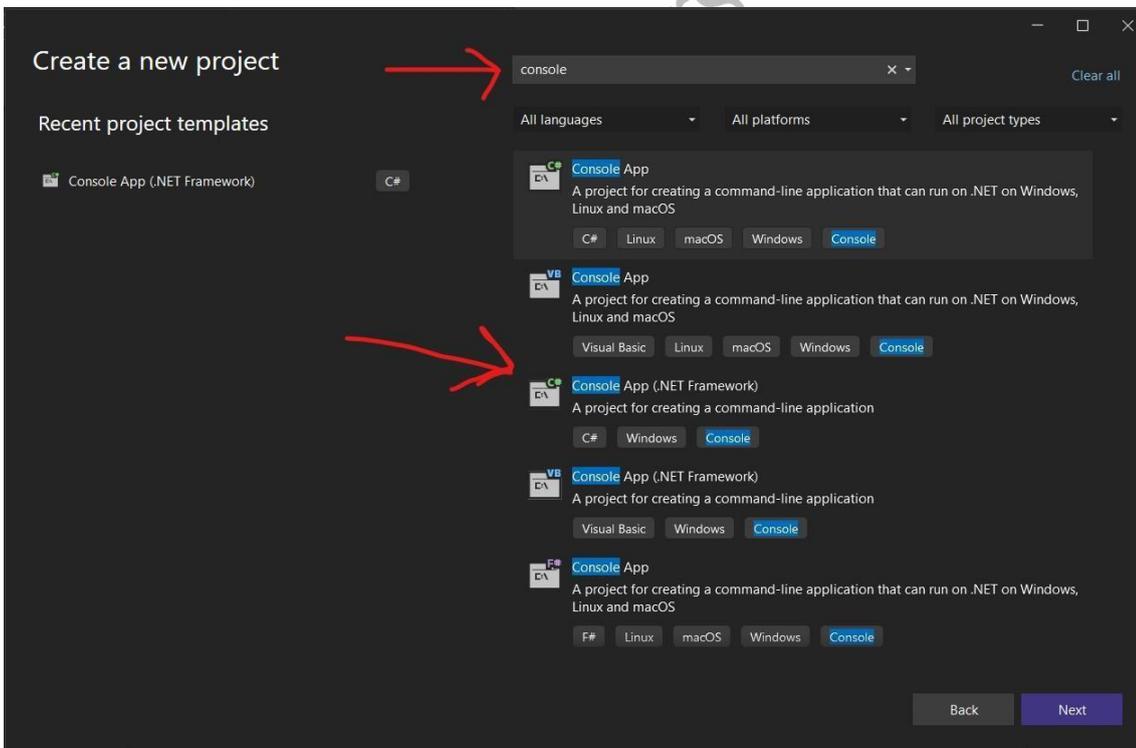
```
==== RESTART: C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\2
Enter temperaure in Fahrenheit :1200
The Fahrenheit Temperature 1200.0 in Celsius is 648.8888888888889
```

---

## .Net Client

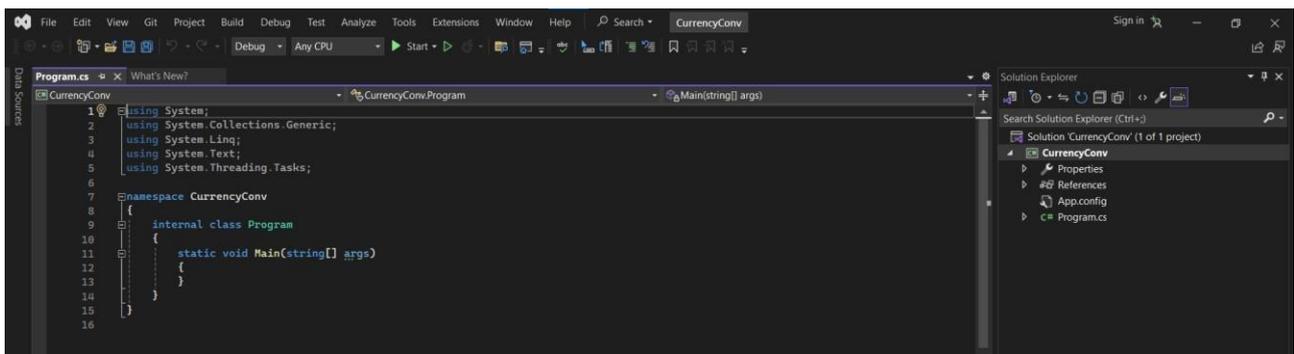
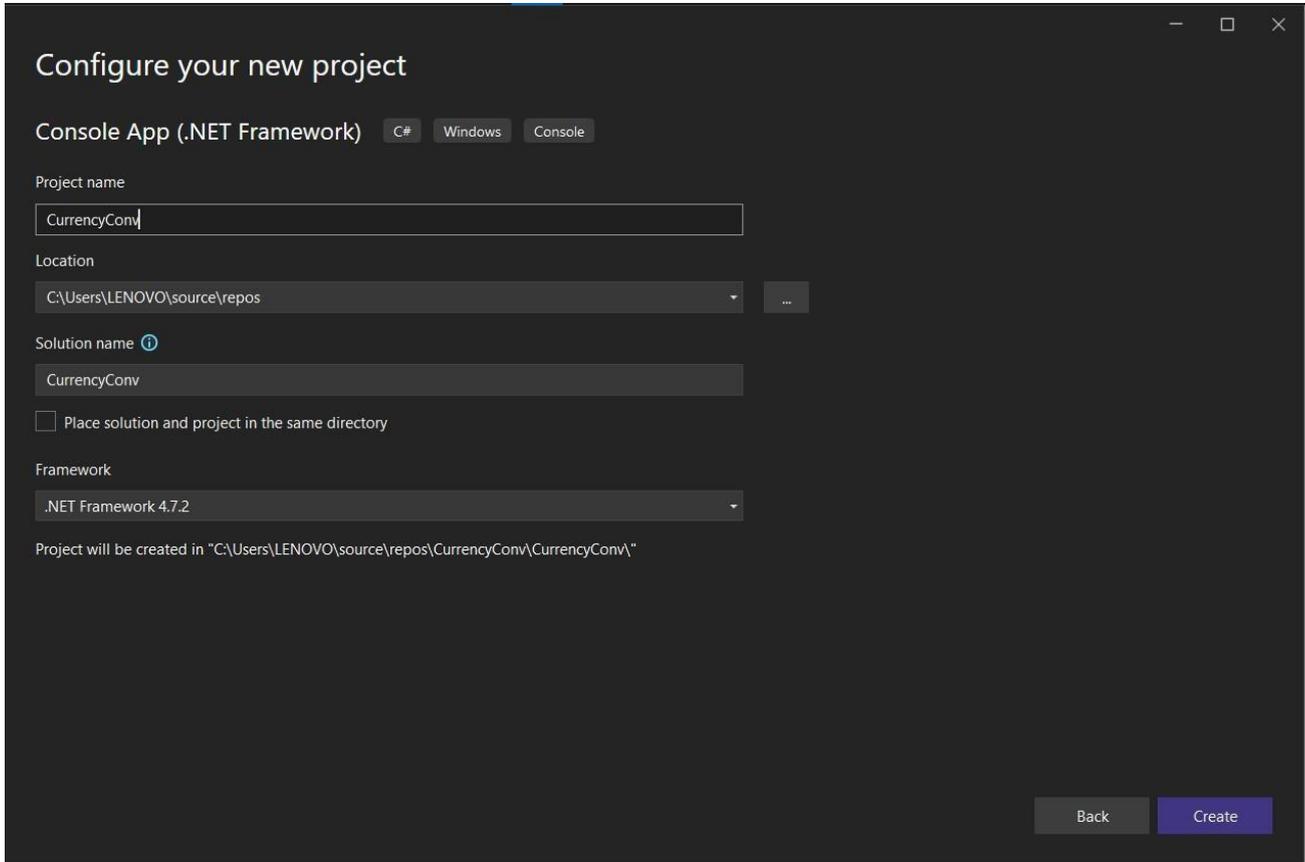
---

1] open visual studio and select create a new project the following window will appear:



Here in search bar type “console c#” and select console app for .Net frame work as shown in above image.

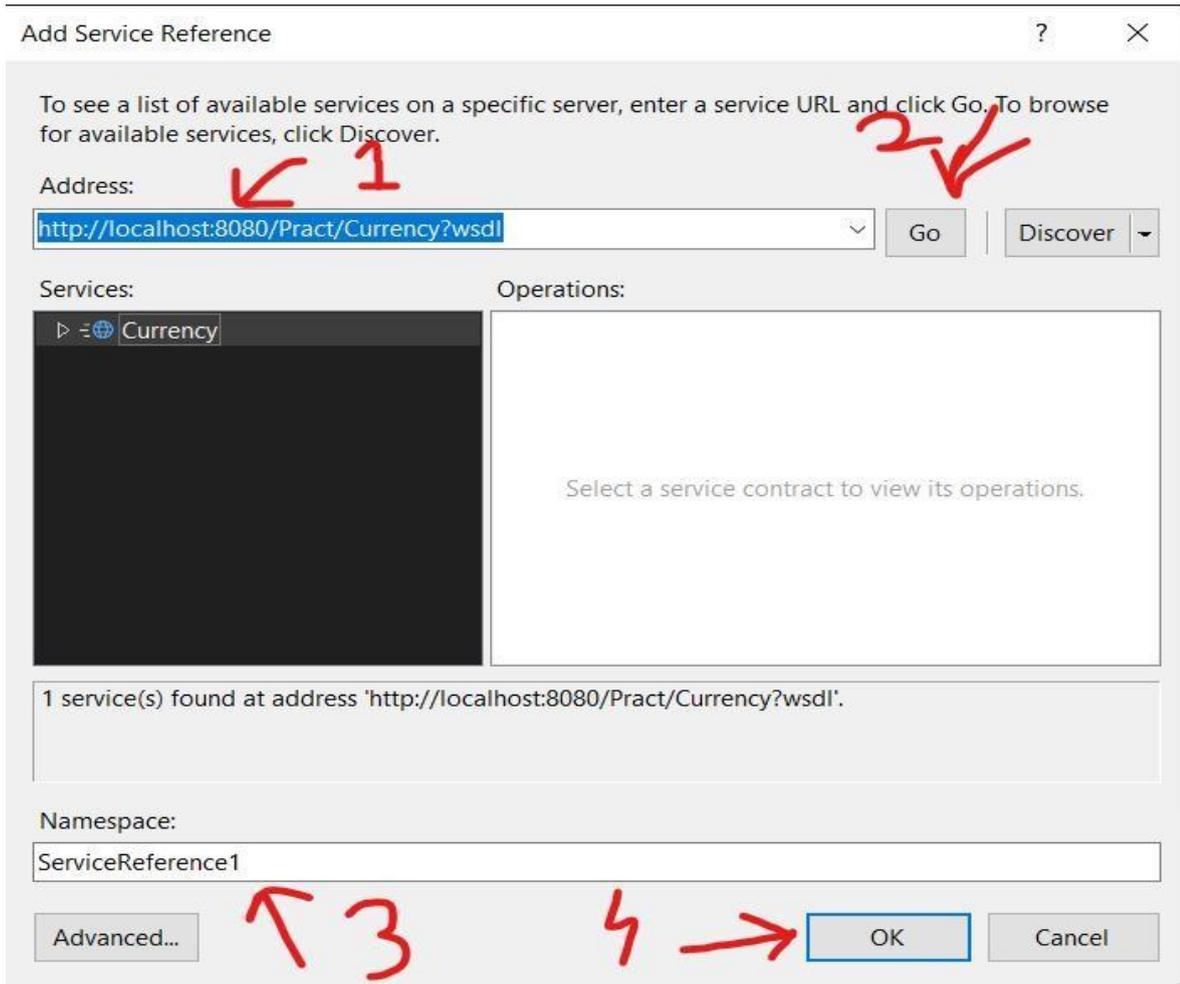
2] Give name to the project and press create button as shown in the image:



This window should appear after clicking create button now In right side there is solution Explorer.

In that right click on your project name and click on add > Service Reference . You will get the following window :

**Note: before executing the code make sure that the web service is deployed.**



Here in address bar paste the wsdl url the same which is used I python code. And press go button . agter set the namespace In my case I am letting is as it is then press ok button

3] now in code write the following line as shown in image:

```
C# CurrencyConv CurrencyConv.Program
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using CurrencyConv.ServiceReference1;
7
8 namespace CurrencyConv
9 {
10     internal class Program
11     {
12         static void Main(string[] args)
13         {
14         }
15     }
16 }
17
```

Here type:

Using your\_projectname . your\_serviceReference\_name ;

4] now write the code as shown in the following image:

```
static void Main(string[] args)
{
    convClient client = new convClient();
    Console.WriteLine("Enter the Temperature: ");
    double d = double.Parse(Console.ReadLine());
    Console.WriteLine(client.FtoC(d));
    Console.WriteLine("Enter any key to Exit...");
    Console.ReadKey();
    Console.ReadLine();
}
```

**Note:**

Here my webservice class name is conv and convClient is my webservice client class name and FtoC is method name as shown you can also check your own:

5] when you run the above code you get the following output:

```
C:\Windows\system32\cmd.exe
Enter the Temperature in Fahrenheit :
1200
The Fahrenheit Temperature 1200.0 in Celsius is 648.8888888888889
```

## Practical-03A

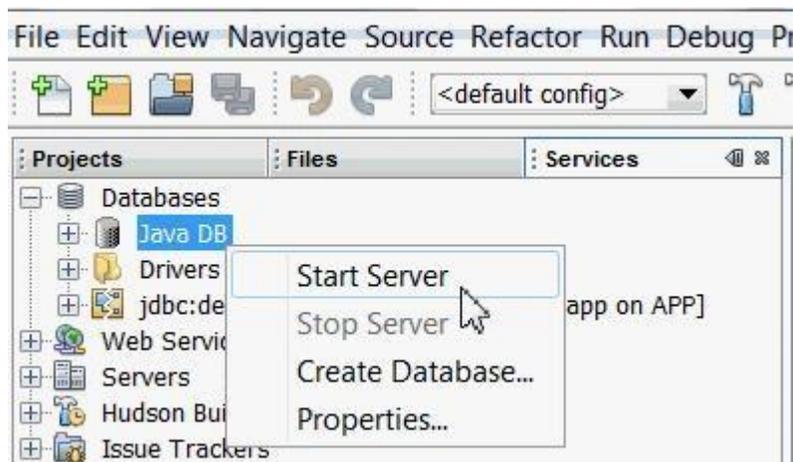
Create a Simple REST Service

- Demonstrate CRUD operations with suitable database using RESTful Web service.

1. To start the Java DB Database from NetBeans, perform the following steps.

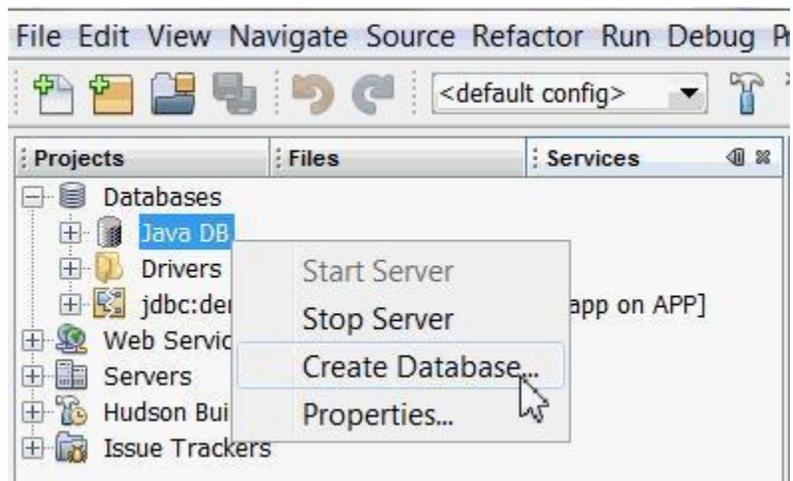
Click Services tab -> Expand Databases node. ->Right-click Java DB icon.

->Select Start Server



2. To create playerDB database, perform the following steps:

- a. Right-click **Java DB** icon, select **Create Database**.



b. Enter the following information for the database:

Database Name: **playerDB**

User Name: **john**

Password: **john**

Confirm Password: **john**

c. Click OK.

3. To connect to the newly created database playerDB, perform the following steps

a. Right-click jdbc:derby://localhost:1527/playerDB connection.

b. Select Connect.

4. Create tables and populate them with data in **playerDB** database.

a. In NetBeans select File > Open File.

b. In the file select playersDB.sql.

c. Click Open. The script automatically opens in the SQL Editor.

5. Examine the contents of the database.

a. In the Services window, expand the jdbc:derby://localhost:1527/playerDB connection under the Databases node.

b. Right-click the connection and select Refresh.

c. Expand the john schema. You see the nodes for Tables, Views, and Procedures. d. Expand the Tables node to see the PLAYER and TEAM tables.

e. Right-click the PLAYER table node and select View Data.

## Player server

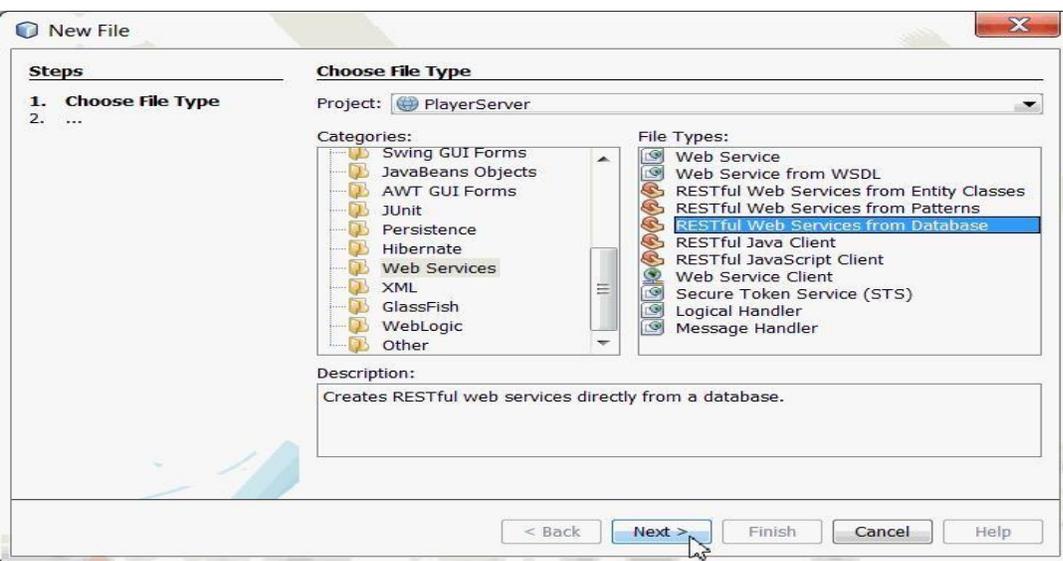
To create RESTful Web Services, you need a Java Web application project. In the below section you will create a demo Java web project, PlayerServer.

To create new Java Web Project, select File -> New Project -> Java web -> Web Application.

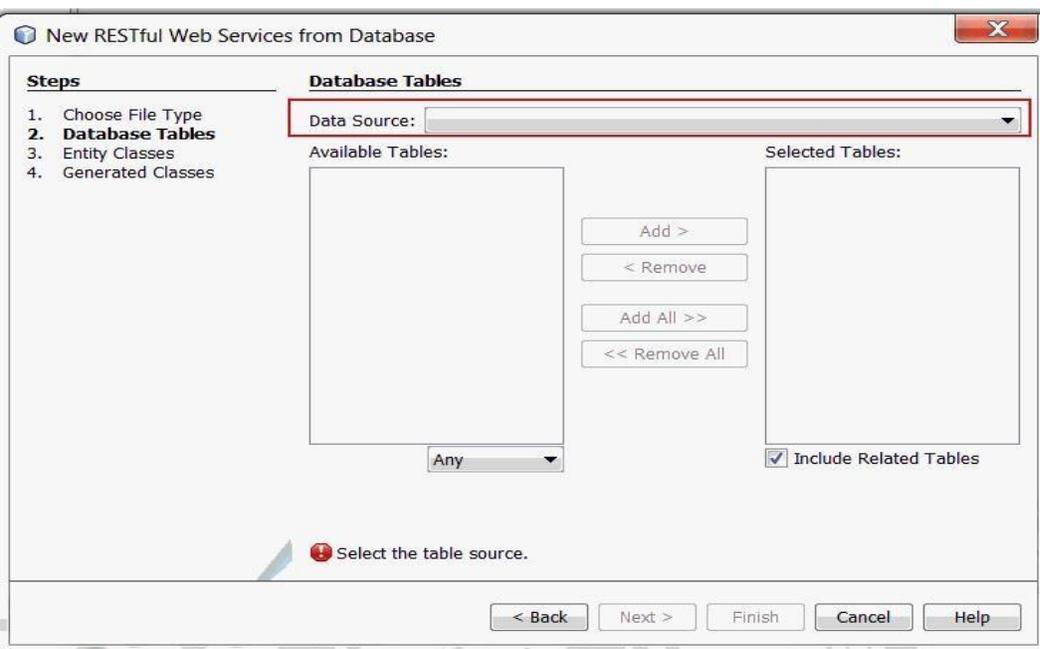
## Generating web service

To generate RESTful Web Services do the following:

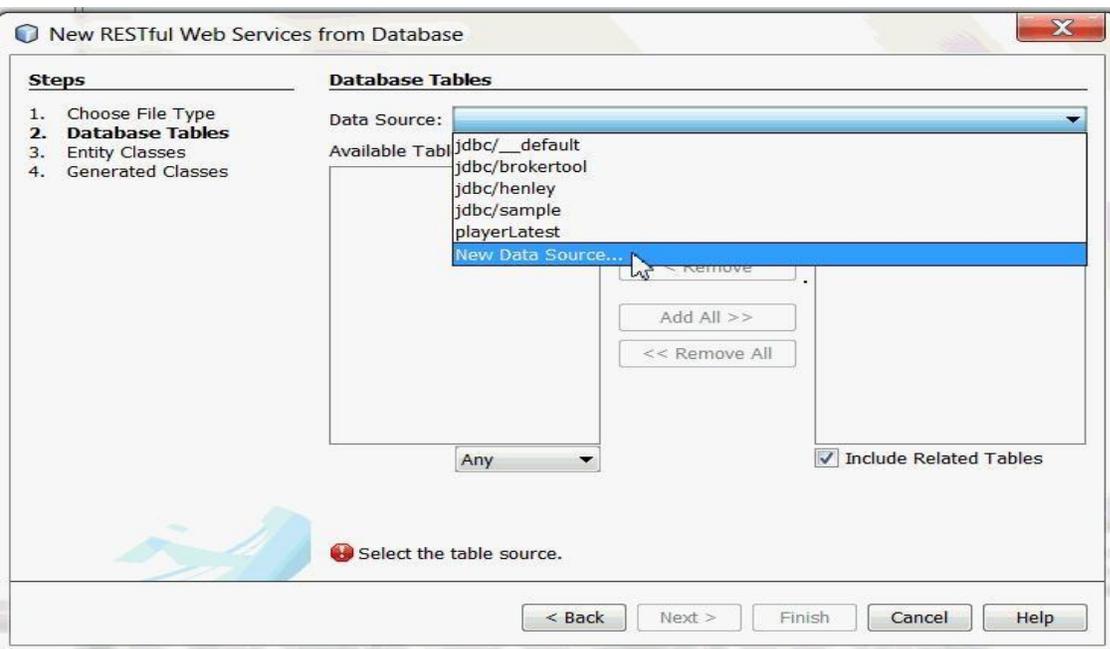
1. Right-click the PlayerServer and choose New > Other > Web Services > RESTful Web Services from Database. The New RESTful Web Service wizard opens on the Database Tables panel.



2. In the Database Tables window, select Data Source.



3. Next select "New Data Source" from the Drop-down list.



a. In the Create Data Source Window, enter the following information:

- o **JNDI name:** jdbc/playerDB
- o **Database connection :** select jdbc:derby://localhost:1527/playerDB[john on JOHN]

b. Click OK.



The PLAYER and TEAM tables are displayed under the Available Tables column.

c. Click **Add All**. The PLAYER and TEAM tables are displayed under the Selected Tables column.



d. Click Next.

4. In the **Entity Classes** window, complete the following steps:

- a. Enter the package name as com.playerentity.
- b. Click Next.

5. In the **Generated Classes** Window, click Finish with default values.

Test RESTful web Services

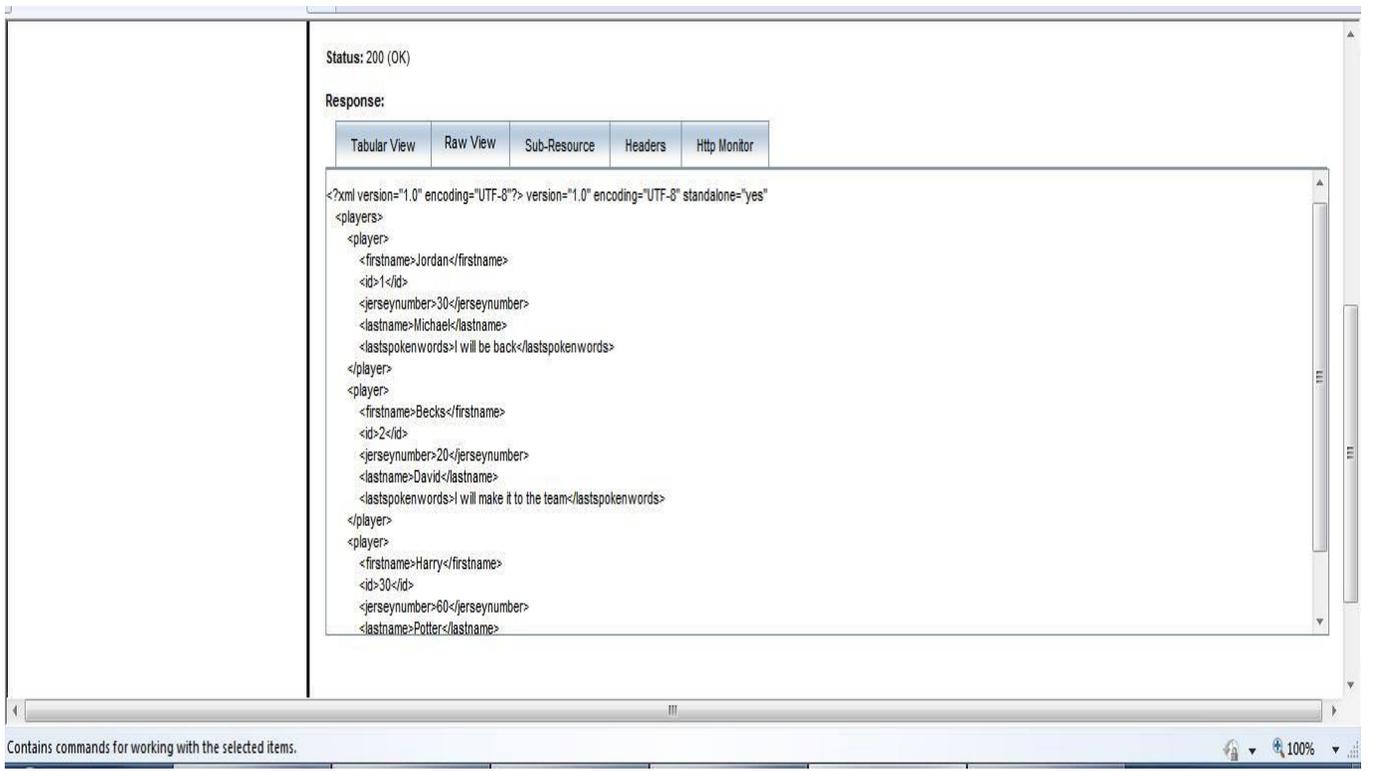
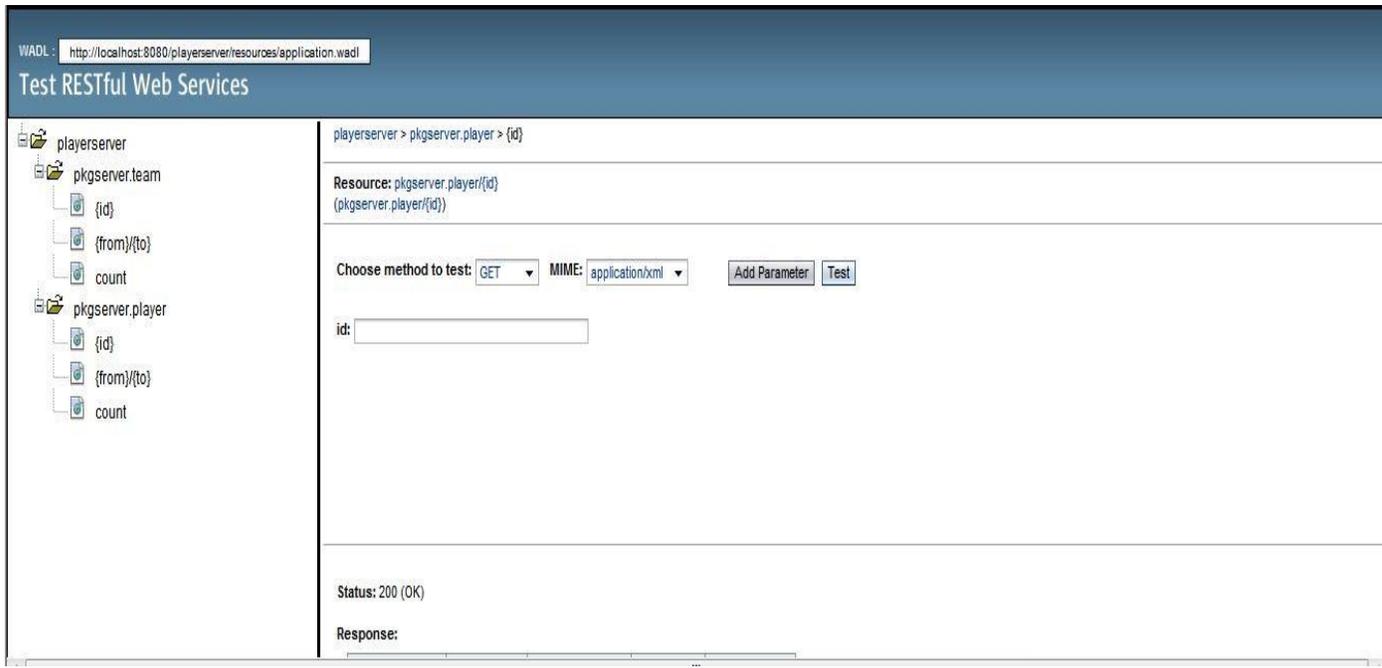
1. To Generate Web Services Test client, complete the following steps.

- a. Select **PlayerServer** project.
- b. Right-click and select **Test RESTful Web Services**.

2. Select "Web Test Client in project" and click Browse.

3. a. In the Select Project dialog box, select **PlayerServer** and click OK.

b. **Configure Rest Test Client window** is displayed, click OK.



## Player Client

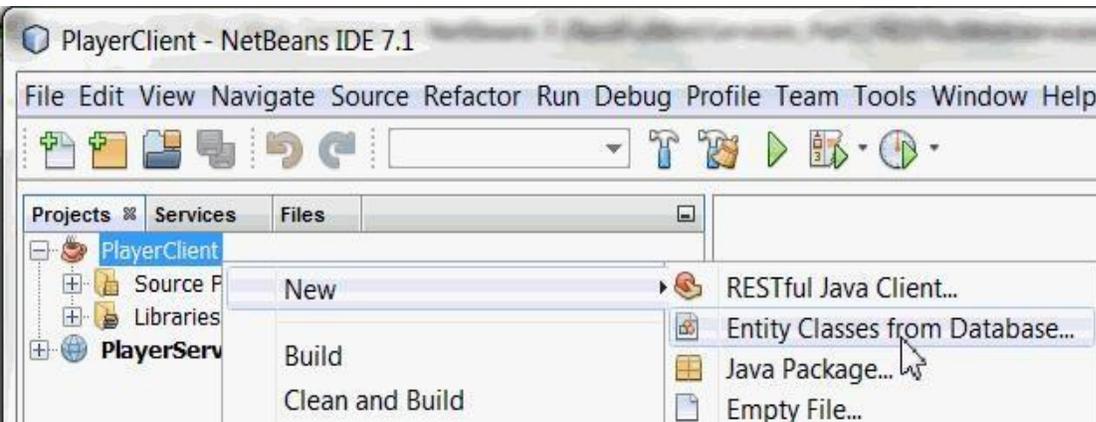
To create new Java Project, select File > New Project > Playerclient

### ***Generating Entity Classes from Database***

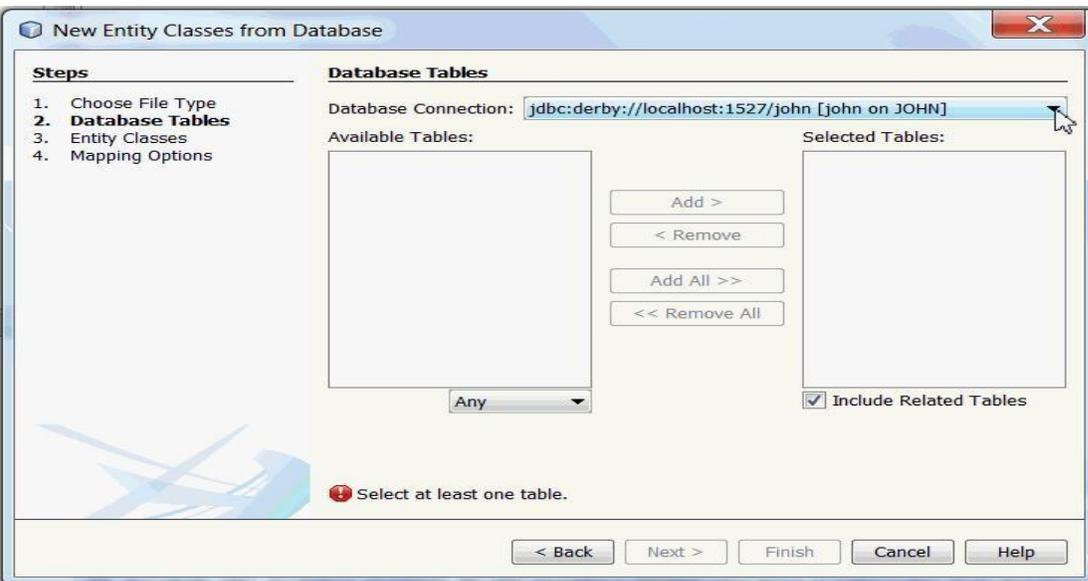
Implementing CRUD operations on the database, requires creation of Entity Classes to communicate with the database. The following section demonstrates how to create Entity

Classes from database.

1. Right-click **PlayerClient** Project and select **New > Entity Classes From Database**.

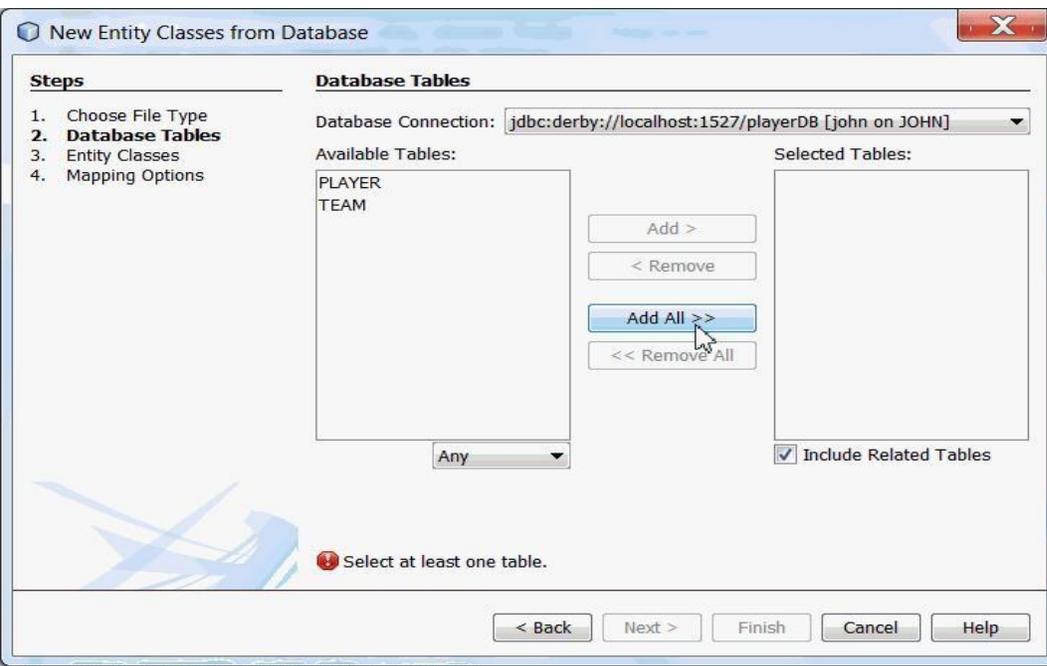


2. a. In the Database Tables window, **Database Connection** field select jdbc:derby://localhost:1527/playerDB[john on JOHN] from the drop-down list.

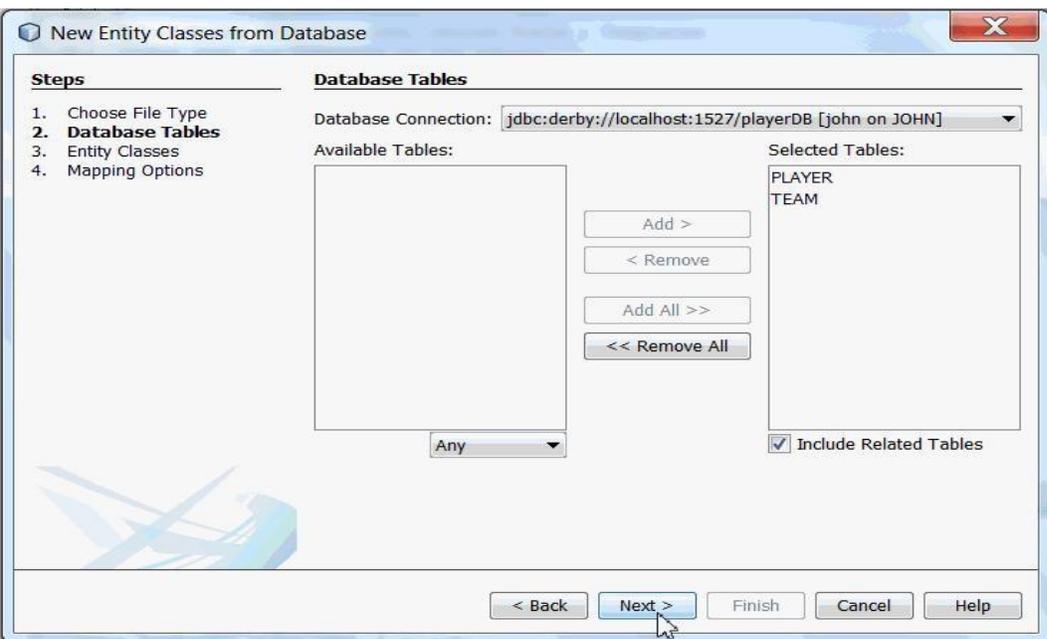


You see PLAYER and TEAM tables in Available Tables category

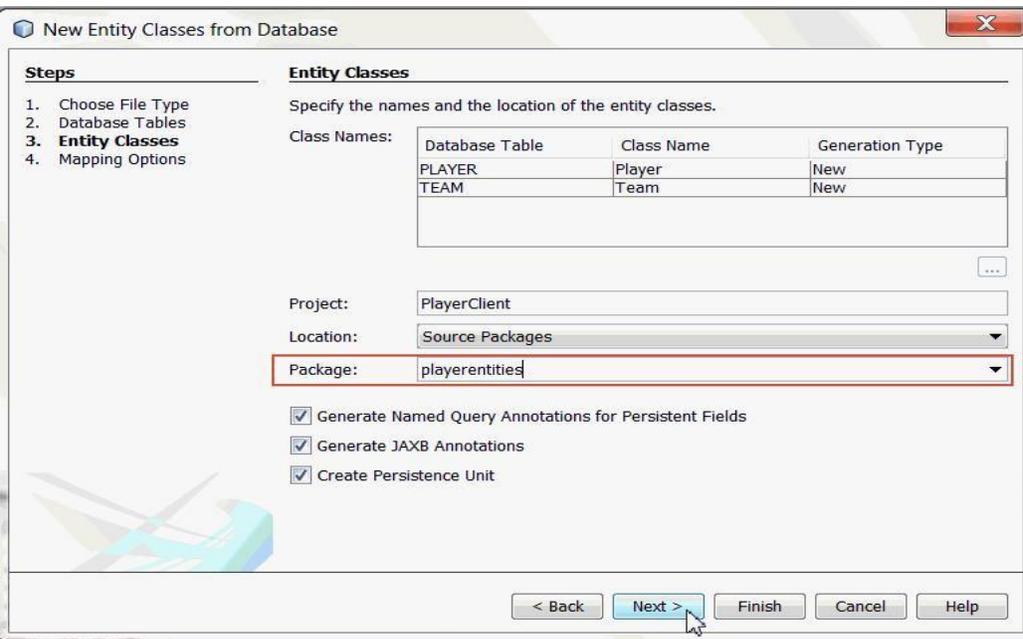
- b. Click Add All



You see both the tables PLAYER and TEAM in Selected Tables Category.  
c. Click Next



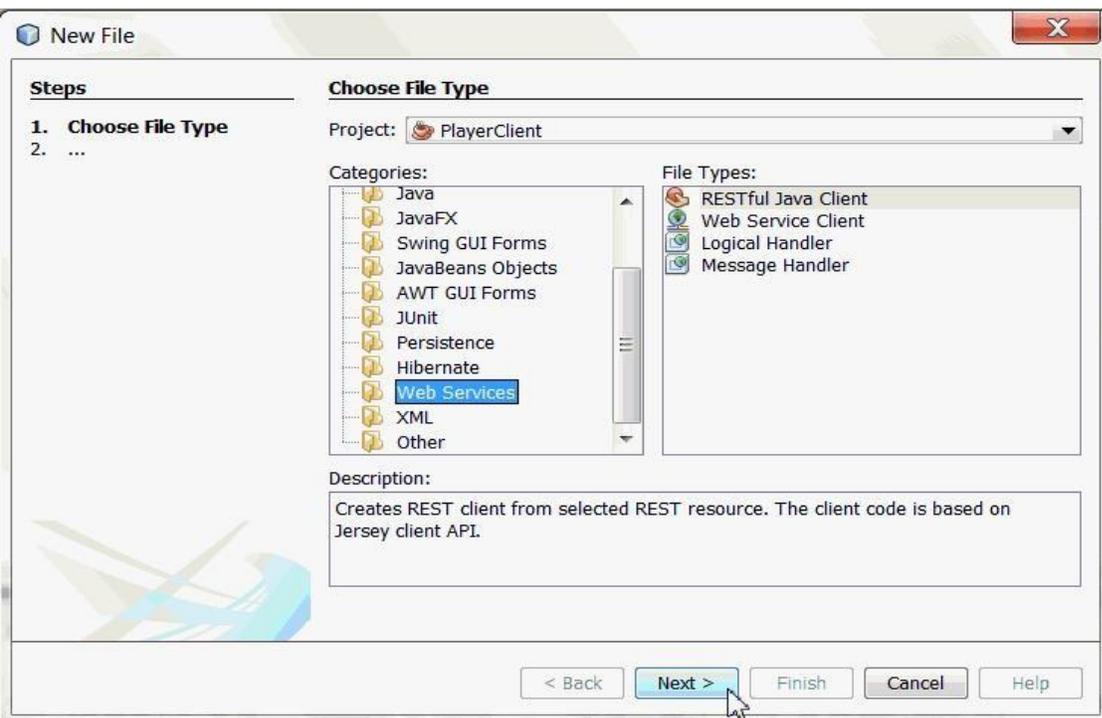
3. In the Entity classes Window, enter the Package Name as **playerentities** and click Next.



## Create Client

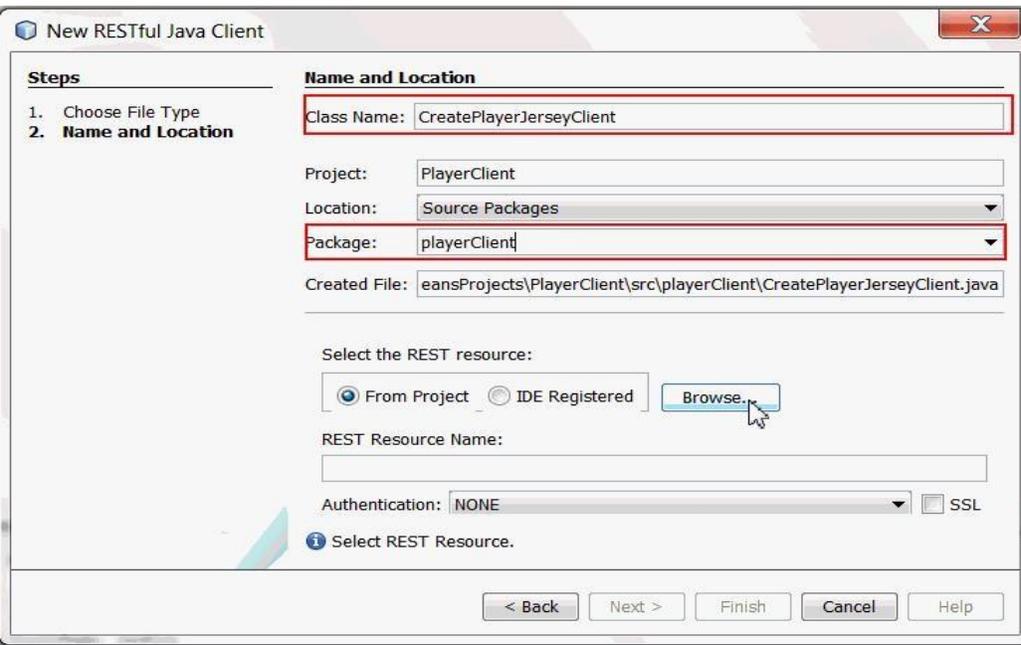
Complete the following steps to create RESTful client in PlayerClient project.

- a. Right-click the PlayerClient and choose New > Other > Web Services > RESTful Java Client.
- b. Click Next.

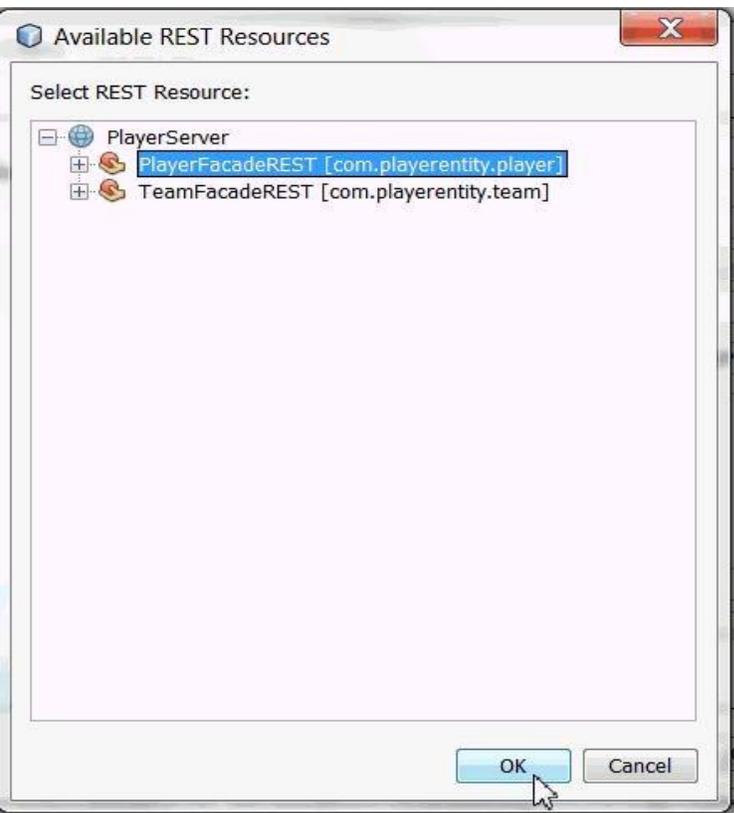


1. a. Enter the following details for the fields in the New Restful Java Client window :

- Class Name: CreatePlayerJerseyClient
- Package: playerClient
- Select the REST resource: From Project and Click Browse.



- b. Select PlayerServer.
- c. Expand the PlayerServer application.
- d. Select PlayerFacadeREST[com.playerentity.player].



2. Add the main method.

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

```
    CreateClient client1 = new CreateClient();
```

```
    ClientResponse response = client1.findAll_XML(ClientResponse.class);
```

```
    GenericType<List<Player>> genericType = new GenericType<List<Player>>({});
```

```
    List<Player> data = new ArrayList<Player>();
```

```
    data = (response.getEntity(genericType));
```

```
    Player p = new Player();
```

```
    p.setFirstname("Harry");
```

```
    p.setId(30);
```

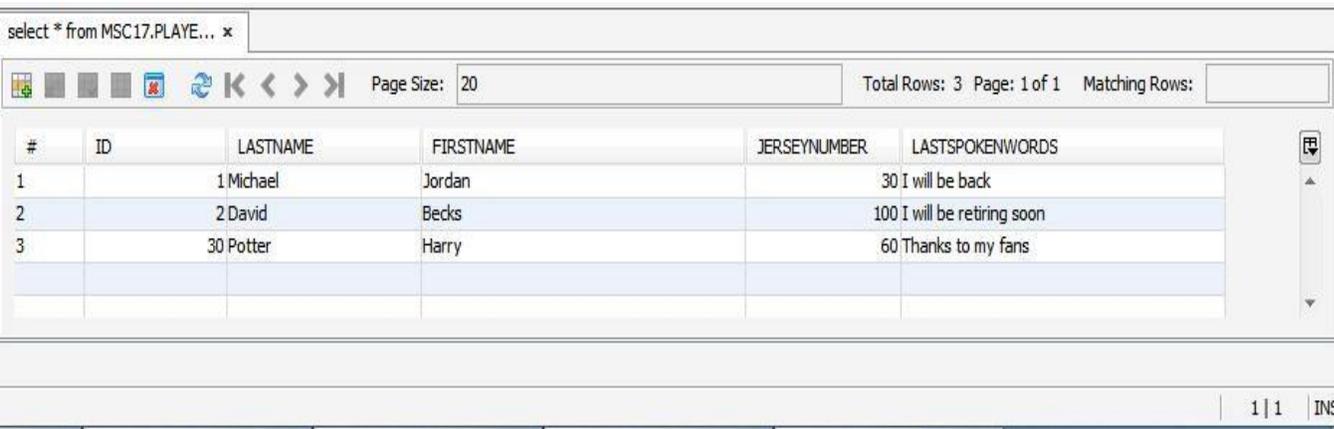
```
    p.setJerseynumber(60);
```

```
    p.setLastname("Potter");
```

```
    p.setLastspokenwords("Thanks to my fans");
```

```
client1.create_XML(p);
```

```
}
```



The screenshot shows a database query result window with the following data:

#	ID	LASTNAME	FIRSTNAME	JERSEYNUMBER	LASTSPOKENWORDS
1		1 Michael	Jordan		30 I will be back
2		2 David	Becks		100 I will be retiring soon
3		30 Potter	Harry		60 Thanks to my fans

## Read Client

```
public static void main(String[]args){  
    ReadClient client1 = new ReadClient();  
    ClientResponse response = client1.findAll_XML(ClientResponse.class);  
    GenericType<List<Player>> genericType = new GenericType<List<Player>>({});  
    List<Player> data = new ArrayList<Player>();  
    data = (response.getEntity(genericType));  
    System.out.println("Retreiving and Displaying Player Details");  
    for(Player players:data){  
        System.out.println("FirstName:"+players.getFirstname());  
        System.out.println("PlayerID:"+players.getId());  
        System.out.println("JerseyNo:"+players.getJerseynumber());  
    }  
    System.out.println("Data Retreived!!");  
}
```

**Output**

Java DB Database Process x GlassFish Server 3.1.1 x SQL Command 1 execution x playerclient (run) x

```

run:
Retreiving and Displaying Player Details
FirstName:Jordan
PlayerID:1
JerseyNo:30
FirstName:Becks
PlayerID:2
JerseyNo:100
FirstName:Harry
PlayerID:30
JerseyNo:60
Data Retreived!!
BUILD SUCCESSFUL (total time: 1 second)

```

Output Tasks

## Delete Client

```

public static void main(String[]args){
    DeleteClient client1 = new DeleteClient();
    client1.remove("1");
}

```

select \* from MSC17.PLAYE... x

Page Size: 20 Total Rows: 2 Page: 1 of 1 Matching Rows:

#	ID	LASTNAME	FIRSTNAME	JERSEYNUMBER	LASTSPOKENWORDS
1	2	David	Becks	100	I will be retiring soon
2	30	Potter	Harry	60	Thanks to my fans

## Update Client

```

public static void main(String[]args){
    UpdateClient client1 = new UpdateClient();
}

```

```

ClientResponse response = client1.find_XML(ClientResponse.class, "2");
GenericType<Player>genericType = new GenericType<Player>({});
Player player = response.getEntity(genericType);
System.out.println("First Name: "+player.getFirstname());
System.out.println("PlayerId: "+player.getId());
System.out.println("JerseyNo: "+player.getJerseynumber());
System.out.println("Last Name: "+player.getLastname());
System.out.println("Last Spoken Words: "+player.getLastspokenwords());
player.setJerseynumber(100);
player.setLastspokenwords("I will be retiring soon");
client1.edit_XML(player);
}

```

select \* from MSC17.PLAYE... x

Page Size: 20 Total Rows: 3 Page: 1 of 1 Matching Rows:

#	ID	LASTNAME	FIRSTNAME	JERSEYNUMBER	LASTSPOKENWORDS
1	1	Michael	Jordan	30	I will be back
2	2	David	Becks	100	I will be retiring soon
3	30	Potter	Harry	60	Thanks to my fans
			Becks		

## Practical No. 3B

- Develop Micro-blogger application (like Twitter) using RESTful Web services.

### Blogger Server

BloggerServer > pkgserver.bloggerdb > {id}

Resource: pkgserver.bloggerdb/{id}  
(pkgserver.bloggerdb/{id})

Choose method to test: GET MIME: application/xml Add Parameter Test

id: 1

Status: 200 (OK)

Response:

Tabular View	Raw View	Sub-Resource	Headers	Http Monitor
--------------	----------	--------------	---------	--------------

```
<?xml version="1.0" encoding="UTF-8"?> version="1.0" encoding="UTF-8" standalone="yes"
<htmlnerth>
```

Contains commands for working with the selected items.

Status: 200 (OK)

Response:

Tabular View	Raw View	Sub-Resource	Headers	Http Monitor
--------------	----------	--------------	---------	--------------

```
<?xml version="1.0" encoding="UTF-8"?> version="1.0" encoding="UTF-8" standalone="yes"
<bloggerdb>
  <content>Ron the chess Champion</content>
  <messageId></messageId>
  <password>fleaur</password>
  <username>Ronald</username>
</bloggerdb>
```

Contains commands for working with the selected items.

### Blogger Client Create Client

```

public static void main(String[] args){
    CreateClient client1 = new CreateClient();
    ClientResponse response = client1.findAll_XML(ClientResponse.class);
    GenericType<List<Bloggerdb>> genericType = new GenericType<List<Bloggerdb>>({});
    List<Bloggerdb> data = new ArrayList<Bloggerdb>();
    data = (response.getEntity(genericType));
    Bloggerdb b = new Bloggerdb();
    b.setMessageld(4);
    b.setContent("Draco the Slytherine Champion");
    b.setUsername("Draco");
    b.setPassword("Malfoy");
    client1.create_XML(b);
}

```

#	MESSAGE_ID	USERNAME	CONTENT	PASSWORD
1	1	Ronald	Ron the chess Champion	fleaur
2	2	Harry	Harry the Quidditch Champion	Ginny
3	3	Hermione	Hermione the scholar	Viktor
4	4	Draco	Draco the Slytherine Champion	Malfoy

## Update Client

```

public static void main(String[] args){
    UpdateClient client1 = new UpdateClient();
    ClientResponse response = client1.find_XML(ClientResponse.class, "1");
    GenericType<Bloggerdb> genericType = new GenericType<Bloggerdb>({});
    Bloggerdb b = response.getEntity(genericType);
    System.out.println("Message id:"+b.getMessageld());
    System.out.println("Content:"+b.getContent());
    System.out.println("Username:"+b.getUsername());
    System.out.println("Password:"+b.getPassword());
    b.setContent("I love food");
    client1.edit_XML(b);
}

```

#	MESSAGE_ID	USERNAME	CONTENT	PASSWORD
1	1	Ronald	I love food	fleur
2	2	Harry	Harry the Quidditch Champion	Ginny
3	3	Hermione	Hermione the scholar	Viktor
4	4	Draco	Draco the Slytherine Champion	Malfoy

### Delete Client

```
public static void main(String[] args){
    DeleteClient client1 = new DeleteClient();
    client1.remove("4");
    System.out.println("Successfully Deleted!!");
}
```

#	MESSAGE_ID	USERNAME	CONTENT	PASSWORD
1	1	Ronald	I love food	fleur
2	2	Harry	Harry the Quidditch Champion	Ginny
3	3	Hermione	Hermione the scholar	Viktor

### Read Client

```
public static void main(String[] args){
    ReadClient client1 = new ReadClient();
    ClientResponse response = client1.findAll_XML(ClientResponse.class);
    GenericType<List<Bloggerdb>> genericType = new GenericType<List<Bloggerdb>>();
    List<Bloggerdb> data = new ArrayList<Bloggerdb>();
    data = (response.getEntity(genericType));
    System.out.println("Retreiving and Displaying Player Details");
    for(Bloggerdb b:data){
        System.out.println("Content: "+b.getContent());
        System.out.println("Username: "+b.getUsername());
        System.out.println("Password:"+b.getPassword());
    }
    System.out.println("Data Retreived!!");
}
```

}

Output x Tasks

Java DB Database Process x GlassFish Server 3.1.1 x BloggerClient (run) x

```
run:
Retreiving and Displaying Player Details
Content: I love food
Username: Ronald
Password:fleaur
Content: Harry the Quidditch Champion
Username: Harry
Password:Ginny
Content: Hermione the scholar
Username: Hermione
Password:Viktor
Data Retreived!!
BUILD SUCCESSFUL (total time: 1 second)
```

## Practical-4

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

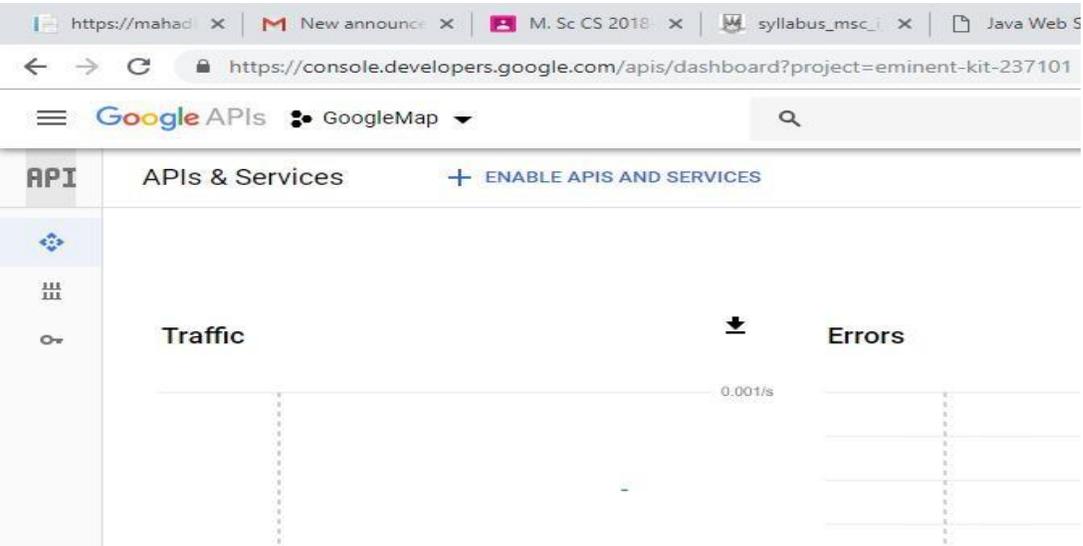
- Develop application to consume Google's search / Google's Map RESTful Web service.

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

2. The code inside the input.jsp will be similar to this input.jsp Input.jsp

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

3. 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).



- Create a new API Project.



- Enter the name to your project.

https://mahad x M New announc x M. Sc CS 2018 x syllabus\_msc\_i x Java Web Serv x

https://console.developers.google.com/projectcreate?previousPage=%2Fapis%2Fdashboard

Google APIs

### New Project

You have 7 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name \*  
My Project 81907

Project ID: zeta-environs-237517. It cannot be changed later. [EDIT](#)

Location \*  
No organization [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

- Enable the Google Maps API V3.

https://mahad x M New announc x M. Sc CS 2018 x syllabus\_msc\_i x Java Web Serv x Microsoft Wor x (19) Alan V

https://console.developers.google.com/apis/library/maps-backend.googleapis.com?id=fd73ab50-9916-4cde-a0f6-dc8be0a0d425&

Google APIs GoogleMap

### API Library

#### Maps JavaScript API

Google

Maps for your website

[MANAGE](#)

API enabled

Type  
APIs & services

Last updated  
1/10/19, 2:02 AM

Category  
Maps

Service name  
maps-backend.googleapis.com

Overview

Add a map to your website, providing imagery and local data from the same source as Google Maps. Style the map to suit your needs. Visualize your own data on the map, bring the world to life with Street View, and use services like geocoding and directions.

About Google

Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.

Tutorials and documentation

4. Create another file index.jsp

**Index.jsp**

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

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
#map {
    height: 400px;
    width: 100%;
}
</style>
</head>
<body>

<%
    double lati=Double.parseDouble(request.getParameter("t1"));
    double longi=Double.parseDouble(request.getParameter("t2"));
%>
<h3> Google Maps </h3>
<div id="map"></div>
<script lang="javascript">
    function initMap() {
        var info={lat: <%=lati%>, lng: <%=longi%>};
        var map = new google.maps.Map(document.getElementById('map'),
            {
                zoom: 4, center: info
            });
        var marker = new google.maps.Marker({ position:
            info,
            map: map
        });
    }
</script>
<script async defer
    src="put your key here">
```

</script>

</body>

</html>

### Output :-

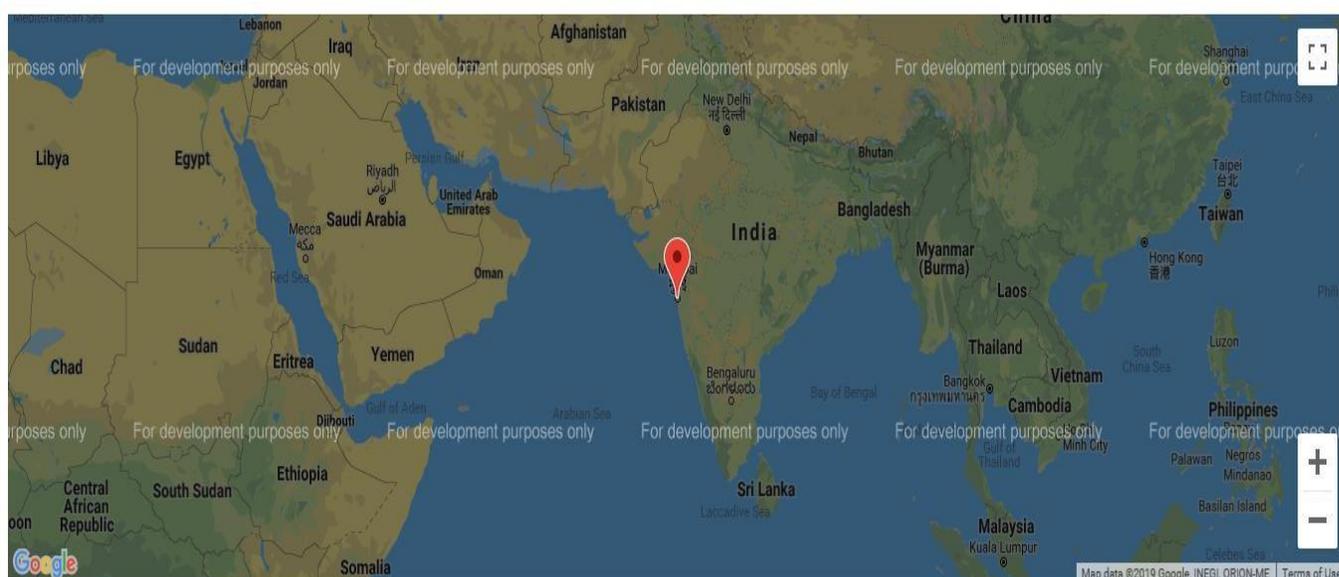


Enter latitude:

Enter longitude:



### Google Maps



## Practical-5

Installation and Configuration of virtualization using KVM

## Practical-6

Develop application to download image/video from server or upload image/video to server using MTOM techniques

MTOMClient

Index.jsp

```
<%@page import="java.io.BufferedOutputStream">
<%@page import="java.io.FileOutputStream">
<%@page import="java.io.FileInputStream">
<%@page import="java.io.BufferedInputStream">
<%@page import="javax.imageio.stream.FileImageInputStream">
<%@page import="java.io.File">
<%@page import="javax.xml.ws.soap.MTOMFeature">
<%@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>
```

```

<%-- start web service invocation --><hr/>
<%
try {
pkg.ImageWS_Service service = new pkg.ImageWS_Service();
pkg.ImageWS port = service.getImageWSPort(new MTOMFeature(60000));
// TODO initialize WS operation arguments here
String filePath="c:/msc/ABC.jpg";
File file=new File(filePath);
FileInputStream fis=new FileInputStream(file);
BufferedInputStream bis=new BufferedInputStream(fis);
java.lang.String filename = file.getName();
byte[]imageBytes=new byte[(int)file.length()];
bis.read(imageBytes);
port.upload(filename, imageBytes);
bis.close();
out.println("File uploaded :"+filePath);
} catch (Exception ex) {
// TODO handle custom exceptions here
}
}
>
<%-- end web service invocation --><hr/>

```

```

<%-- start web service invocation --><hr/>
<%
try {

pkg.ImageWS_Service service = new pkg.ImageWS_Service();
pkg.ImageWS port = service.getImageWSPort();
// TODO initialize WS operation arguments here
java.lang.String filename = "ABC.jpg";
String filePath="c:/msc/download/"+filename;
// TODO process result here
byte[] fileBytes = port.download(filename);
FileOutputStream fos=new FileOutputStream(filePath);
BufferedOutputStream bos=new BufferedOutputStream(fos);
bos.write(fileBytes);
bos.close();
out.println("File downloaded"+filePath);
} catch (Exception ex) {
// TODO handle custom exceptions here
}
}
>
<%-- end web service invocation --><hr/>

```

```

</body>
</html>
MTOMServer
ImageWS.java
package mypkg;
import java.io.*;
import javax.jws.Oneway;

```

```

import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.xml.ws.soap.MTOM;
@MTOM(enabled=true,threshold=60000)
@WebService(serviceName = "ImageWS")
public class ImageWS {
    @WebMethod(operationName = "upload")
    @Oneway
    public void upload(@WebParam(name = "Filename") String Filename, @WebParam(name =
    "ImageBytes") byte[] ImageBytes){
    String filePath="C:/MSC/upload/"+Filename;
    try
    {
    FileOutputStream fos=new FileOutputStream(filePath);
    BufferedOutputStream bos=new BufferedOutputStream(fos);
    bos.write(ImageBytes);
    bos.close();
    System.out.println("Recieved file :"+filePath);
    }
    catch(Exception ex)

    {
    System.out.println(ex);
    }
    }
    @WebMethod(operationName = "download")
    public byte[] download(@WebParam(name = "Filename") String Filename) {
    String filePath="C:/MSC/upload/"+Filename;
    System.out.println("Sending file :"+filePath);
    try
    {
    File file=new File(filePath);
    FileInputStream fis=new FileInputStream(file);
    BufferedInputStream bis=new BufferedInputStream(fis);
    byte[] fileBytes=new byte[(int)file.length()];
    bis.read(fileBytes);
    bis.close();
    return fileBytes;
    }
    catch(Exception ex)
    {
    System.out.println(ex);
    }
    return null;
    }
    }
    }
    Output:

```

## **Practical-07**

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

## **FOSS Cloud Installation**

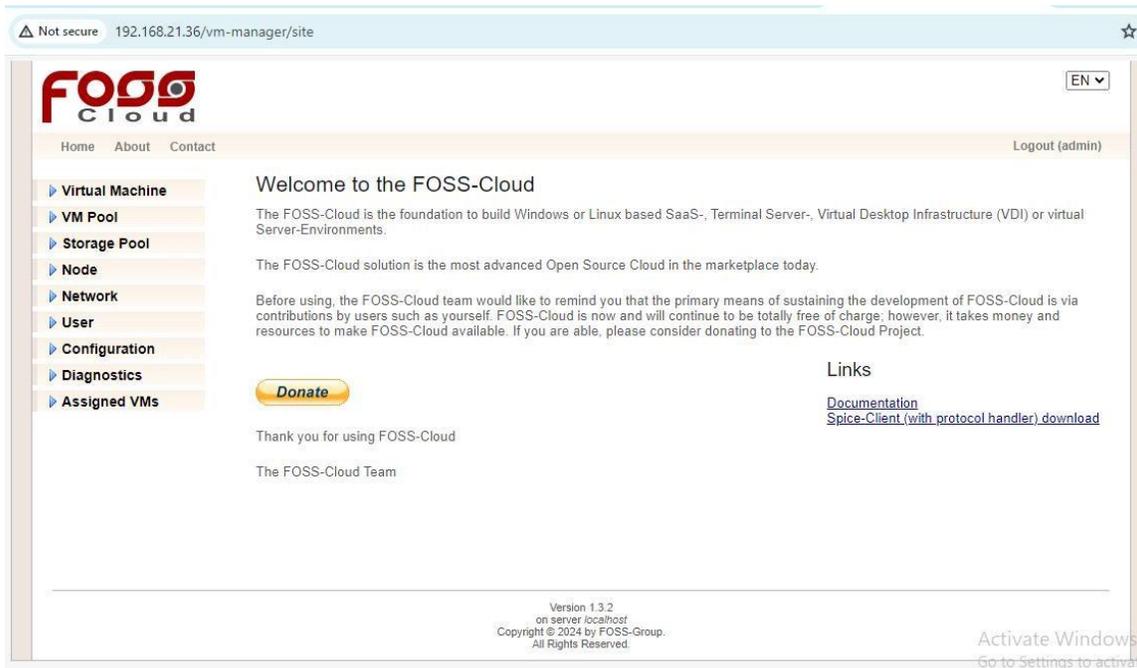
### **Steps:**

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"

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

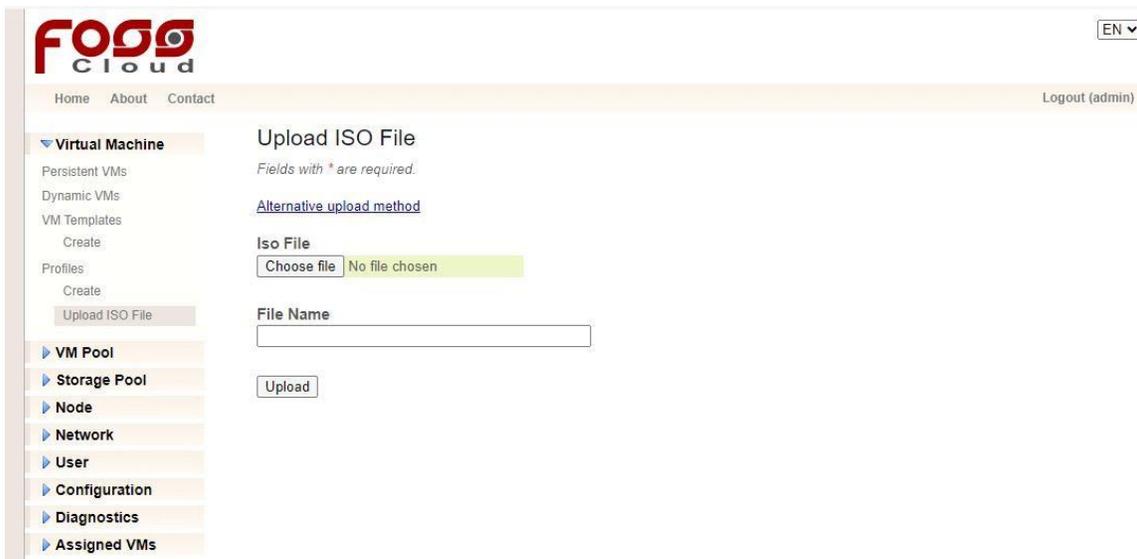
1. Install virt-viewer.
2. Install spice-client.0.6.3.
3. Open Browser.

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

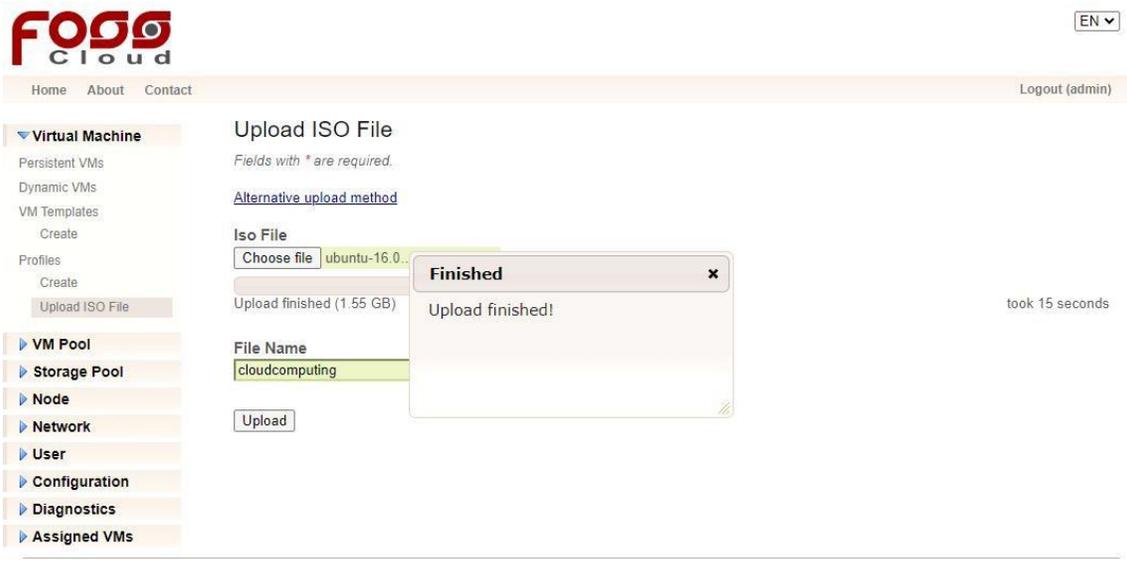


## •Uploading ISO files to Foss cloud

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 a 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")

**Virtual Machine**

Persistent VMs  
Dynamic VMs  
VM Templates  
Create  
Profiles  
Create  
Upload ISO File

**VM Pool**

**Storage Pool**

**Node**

**Network**

**User**

**Configuration**

**Diagnostics**

**Assigned VMs**

## Create VM Profile

Fields with \* are required.

**Step I**  
Please select a profile first!

**BaseProfile**

- linux
  - windows
    - default
    - nasir34
    - vip
    - TYCS2024
      - x86\_64
      - en-US**
  - foss\_server
  - ubuntu\_19
  - priyanka
  - kaypan
  - ky pn
  - Ubuntu VM
  - VM414
  - khushboo
  - ubuntusk
  - Roman
  - virtual
  - R\_UBUNTU
  - VM441

**Step II**  
Override the default values if necessary!

**Isofile**

ubuntu.iso  
Students.iso  
ubuntu-16.04.5-desktop-i386.iso  
CC.iso

**Name \***  
TYCS2024

**Description \***  
Welcome To The World Of Programming

**Memory \***  
256 MB ————— 128 GB ————— 2 GB

**Volume Capacity \***  
10 GB ————— 2048 GB ————— 10 GB

**CPU \***  
1

**Clock Offset \***  
localtime

Activate \ Go to Setting

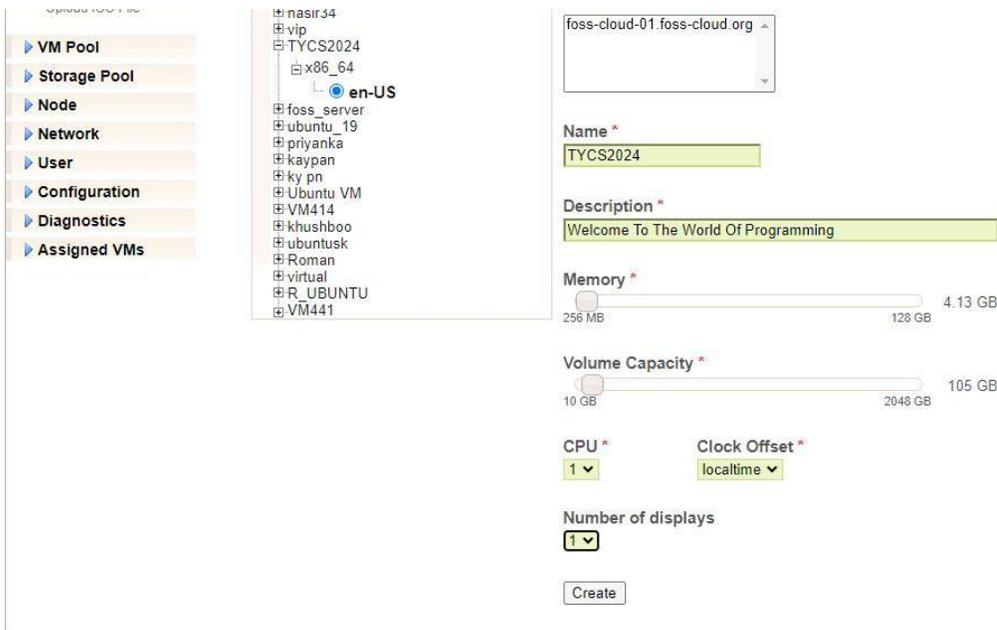
Create

Now, we will create a VM template.

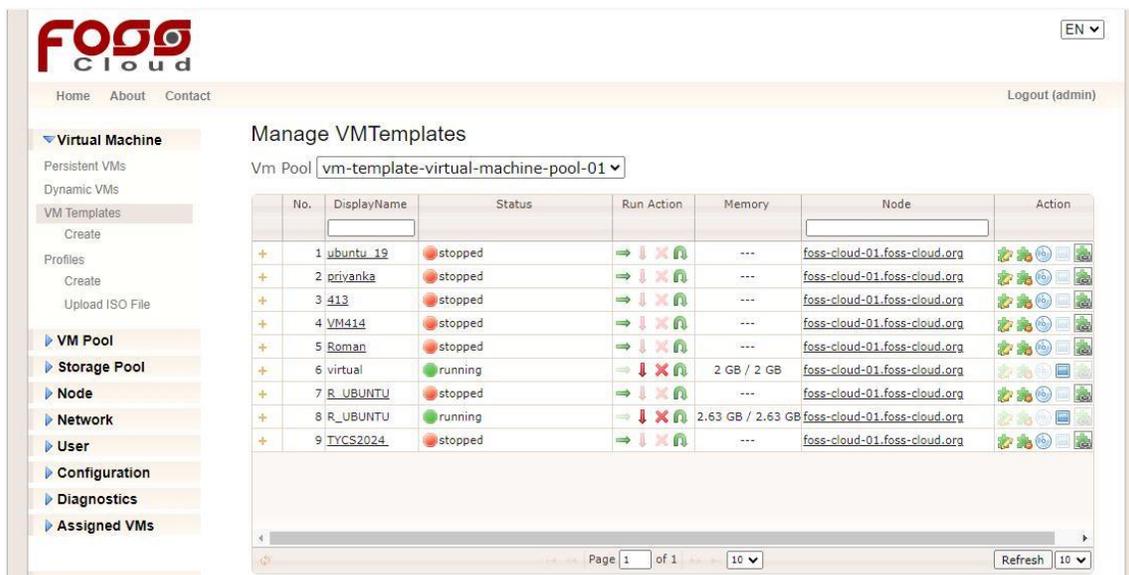
### Creating Template

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.

## Virtual Machine

Persistent VMs

Dynamic VMs

## VM Templates

 Create  
 Profiles  
 Create  
 Upload ISO File

## VM Pool

## Storage Pool

## Node

## Network

## User

## Configuration

## Diagnostics

## Assigned VMs

## Manage VM Templates

Vm Pool vm-template-virtual-machine-pool-01

No.	DisplayName	Status	Run Action	Memory	Node	Action
+	1 ubuntu_19	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	2 privanka	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	3 413	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	4 VM414	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	5 Roman	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	6 virtual	running	→ ↓ × ↻	2 GB / 2 GB	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	7 R_UBUNTU	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	8 R_UBUNTU	running	→ ↓ × ↻	2.63 GB / 2.63 GB	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	9 TYCS2024	running	→ ↓ × ↻	4 GB / 4 GB	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️

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

Open remote-viewer?  
 http://192.168.21.36 wants to open this application.

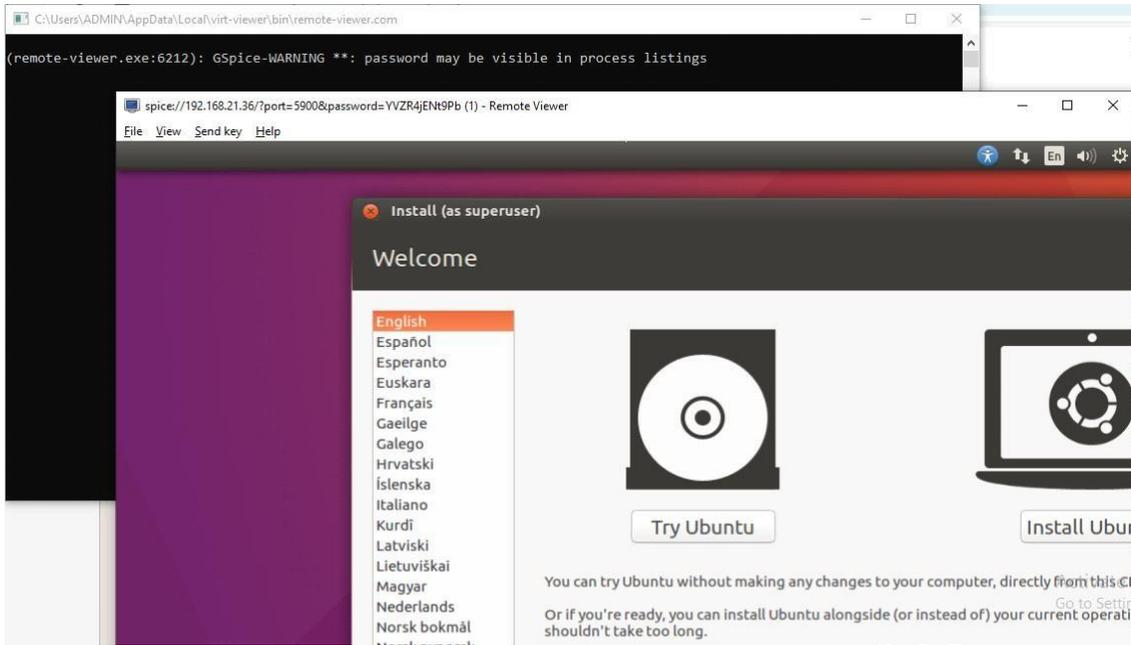
Open remote-viewer Cancel

Manage VM Templates  
 Vm Pool vm-template-virtual-machine-pool-01

No.	DisplayName	Status	Run Action	Memory	Node	Action
+	1 ubuntu_19	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	2 privanka	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	3 413	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	4 VM414	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	5 Roman	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	6 virtual	running	→ ↓ × ↻	2 GB / 2 GB	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	7 R_UBUNTU	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	8 R_UBUNTU	running	→ ↓ × ↻	2.63 GB / 2.63 GB	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️
+	9 TYCS2024	running	→ ↓ × ↻	4 GB / 4 GB	foss-cloud-01.foss-cloud.org	🌱 🔄 📄 🗑️

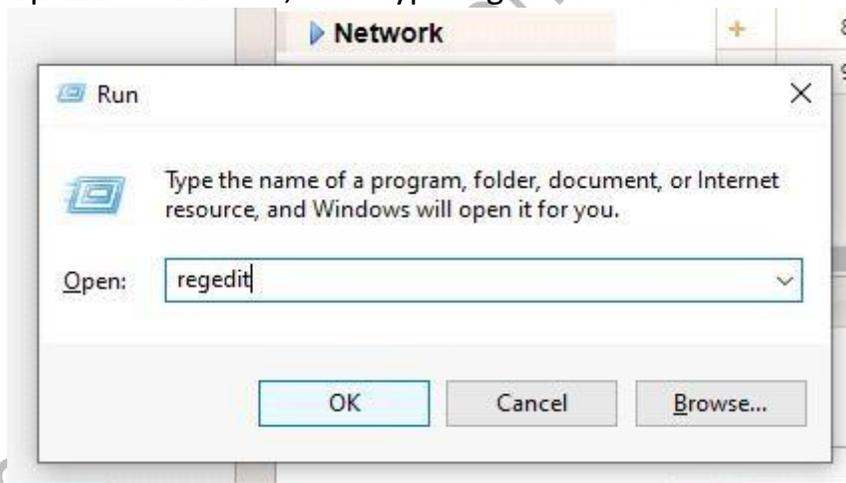
Page 1 of 1 Refresh 10

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.



**Note :- If you don't see the above screen follow the below given steps.**

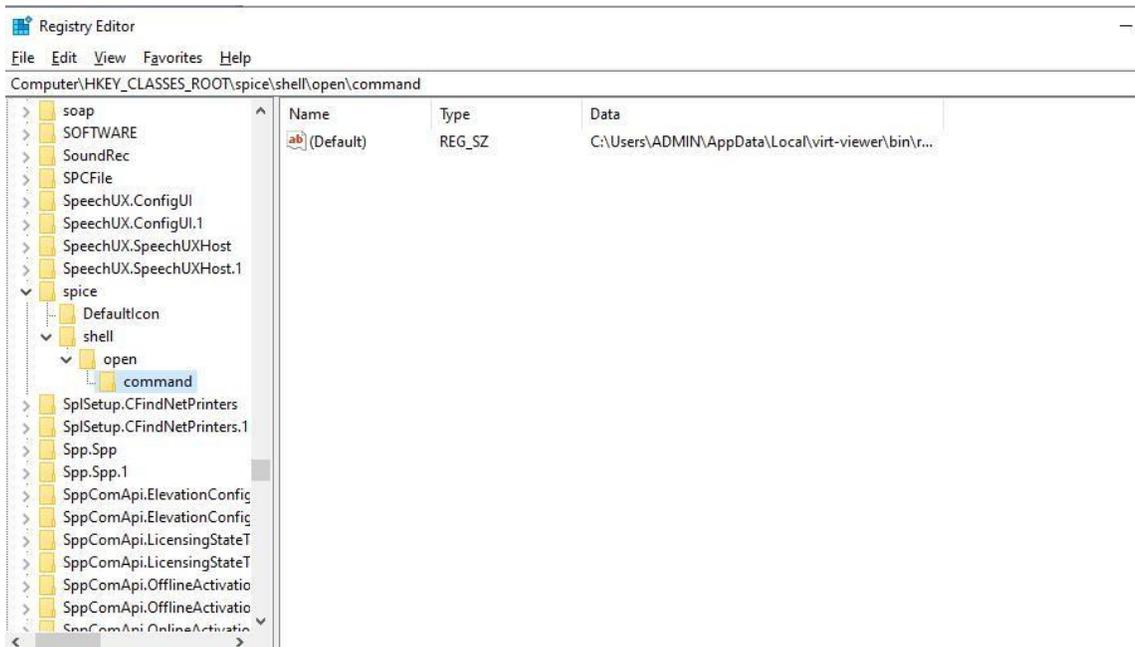
1. First press window + r, then type regedit and click ok.



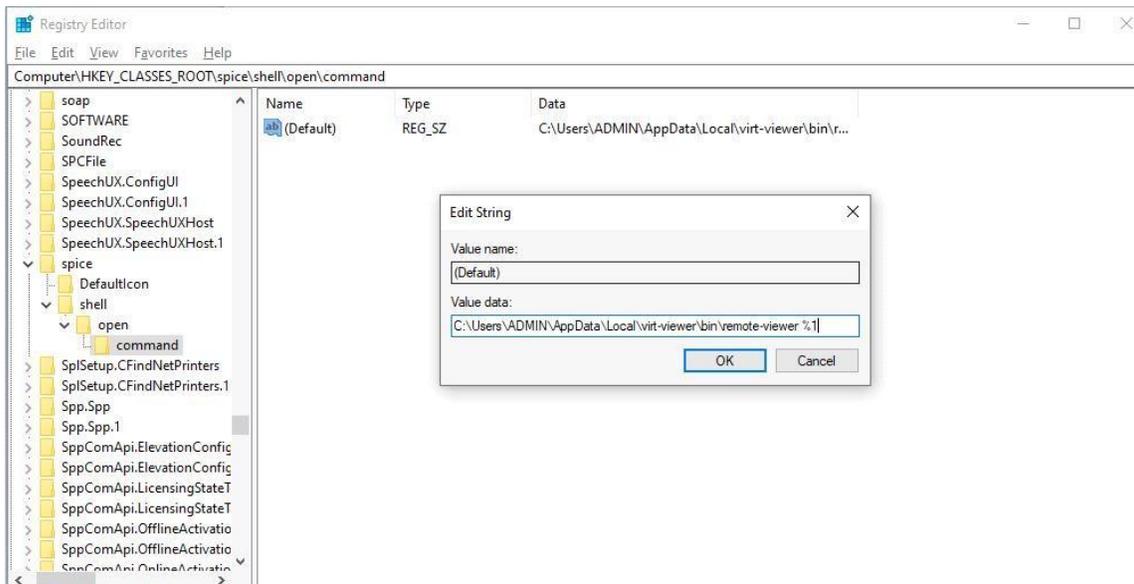
2. Further permissions tab will open then click yes, and a regedit editor will get open.

3. In the Registry Editor a list of directories would appear, in those lists of directories search for the spice directory.

4. Keep on expanding the spice directory until you reach the command section.

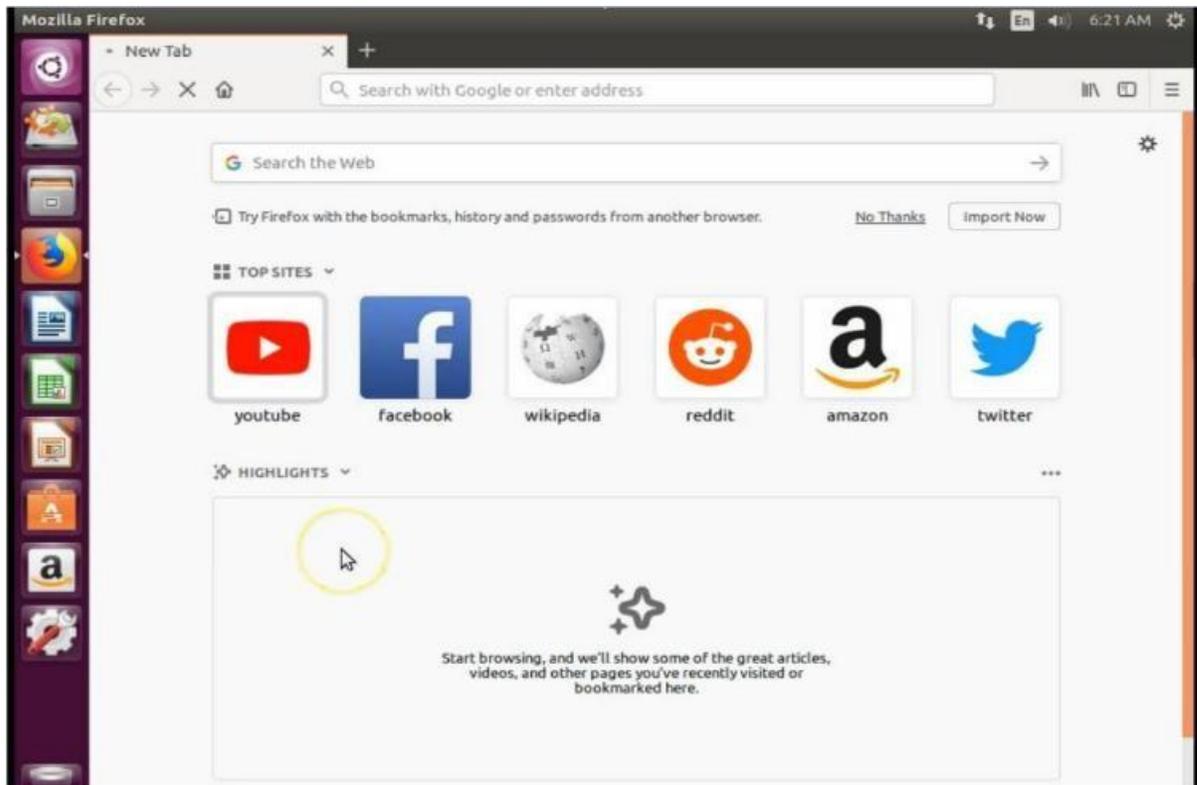
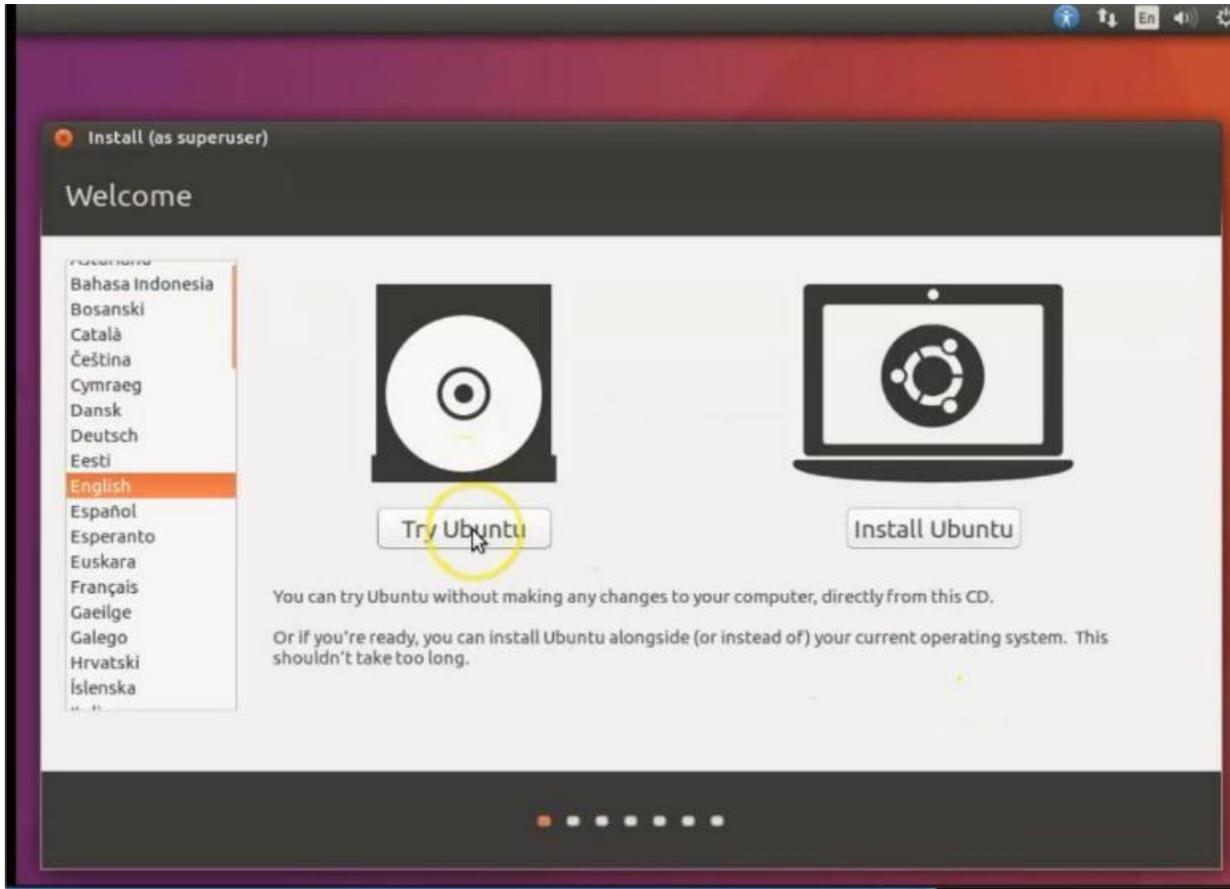


5. Then click on the default file and a pop up will appear, then you have to paste the path **C:\Users\ADMIN\AppData\Local\virt-viewer\bin** in the value data and **also type %1 in last**. Once done the above screen will appear.



Finally, after clicking on remote viewer button a command line come where a graphical control loading and getting connected will be displayed and then finally the ubuntu will get open.

On opening of ubuntu click on try ubuntu. Now, you can use ubuntu on your virtual machine.

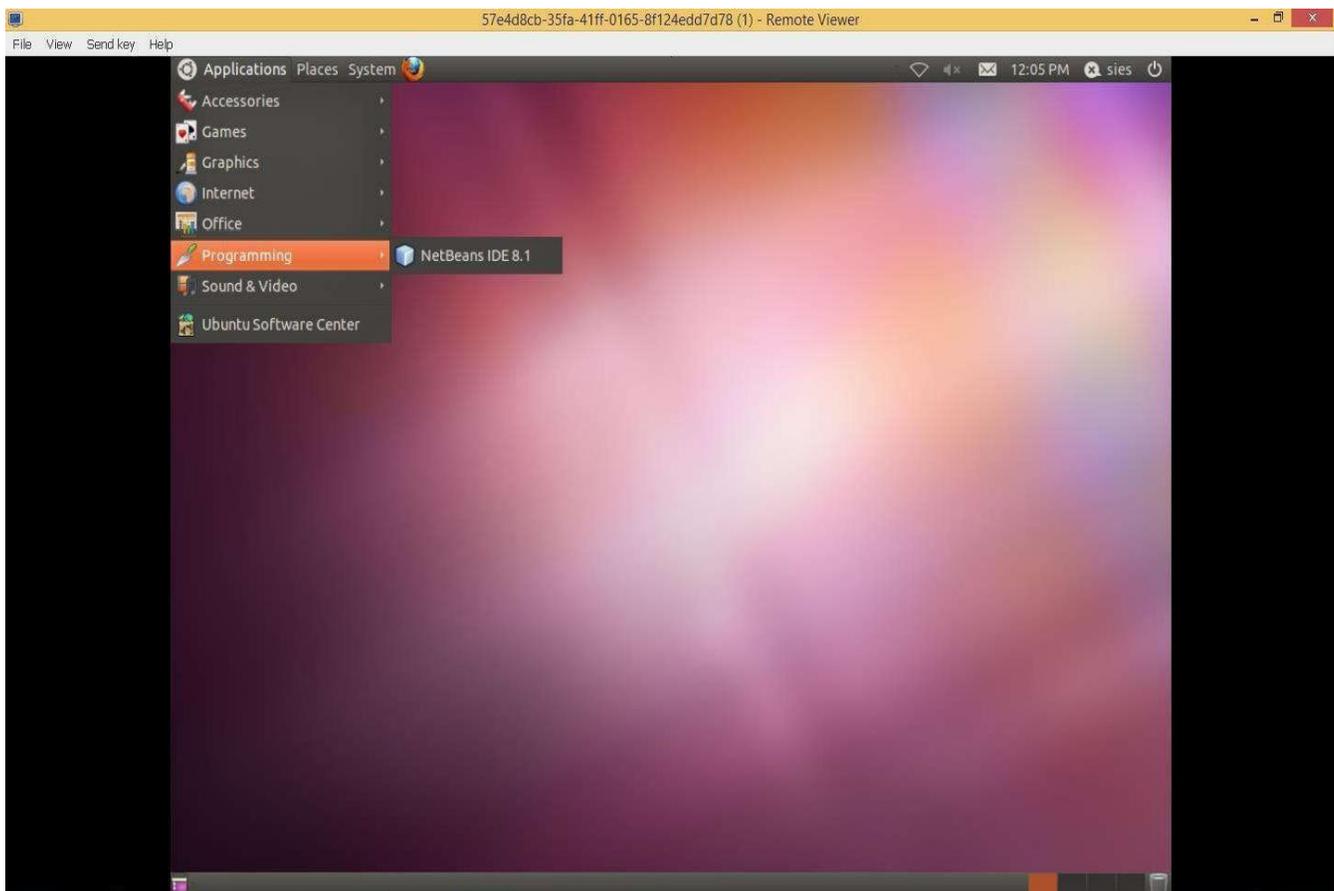


## Practical-08

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.

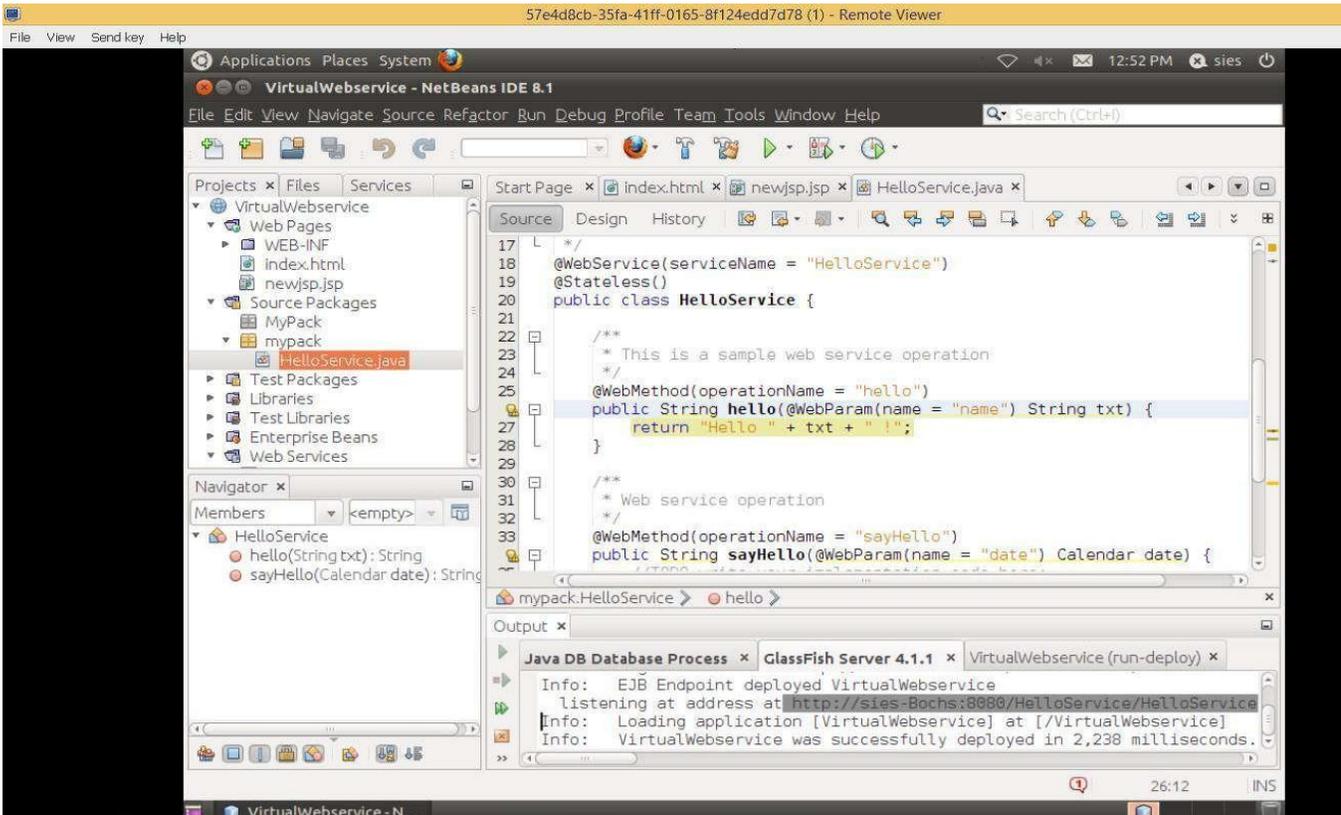
Installation of Netbeans, Eclipse, Visual Studio and DBMS can be done in the appropriate Virtual Machines.



**Aim:** Implement FOSS-Cloud Functionality VSI Software as a Service (SaaS)

Applications created and deployed in the virtual machines can be accessed by outside world.

This can be implemented as Software as Service.



Applications Places System  
Mozilla Firefox  
File Edit View History Bookmarks Tools Help  
http://sies-boc...lloService?wsdl  
http://sies-bochs:8080/HelloService/HelloService?wsdl  
Google

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!--  
  Published by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b608 (trunk-7979; 2015-01-21T12:50:19+0000) JAXWS-RI/2.2.11-b1  
-->  
<!--  
  Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b608 (trunk-7979; 2015-01-21T12:50:19+0000) JAXWS-RI/2.2.11-b1  
-->  
<definitions targetNamespace="http://mypack/" name="HelloService">  
  <types>  
    <xsd:schema>  
      <xsd:import namespace="http://mypack/" schemaLocation="http://sies-bochs:8080/HelloService/HelloService?xsd=1"/>  
    </xsd:schema>  
  </types>  
  <message name="hello">  
    <part name="parameters" element="tns:hello"/>  
  </message>  
  <message name="helloResponse">  
    <part name="parameters" element="tns:helloResponse"/>  
  </message>  
  <message name="sayHello">  
    <part name="parameters" element="tns:sayHello"/>  
  </message>  
  <message name="sayHelloResponse">  
    <part name="parameters" element="tns:sayHelloResponse"/>  
  </message>  
</portType name="HelloService">
```

## INDEX

Sr.No	Name of the Practical	Date	Signature
1	Define a simple services like Converting Rs into Dollar and Call it from different platform like JAVA and .NET		<u>Mugolhg</u>
2	Create a Simple SOAP service.		<u>Mugolhg</u>
3	Create a Simple REST Service.		<u>Mugolhg</u>
4	Develop application to consume Google's search / Google's Map RESTful Web service.		<u>Mugolhg</u>
5	Installation and Configuration of virtualization using KVM.		<u>Mugolhg</u>
6	Develop application to download image/video from server or upload image/video to server using MTOM techniques		<u>Mugolhg</u>
7	Implement FOSS-Cloud Functionality VSI (Virtual Server Infrastructure) Infrastructure as a Service (IaaS), Storage		<u>Mugolhg</u>
8	Implement FOSS-Cloud Functionality - VSI Platform as a Service (PaaS),		<u>Mugolhg</u>
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		<u>Mugolhg</u>
10	Implementation of Openstack with user and private network creation.		<u>Mugolhg</u>

# Practical No 1

## By Chandan

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

### Solution:

#### currency\_converter.py

```
from flask import Flask, request, jsonify
app = Flask(__name__)
# Conversion rate: 1 INR = 0.012 USD (example rate) conversion_rate =
0.012
@app.route('/convert', methods=['GET']) def
convert_currency():
    inr = request.args.get('inr') if
    inr:
        try:
            inr_value = float(inr)
            usd_value = inr_value * conversion_rate return
            jsonify({"INR": inr_value, "USD": usd_value})
        except ValueError:
            return jsonify({"error": "Invalid INR amount"}), 400
    return jsonify({"error": "INR amount missing"}), 400
if __name__ == '__main__': app.run(debug=True)
```

#### CurrencyConverterClient.java

```
import java.io.BufferedReader; import
java.io.InputStreamReader; import
java.net.HttpURLConnection; import
java.net.URL;
import java.util.Scanner;
public class CurrencyConverterClient {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter amount in INR: "); String
        inrAmount = scanner.nextLine(); // User input
        try {
            String urlString = "http://127.0.0.1:5000/convert?inr=" + inrAmount;
```

```

URL url = new URL(urlString);
URLConnection conn = (URLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Accept", "application/json");
if (conn.getResponseCode() != 200) { throw new
    RuntimeException("Failed : HTTP error code : "
        + conn.getResponseCode());
}
BufferedReader br = new BufferedReader(new InputStreamReader(
    (conn.getInputStream())));
String output;
System.out.println("Currency Conversion:"); while
((output = br.readLine()) != null) {
    System.out.println(output);
}
conn.disconnect();
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

### Output:

```

PS D:\TYCS\CC\cc testing-20241212T025333Z-001\cc testing> & "C:/Program Files/Py
thon313/python.exe" "d:/TYCS/CC/cc testing-20241212T025333Z-001/cc testing/curre
ncy_converter.py"
* Serving Flask app 'currency_converter'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 379-447-875

```

Now take new terminal

```

Enter amount in INR: 2000
Currency Conversion:
{
  "INR": 2000.0,
  "USD": 24.0
}

```

# Practical No 2

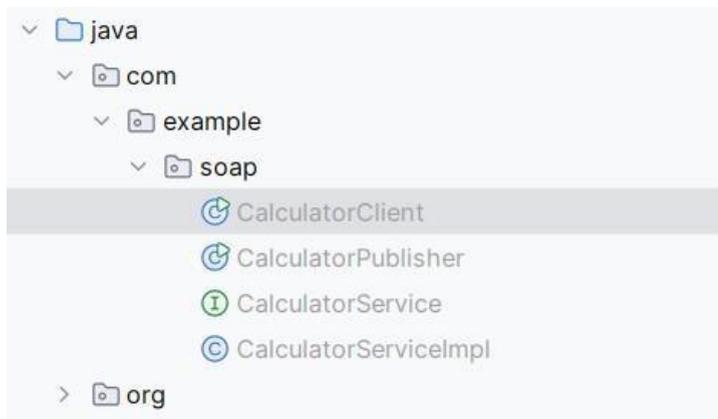
## By Chandan (java 8v)

**Aim:** Create a Simple SOAP service

**Solution:**

**Step 1:** Download JDK 8

**Step 2:** Create New Project in IntelliJ Idea & Select Java 8 & Maven Structure



**CalculatorServiceImpl.java**

```
package com.example.soap;
```

```
import javax.xml.ws.WebService;
```

```
@WebService(endpointInterface =  
"com.example.soap.CalculatorService") public class  
CalculatorServiceImpl implements CalculatorService {
```

```
    @Override public int  
add(int num1, int num2) {  
return num1 + num2;  
    }  
}
```

**CalculatorService.java**

```
package com.example.soap;
```

```
import javax.xml.ws.WebService;
```

```
@WebService
```

```
public interface CalculatorService
{   public int add(int num1, int
num2);
}
```

### CalculatorPublisher.java

```
package com.example.soap;
```

```
import javax.xml.ws.Endpoint;
```

```
public class CalculatorPublisher {
```

```
    public static void main(String[] args) {
        // Publishing the service at a specific URL
        Endpoint.publish("http://localhost:8080/calculator", new CalculatorServiceImpl());
        System.out.println("Service is running at http://localhost:8080/calculator");
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
Service is running at http://localhost:8080/calculator
```

```
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<!-- Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.3-1.0-201206-1835. SW-RI's version is 2.2.3-1.0-201206-1835. -->
<!-- Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.3-1.0-201206-1835. SW-RI's version is 2.2.3-1.0-201206-1835. -->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://soap.example.com/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns-="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://soap.example.com/" name="CalculatorService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://soap.example.com/" schemalocation="http://localhost:8080/calculator?xsd-1"/>
    </xsd:schema>
  </types>
  <message name="add">
    <part name="parameters" element="tns:add"/>
  </message>
  <message name="addResponse">
    <part name="parameters" element="tns:addResponse"/>
  </message>
  <portType name="CalculatorService">
    <operation name="add">
      <input wsam:Action="http://soap.example.com/CalculatorService/addRequest" message="tns:add"/>
      <output wsam:Action="http://soap.example.com/CalculatorService/addResponse" message="tns:addResponse"/>
    </operation>
  </portType>
  <binding name="CalculatorServiceImplPortBinding" type="tns:CalculatorService">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  </binding>
  <operation name="add">
    <soap:operation soapAction="">
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </operation>
```

### CalculatorClient.java

```
package com.example.soap;
```

```
import
```

```
javax.xml.namespace.QName;
```

```
import javax.xml.ws.Service;
import java.net.URL;

public class CalculatorClient {

    public static void main(String[] args) throws Exception {
        // URL to the WSDL
        URL url = new URL("http://localhost:8080/calculator?wsdl");

        // Correct QName based on the WSDL (Check the actual service name)
        QName qname = new QName("http://soap.example.com/",
            "CalculatorServiceImplService");

        // Creating service instance
        Service service = Service.create(url, qname);

        // Getting the port and invoking the method
        CalculatorService calculator =
            service.getPort(CalculatorService.class);

        int result = calculator.add(10, 20);
        System.out.println("Result: " + result);
    }
}
```

### Output:

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
```

```
Result: 30
```

```
Process finished with exit code 0
```

# Practical No 3

By Chandan

**Aim:** Create a Simple REST Service.

**Solution:**

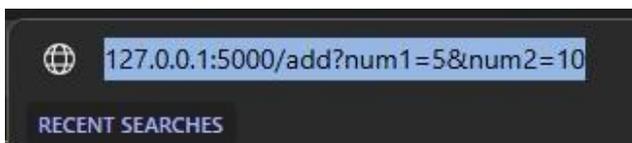
**add\_numbers.py**

```
from flask import Flask, request, jsonify
app = Flask(__name__)
# Route to add two numbers

@app.route('/add', methods=['GET'])
def add_numbers():
    # Get numbers from query parameters
    num1 = float(request.args.get('num1'))
    num2 = float(request.args.get('num2'))
    # Calculate the sum
    result = num1 + num2
    # Return the result as JSON
    return jsonify({"result": result})

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

```
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 379-447-875
```



**Output:**



# Practical No 4

## By Chandan

### Aim:

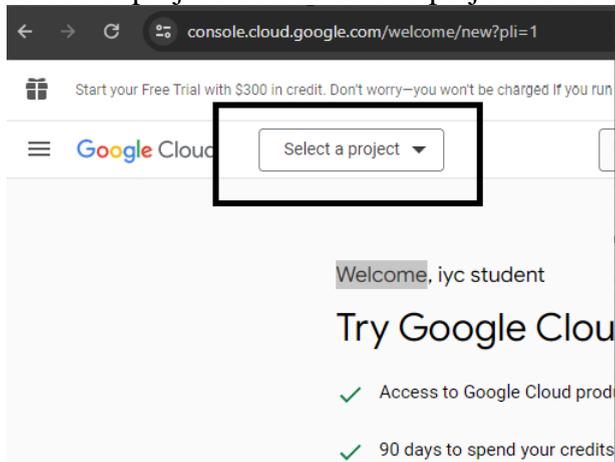
Develop an application to consume Google's search / Google's Map RESTful Web service.

### Solution:

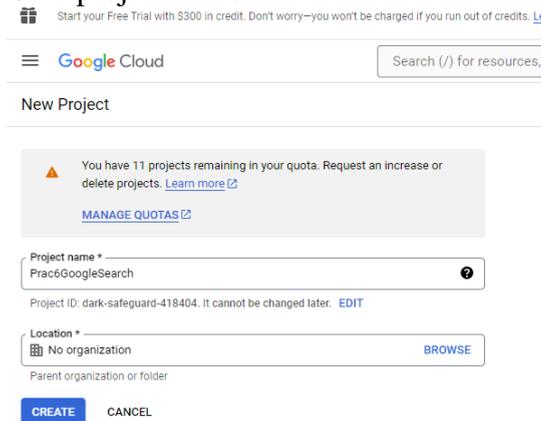
**Step 1:** Search "console cloud google" on chrome and sign up

<https://console.cloud.google.com>

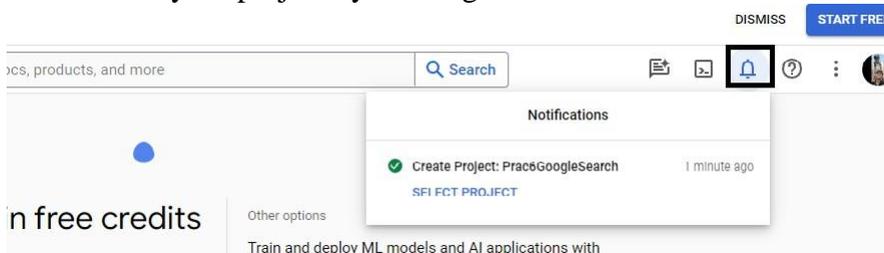
Click on project and create new project



Give project name and then click create

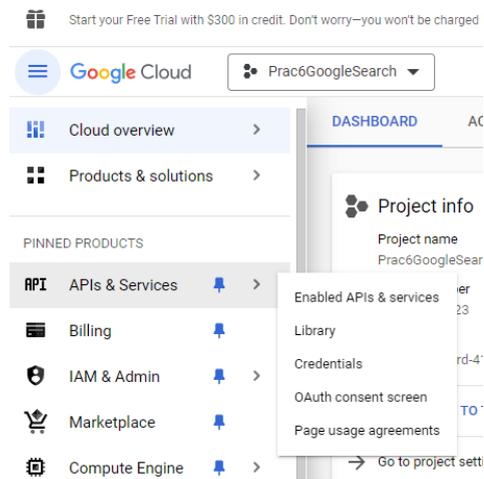


You can see your project by clicking bell icon

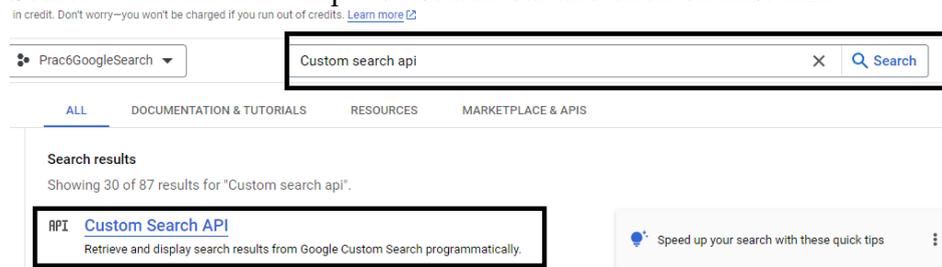


**Step 2:** Click on bell icon and click "select project"

Once you select the project then click on top-left burger icon → Click “APIs and Services” → Enabled APIs and services



Search “Custom search api” on search bar and click on first link

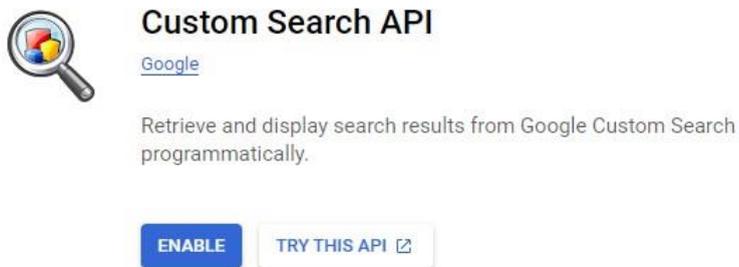


In google cloud project you always have to specify which functionality you want to enable.

Click on enable to enable custom search api

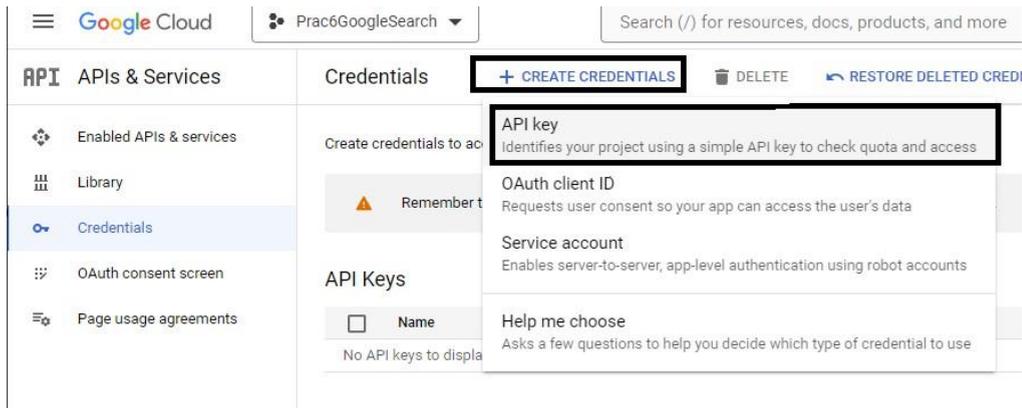
This enable the custom search api for this particular project

← Product details



**Step 3:** Now you need to generate an api key to authenticate yourself from python.

In the Api & Services section click on credentials→ click create credentials→Click API key



## API key created

Use this key in your application by passing it with the `key=API_KEY` parameter.



⚠ This key is unrestricted. To prevent unauthorized use, we recommend restricting where and for which APIs it can be used. [Edit API key](#) to add restrictions. [Learn more](#)

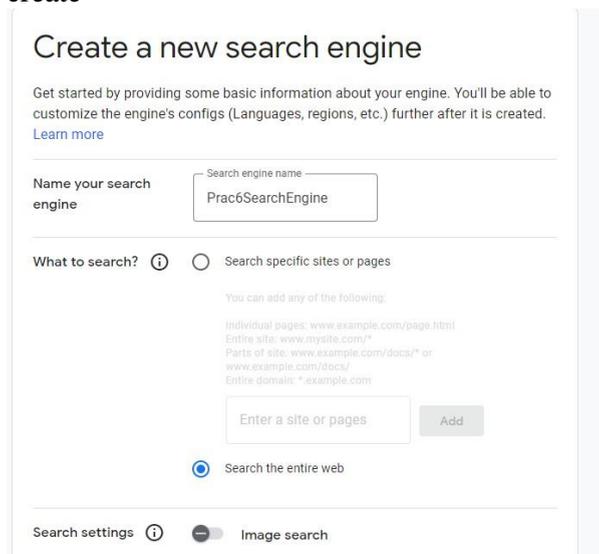
CLOSE

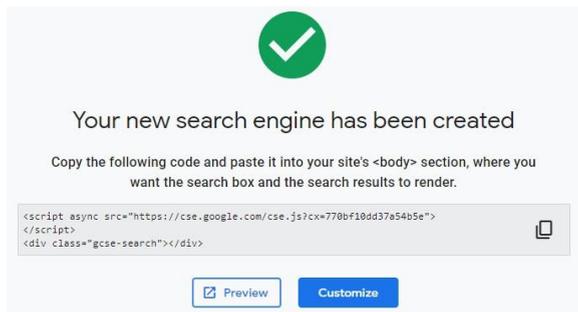
Create a file as `API_KEY` in your project and paste `api_key` value in it

### Step 4: Create a search engine

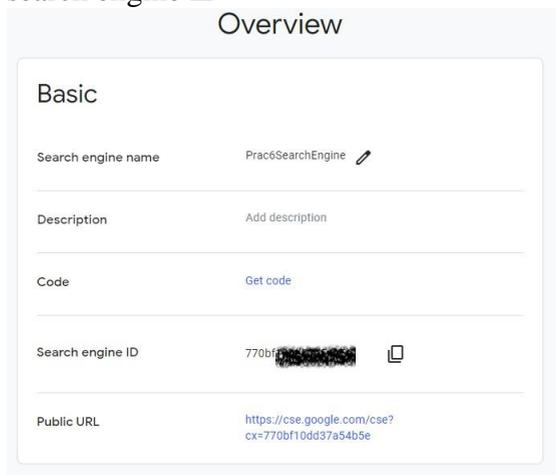
Go to google search programmable search engine → Open first website and click get started

Give name to your search engine and select the "search the entire website" and click create





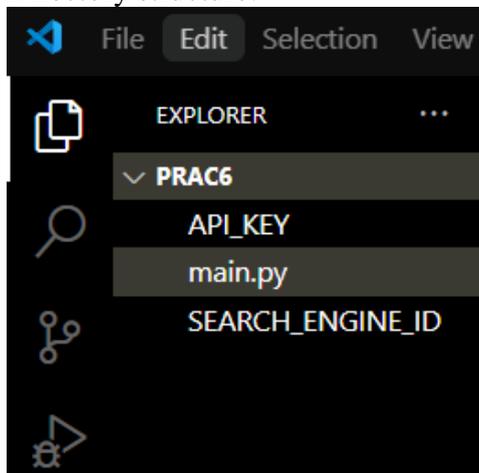
**Step 5:** Click “go back to all search engine” → click on your search engine → Copy the search engine ID



Create a new file on your project as SEARCH\_ENGINE\_ID and paste your search engine ID

Download requests using pip3 install requests in the command line .

Directory structure:



**Step 6:** Get text results using google search

Create a file main.py and write a below code in it

**Code:**

```

import requests

# Replace with your actual API key and Search Engine ID
API_KEY = 'Enter your Api Key'
SEARCH_ENGINE_ID = 'Enter your Engine ID'

search_query = 'ismail yusuf college'

url = 'https://www.googleapis.com/customsearch/v1'

params = {
    'q': search_query,
    'key': API_KEY,
    'cx': SEARCH_ENGINE_ID
}

# Send request to the API
response = requests.get(url, params=params)

# Get the response data in JSON format
results = response.json()

# Check if there are results and print the first result's link
if 'items' in results:
    print(results['items'][0]['link'])
else:
    print("No results found.")

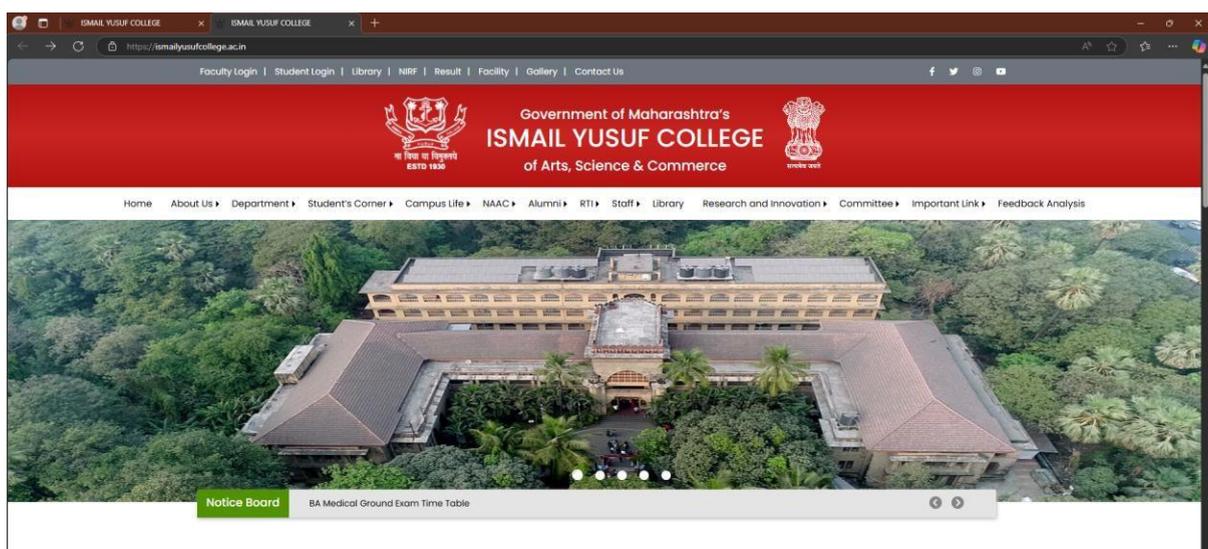
```

## Output:

```

PS D:\TYCS\CC\Prac4> & "C:/Program Files/Python313/python.exe" d:/TYCS/CC/Prac4/Current/main.py
https://ismailyusufcollege.ac.in/
PS D:\TYCS\CC\Prac4>

```





# Practical No 5

By Chandan

## Aim:

Installation and Configuration of virtualization using KVM.

## Solution:

**Step 1:** Select Ubuntu platform Virtual Machine or Physical Ubuntu Machine recommended: go with Dual Boot

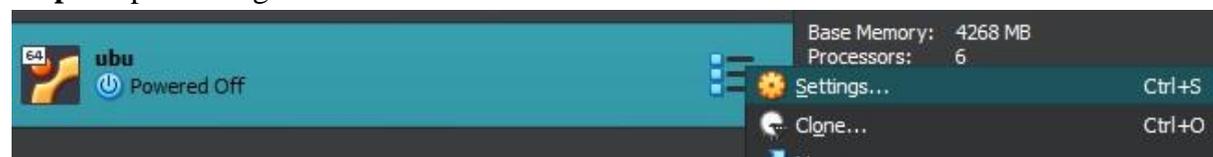
**Step 2:** In case of Virtual Machine use execute these command

“cd C:\Program Files\Oracle\VirtualBox“

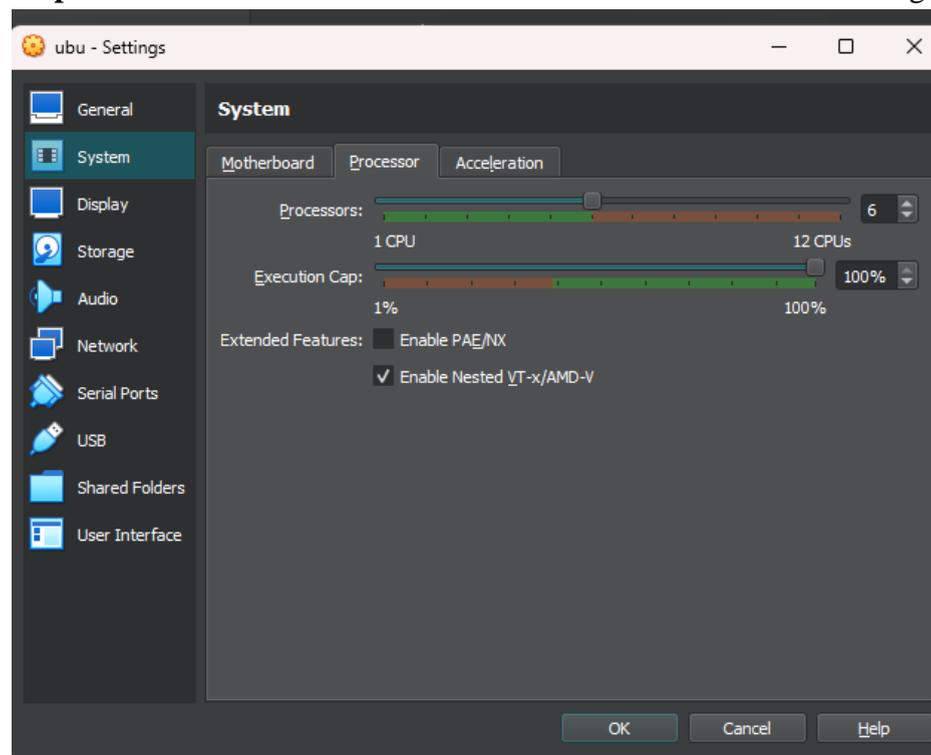
“VBoxManage modifyvm “ubu” --nested--hw-virt on”

```
C:\Users\mehta>cd C:\Program Files\Oracle\VirtualBox
C:\Program Files\Oracle\VirtualBox>VBoxManage modifyvm "ubu" --nested-hw-virt on
C:\Program Files\Oracle\VirtualBox>
```

**Step 3:** open settings



**Step 4:** Give minimum 6 Processors & Nested VTX Should ON through above command



**Step 5:** open terminal & run this command to check processors

```
$ egrep -c '(vm|svm)' /proc/cpuinfo
```

```
ubu@ubu-mac:~$ egrep -c '(vm|svm)' /proc/cpuinfo
6
```

**Step 6:** \$ kvm-ok

```
ubu@ubu-mac:~$ kvm-ok
INFO: Your CPU does not support KVM extensions
INFO: For more detailed results, you should run this as root
HINT:  sudo /usr/sbin/kvm-ok
```

**Step 7:** \$ sudo apt-get install -y qemu-kvm virt-manager

```
ubu@ubu-mac:~$ sudo apt-get install -y qemu-kvm virt-manager
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'qemu-system-x86' instead of 'qemu-kvm'
The following additional packages will be installed:
  dmeventd dmsetup gir1.2-ayatanaappindicator3-0.1 gir1.2-gtk-vnc-2.0
  gir1.2-gtksource-4 gir1.2-libosinfo-1.0 gir1.2-libvirt-glib-1.0
```

**Step 8:** \$ sudo systemctl enable --now libvirt

```
$ sudo systemctl start --now libvirt
```

```
$ sudo systemctl status --now libvirt
```

```
ubu@ubu-mac:~$ sudo systemctl enable --now libvirt
ubu@ubu-mac:~$ sudo systemctl start --now libvirt
ubu@ubu-mac:~$ sudo systemctl status --now libvirt
● libvirt.service - libvirt legacy monolithic daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirt.service; enabled; preset:▶
   Active: active (running) since Tue 2025-01-21 11:10:01 IST; 1min 16s ago
   TriggeredBy: ● libvirt-admin.socket
                 ● libvirt-ro.socket
                 ● libvirt.socket
   Docs: man:libvirt(8)
          https://libvirt.org/
   Main PID: 5460 (libvirt)
   Tasks: 22 (limit: 32768)
   Memory: 10.9M (peak: 12.2M)
```

**Step 9:** \$ sudo usermod -aG kvm \$USER

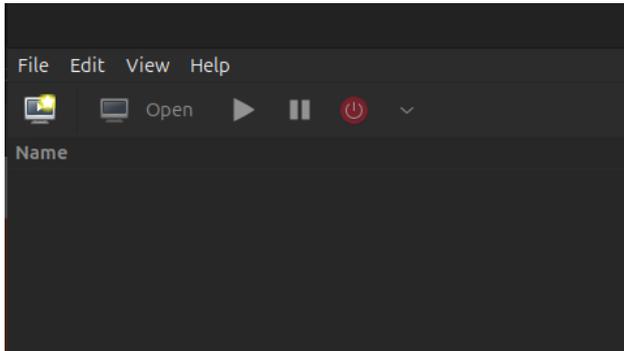
```
$ sudo usermod -aG libvirt $USER
```

```
ubu@ubu-mac:~$ sudo usermod -aG kvm $USER
ubu@ubu-mac:~$ sudo usermode -aG libvirt $USER
sudo: usermode: command not found
ubu@ubu-mac:~$ sudo usermod -aG libvirt $USER
```

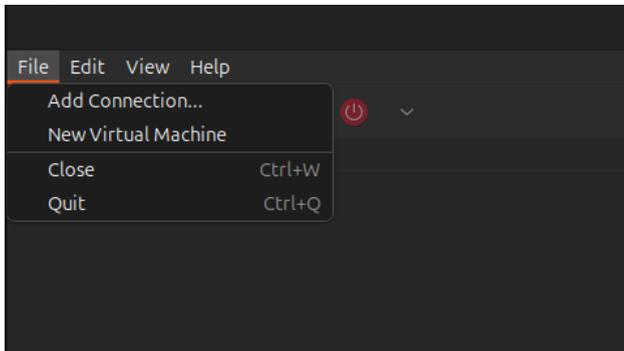
**Step 10:** Search Virtual Machine and open



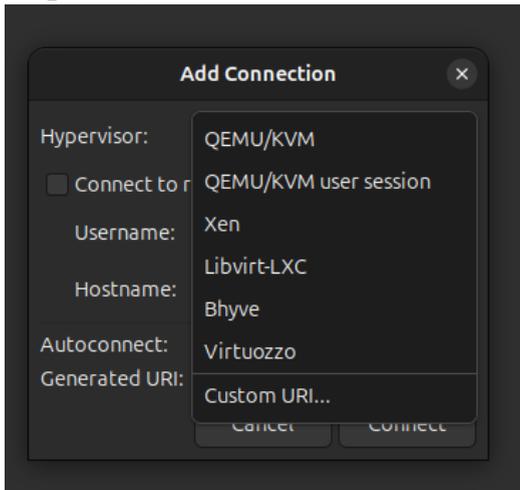
### Step 11: Click on File



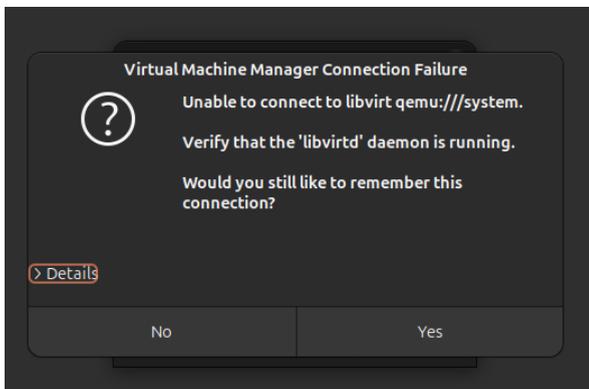
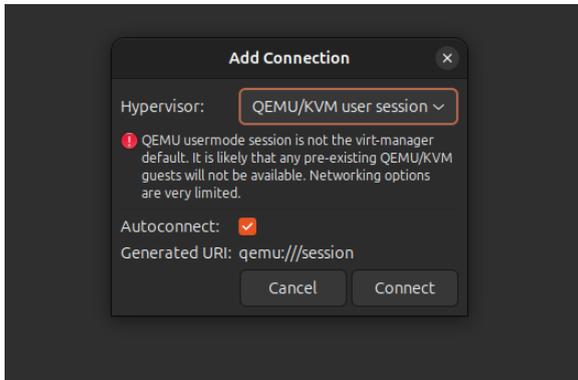
### Step 12: Add Connection



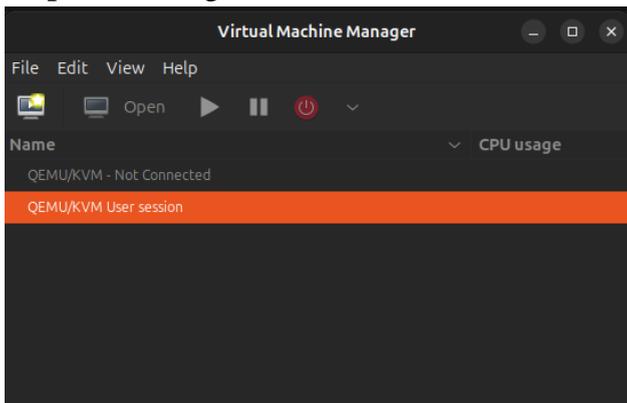
### Step 13: QEMU/KVM user session



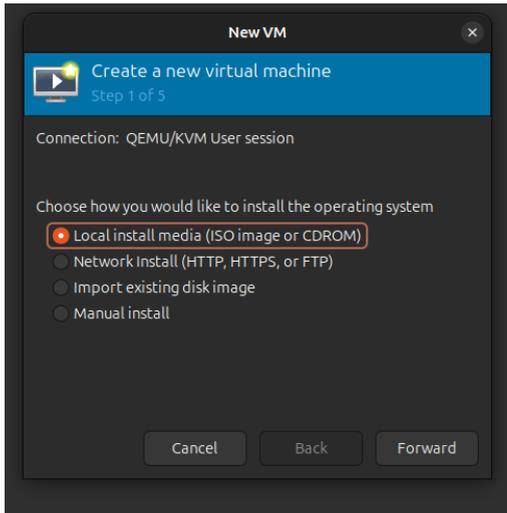
**Step 14:** click on “Connect” and then yes



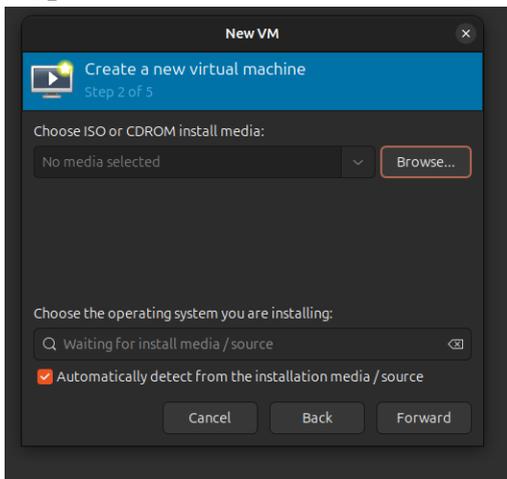
**Step 15:** now right click on “user session” and create new virtual machine



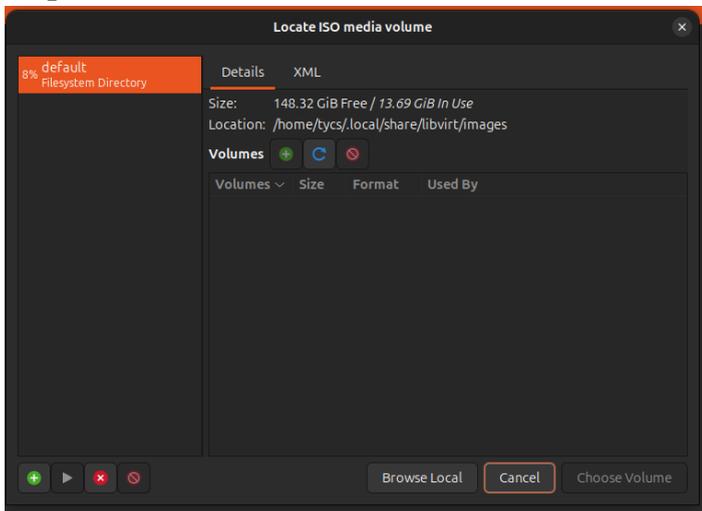
**Step 16:** Select 1<sup>st</sup> option and then forward



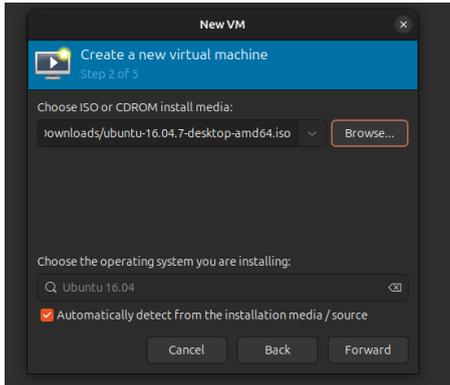
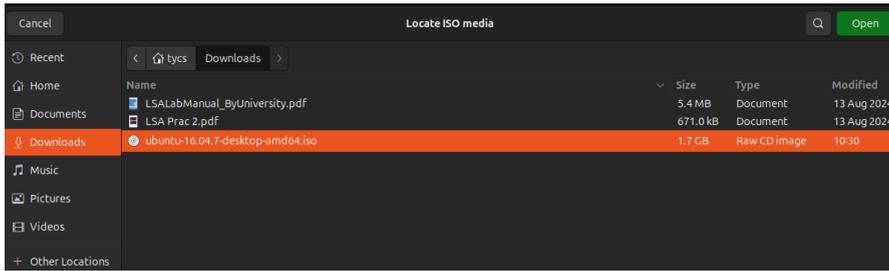
**Step 17:** Download ubuntu ISO File 16 & Click on Browse



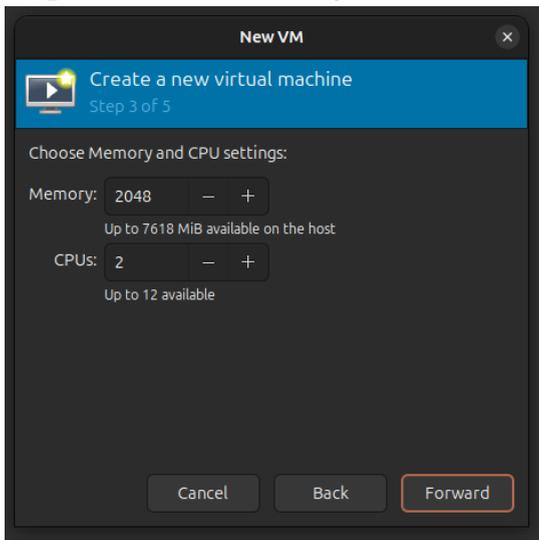
**Step 18:** Browse local



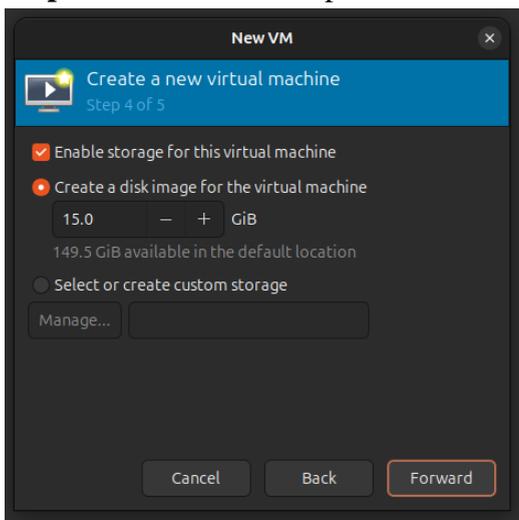
Select ISO file and then forward

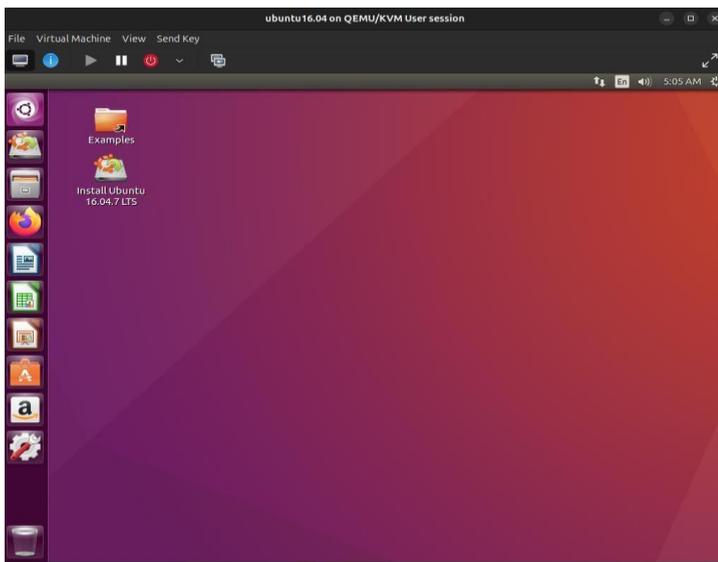
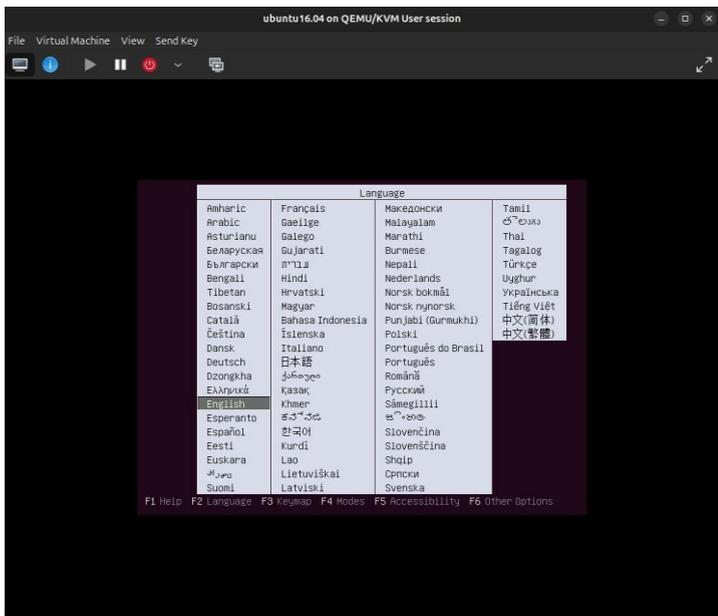
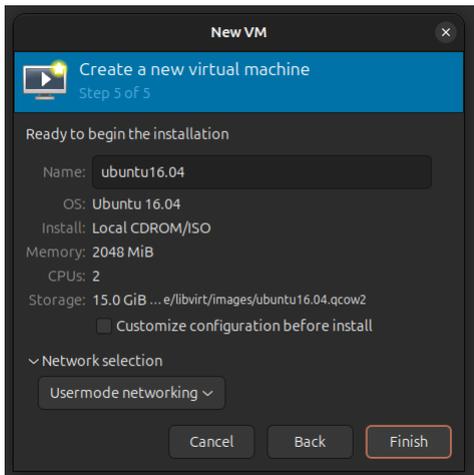


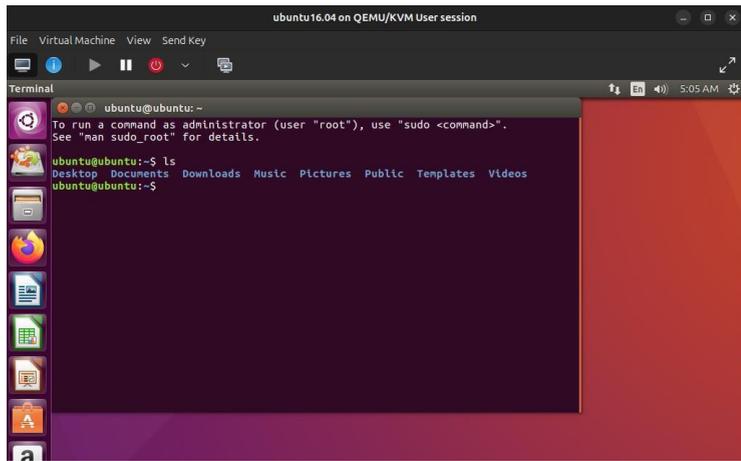
**Step 18:** Allocate Memory & CPUs and then forward



**Step 19:** Allocate Disk Space & then finish







# Practical No 6

By Chandan

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

**Solution:**

Using Flask

**Step 1:** Create app.py file & Run

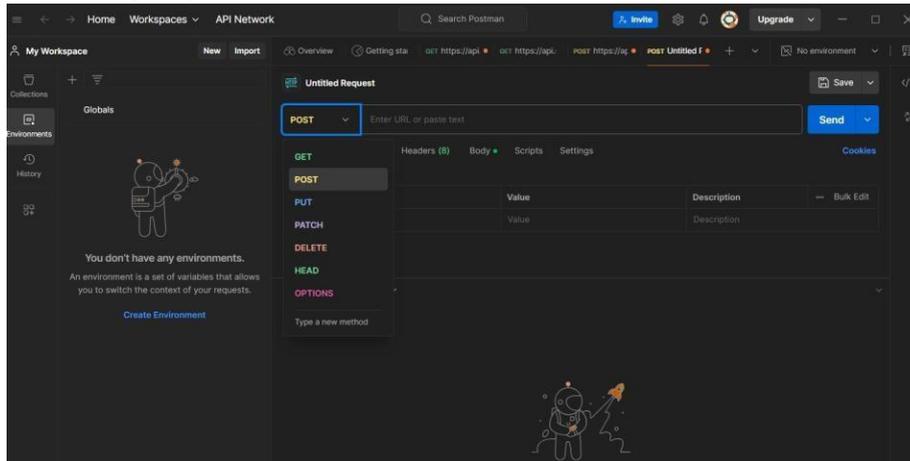
**app.py**

```
from flask import Flask, request,
send_from_directory import os
app = Flask(__name__)
# Namaste Bacho ! Uploads ke liye folder banane ka
UPLOAD_FOLDER = 'uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
# 1. File upload endpoint
@app.route('/upload',
methods=['POST']) def upload_file():
    file =
    request.files.get('file') if
    file:
        file_path = os.path.join(UPLOAD_FOLDER, file.filename)
        file.save(file_path)
        return {"message": f"File '{file.filename}' uploaded successfully!"}, 200
    return {"error": "No file uploaded"}, 400
# 2. File download endpoint
@app.route('/download/<filename>',
methods=['GET']) def download_file(filename):
    try:
        return send_from_directory(UPLOAD_FOLDER, filename, as_attachment=True)
    except FileNotFoundError: return
        {"error": "File not found"}, 404
if __name__ ==
    '__main__':
    app.run(debug=True)
```

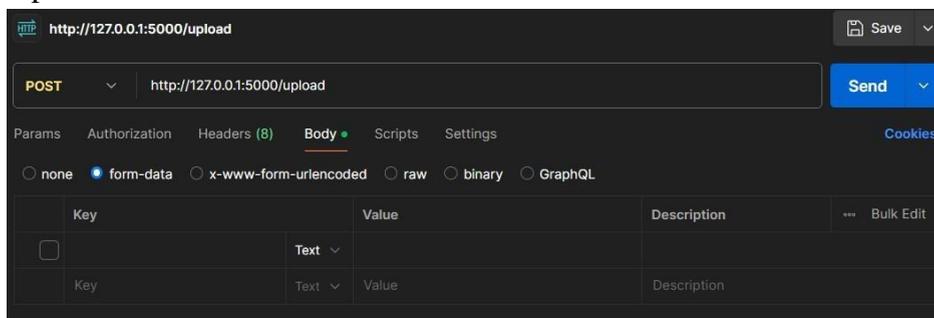
```
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 219-099-923
```

After running this code it will automatically create uploads folder

**Step 2:** Open postman and select method “POST”



**Step 3:** Enter url <http://127.0.0.1:5000/upload> & select Body & then select form-data and replace text With File



**Step 4:** Click on Values and + New file from local machine

## Step 5: After selecting file click on send button

The first screenshot shows the REST client interface for a POST request to `http://127.0.0.1:5000/upload`. The request body is configured as `form-data`. A file named `Screenshot 2024-05-03 22...` is selected for upload. The `Send` button is visible.

The second screenshot shows the same interface after the request is sent. The status bar indicates a `200 OK` response with a response time of `8 ms` and a body size of `247 B`. The response body is displayed in JSON format:

```
{
  "message": "File 'Screenshot 2024-05-03 225508.png' uploaded successfully!"
}
```

Now u can see uploaded file in uploads folder

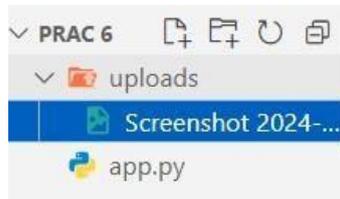
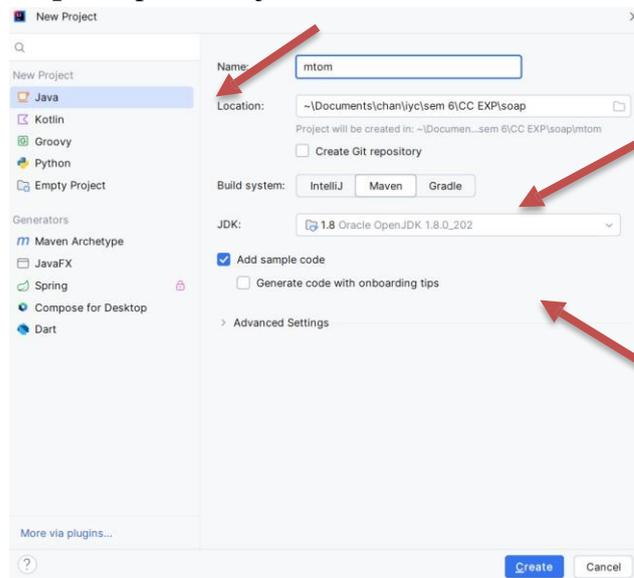


Image file

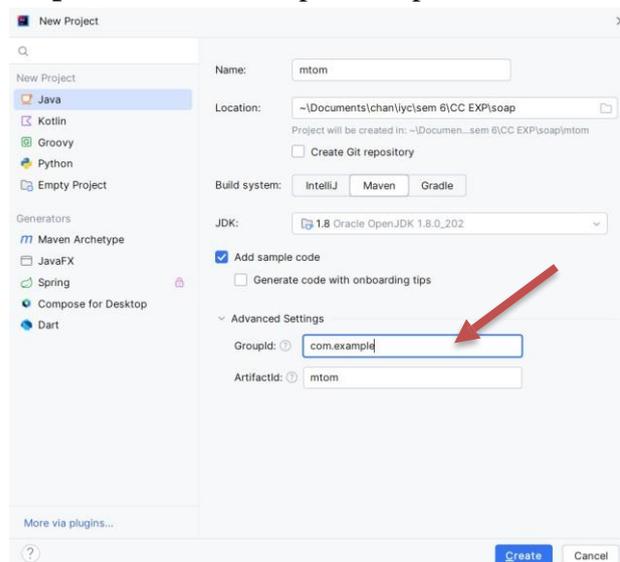
id	name	age	email
1	Charul Ma'am	25	charul@csmail.com
2	Sybal Ma'am	29	sybal@dias.com
3	Mudgul Ma'am	15	mudgul@iyc.com
4	Python Ma'am	35	python@py.com
5	Mehtha Sir	10	ashwin@mehta.com
	NULL	NULL	NULL

## Using Real MTOM Technique

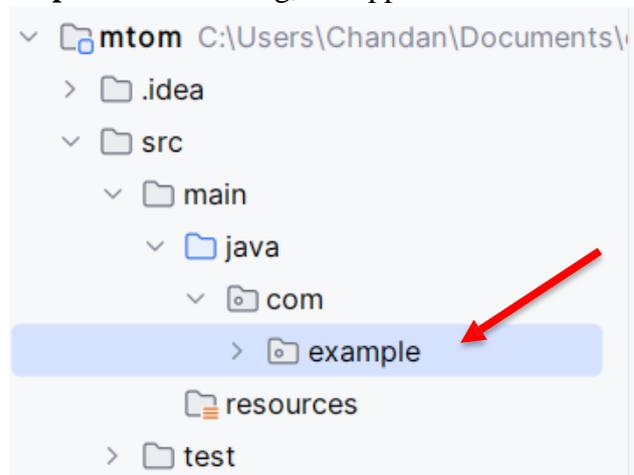
**Step 1:** Open IntelliJee and do the basic work as shown in the screenshots.



**Step 2:** Give the example Group ID.



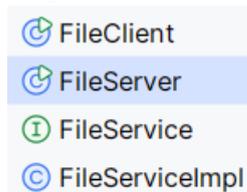
**Step 3:** After creating, the app will look like this.



**Step 4:** Now right-click on the example, and then left-click to create a new Java class. Name it as I'm telling you.



**Step 5:** You have to create all these classes one by one.



**Step 6:** Now look at the code and type it correctly from the files one by one.

#### **FileService.java**

```
package com.example;

import javax.ws.WebMethod;
import javax.ws.WebService;
import javax.xml.ws.soap.MTOM;

@MTOM // Enable MTOM
@WebService
public interface FileService {
    @WebMethod
    String uploadFile(byte[] data, String fileName);

    @WebMethod
    byte[] downloadFile(String fileName);
}
```

#### **FileServiceImpl.java**

```
package com.example;

import javax.ws.WebService;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

@WebService(endpointInterface = "com.example.FileService")
public class FileServiceImpl implements FileService {

    private static final String FILE_DIRECTORY =
"C:/Users/Chandan/Documents/chan/iyc/sem 6/CC
```

```
EXP/soap/mtom/src/main/java/com/example"; // Update path as required
```

```
@Override  
public String uploadFile(byte[] data, String fileName) {  
    try (FileOutputStream fos = new FileOutputStream(FILE_DIRECTORY +  
fileName)) {  
        fos.write(data);  
        return "File uploaded successfully: " + fileName;  
    } catch (IOException e) {  
        e.printStackTrace();  
        return "File upload failed: " + e.getMessage();  
    }  
}
```

```
@Override  
public byte[] downloadFile(String fileName) {  
    try {  
        return Files.readAllBytes(Paths.get(FILE_DIRECTORY + fileName));  
    } catch (IOException e) {  
        e.printStackTrace();  
        return null;  
    }  
}
```

### **FileServer.java**

```
package com.example;
```

```
import javax.xml.ws.Endpoint;
```

```
public class FileServer {  
    public static void main(String[] args) {  
        String serviceUrl = "http://localhost:8080/FileService";  
        Endpoint.publish(serviceUrl, new FileServiceImpl());  
        System.out.println("MTOM Service published at: " + serviceUrl);  
    }  
}
```

### **FileClient.java**

```
package com.example;
```

```
import javax.xml.namespace.QName;  
import javax.xml.ws.Service;  
import java.io.File;  
import java.io.FileOutputStream;  
import java.net.URL;  
import java.nio.file.Files;
```

```
public class FileClient {
```

```

public static void main(String[] args) throws Exception {
    // Connect to the published service
    URL wsdlUrl = new URL("http://localhost:8080/FileService?wsdl");
    QName qname = new QName("http://example.com/",
"FileServiceImplService");
    Service service = Service.create(wsdlUrl, qname);
    FileService fileService = service.getPort(FileService.class);

    // Upload a file
    File fileToUpload = new
File("C:/Users/Chandan/Pictures/Screenshots/Screenshot 2025-01-18 203338.png"); //
Adjust file path
    byte[] fileData = Files.readAllBytes(fileToUpload.toPath());
    String uploadResponse = fileService.uploadFile(fileData, "uploaded.jpg");
    System.out.println(uploadResponse);

    // Download a file
    byte[] downloadedData = fileService.downloadFile("uploaded.jpg");
    String downloadPath = "C:/Users/Chandan/Documents/chan/iyc/sem 6/CC
EXP/downloaded.jpg"; // Specify a full file path
    try (FileOutputStream fos = new FileOutputStream(downloadPath)) {
        fos.write(downloadedData);
        System.out.println("File downloaded successfully to: " + downloadPath);
    }
}
}
}

```

**Step 7:** After writing the code and copying-pasting it, first open the FileServer and run it. Click on the run button under the current file.



The screenshot shows an IDE window titled "FileServer.java". The code in the window is as follows:

```

1 package com.example;
2
3 import javax.xml.ws.Endpoint;
4
5 public class FileServer {
6     public static void main(String[] args) {
7         String serviceUrl = "http://localhost:8080/FileService";
8         Endpoint.publish(serviceUrl, new FileServiceImpl());
9         System.out.println("MTOM Service published at: " + serviceUrl);
10    }
11 }
12

```

In the top right corner of the IDE, there is a toolbar with a dropdown menu labeled "Current File", a green play button (run), a gear icon (debug), and a vertical ellipsis. A red arrow points to the green play button.

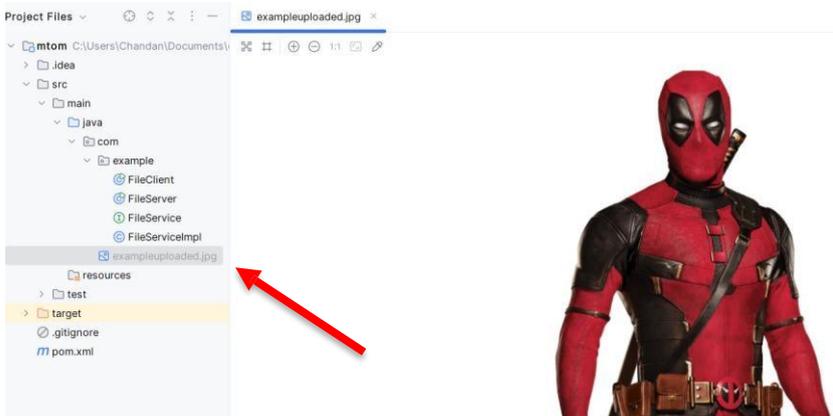
```
Run FileServer x FileClient x
"\"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
MTOM Service published at: http://localhost:8080/FileService
```

**Step 8:** Look kids, the run is successful. Open the server and client, then run them.

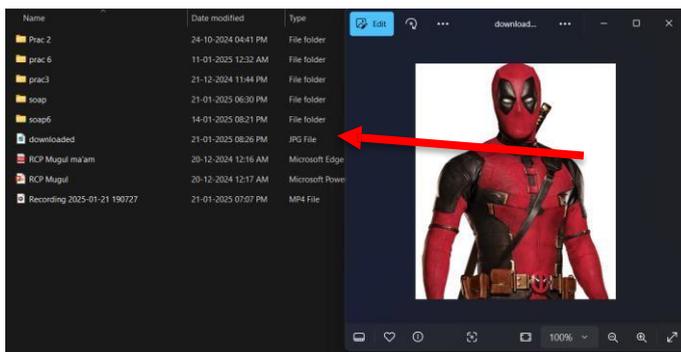
```
FileServer.java FileClient.java x
1 package com.example;
2
3 import javax.xml.namespace.QName;
4 import javax.xml.ws.Service;
5 import java.io.File;
6 import java.io.FileOutputStream;
7 import java.net.URL;
8 import java.nio.file.Files;
9
10 public class FileClient {
11     public static void main(String[] args) throws Exception {
```

**Step 9:** See kids, the file will be uploaded and downloaded using the MTOM technique.

```
Run FileServer x FileClient x
"\"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
File uploaded successfully: uploaded.jpg
File downloaded successfully to: C:/Users/Chandan/Documents/chan/ityc/sem 6/CC EXP/downloaded.jpg
```



- ❖ Uploaded file will appear here.
- ❖ Now, let's see the downloaded file.



# Practical No 7

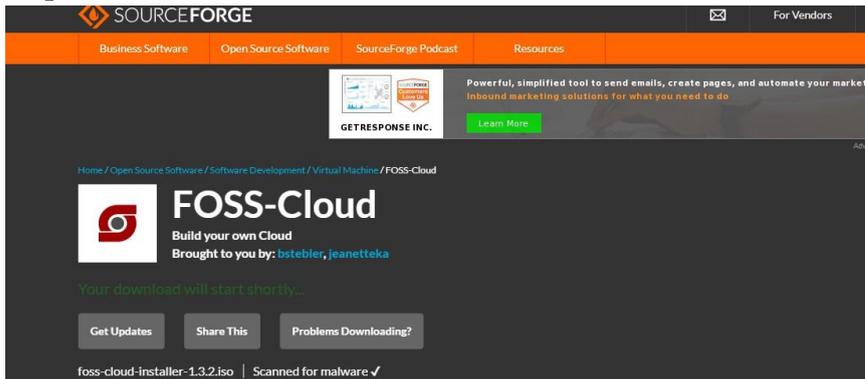
By Chandan

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

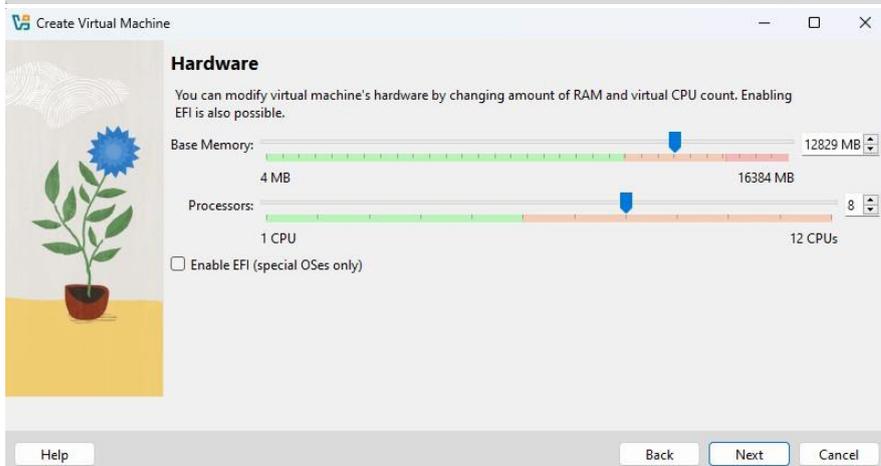
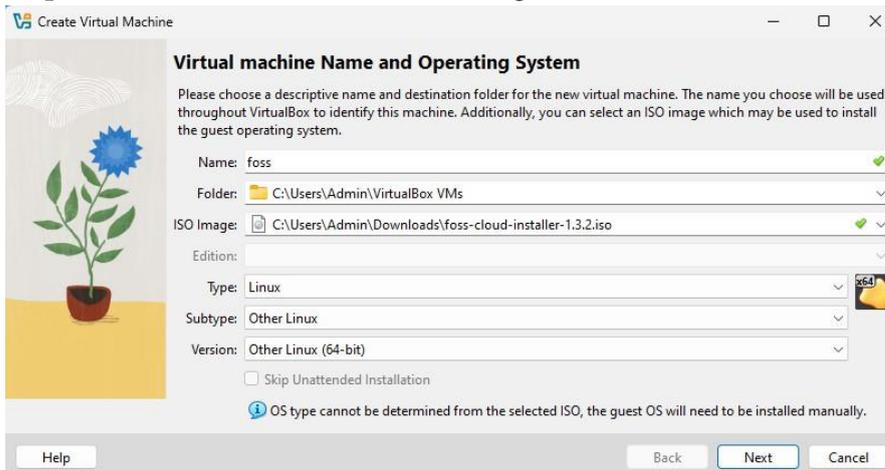
## Solution:

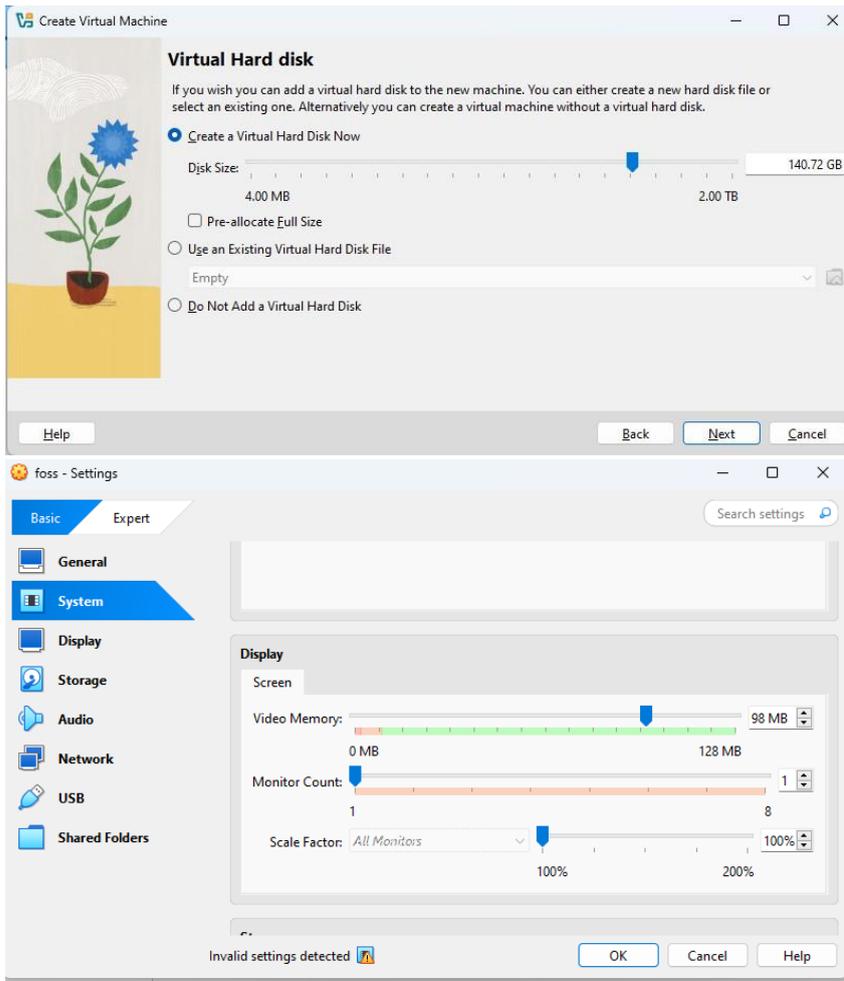
*Note: If your windows machine support then You can do this practical in windows otherwise install virtual box in Physical Dual Booted Ubuntu Machine and follow same steps except one step*

### Step 1: Download Foss Cloud ISO

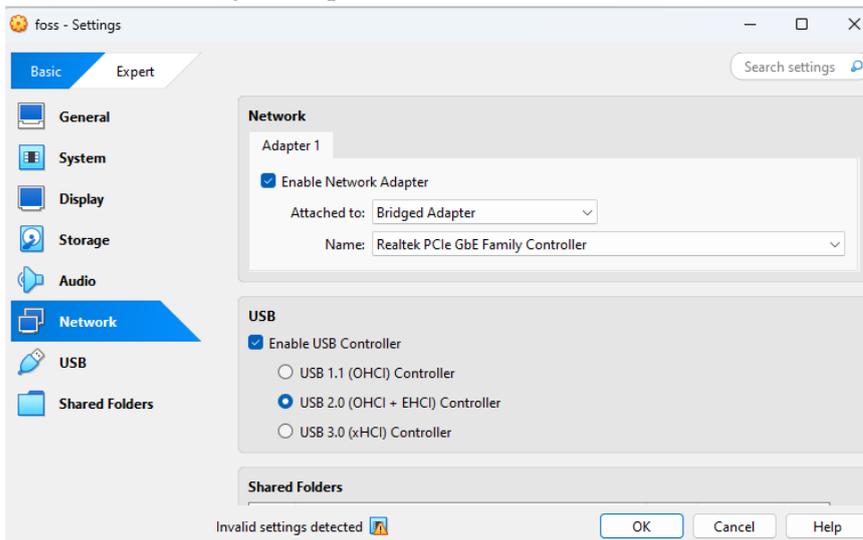


### Step 2: Create Virtual with the below configurations





### Step 3: Select Bridged Adapter



### Step 4: Turn off firewall

## Domain network

Networks at a workplace that are joined to a domain.

Have a question?

[Get help](#)

### Active domain networks

Not connected

Help improve Windows Security

[Give us feedback](#)

### Microsoft Defender Firewall

Helps protect your device while on a domain network.

Off

Change your privacy settings

View and change privacy settings for your Windows 11 Pro device.

[Privacy settings](#)

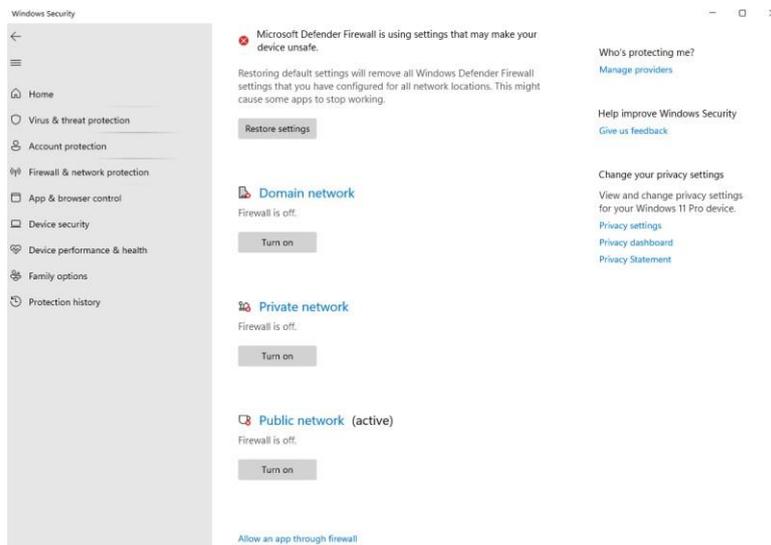
[Privacy dashboard](#)

[Privacy Statement](#)

### Incoming connections

Prevents incoming connections when on a domain network.

Blocks all incoming connections, including those in the list of allowed apps.



**For Linux:** Execute “ufw disable” in root user mode

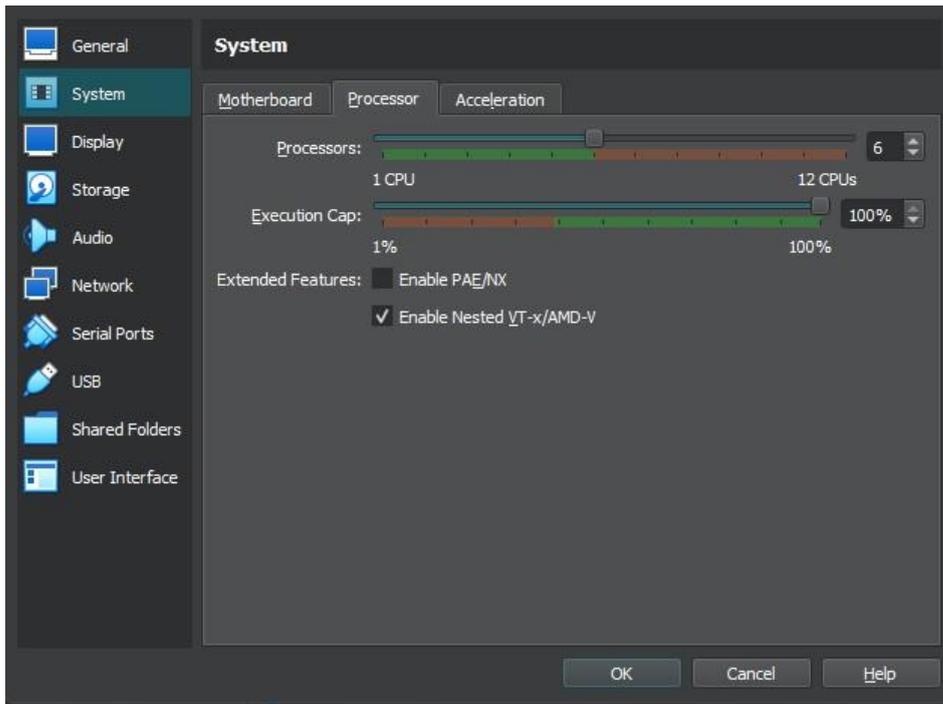
**Step 5:** Open Run cmd as administrative then execute the below commands

“cd C:\Program Files\Oracle\VirtualBox“

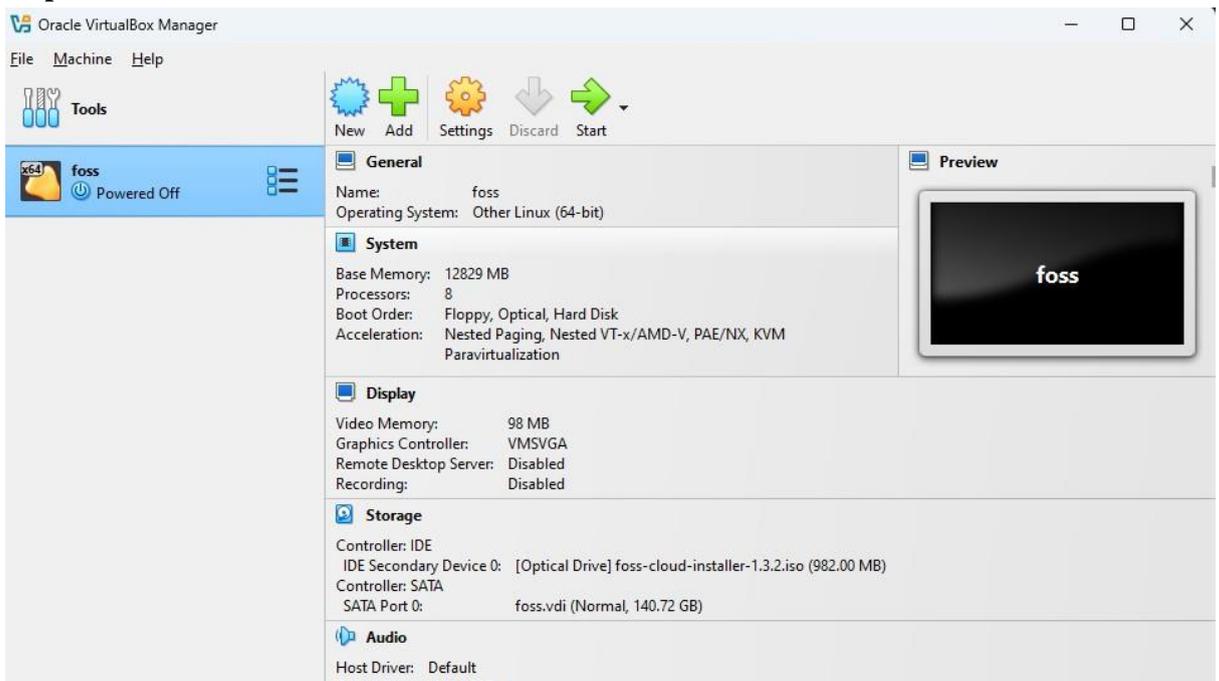
“VBoxManage modifyvm “foss” –nested—hw-virt on”

```
C:\Windows\System32>cd C:\Program Files\Oracle\VirtualBox
C:\Program Files\Oracle\VirtualBox>VBoxManage modifyvm "foss" --nested-hw-virt on
```

**For Linux:** You need to enable it manually by clicking on Enable Nested VT-x/AMD-V



### Step 6: Start the virtual Machine



### Step 7: Select 1<sup>st</sup> option

**FOSS-CLOUD INSTALLER, based on SYSTEM-RESCUE-CD 4.7.2**

- 1) FOSS-Cloud Installer: default boot options
  - 2) FOSS-Cloud Installer: all files cached to memory (docache)
  - 3) FOSS-Cloud Installer: framebuffer console in high resolution
  - 4) FOSS-Cloud Installer: do not ask for keyboard, use US keymap
  - 5) Boot an existing Linux system installed on the disk
  - 6) FOSS-Cloud Installer: alternative kernel with default boot
  - 7) FOSS-Cloud Installer: directly start the graphical environment
- 
- A) Run system tools from floppy disk image... >
  - B) Standard 32bit kernel (rescue32) with more choice... >
  - C) Standard 64bit kernel (rescue64) with more choice... >
  - D) Alternative 32bit kernel (altker32) with more choice... >
  - E) Alternative 64bit kernel (altker64) with more choice... >
- 
- \*) Boot from first hard disk
  - \*) Boot from second hard disk

Automatic boot in 86 seconds...

Press [TAB] to edit options or <F2>, <F3>, <F4>, <F5>, <F6>, <F7> for help

Boot standard kernel with default options (should always work). You should use this entry if you don't know which one to use. You can press [TAB] and add extra boot options after rescue32 or/and rescue64 if required

- Hit Enter

```
[ 25.150413] sr 2:0:0:0: Attached scsi generic sg1 type 5
[ 25.153127] Freeing unused kernel memory: 9764K (ffff81f3f000 - ffffff828c8000)
[ 25.153732] Write protecting the kernel read-only data: 14336k
[ 25.154987] Freeing unused kernel memory: 292K (ffff8800017b7000 - ffff880001800000)
[ 25.159777] Freeing unused kernel memory: 1992K (ffff880001c0e000 - ffff880001e00000)
>> Loading kernel modules...
>> Waiting 1 seconds...
>> Loading keymaps
Please select a keymap from the following list by typing in the appropriate
name or number. You should prefer the name to the number (for example
type 'fr' instead of '16'). Hit Enter for the default 'us' keymap.

  1 azerty   2 be       3 bg       4 br-a     5 br-l     6 by       7 cf
  8 croat   9 cz      10 de     11 dk     12 dvorak  13 es     14 et
 15 fi     16 fr     17 gr     18 hu     19 il     20 is     21 it
 22 jp     23 la     24 lt     25 mk     26 nl     27 no     28 pl
 29 pt     30 ro     31 ru     32 se     33 sg     34 sk-y   35 sk-z
 36 slovene 37 trf    39 ua     40 uk     41 us     42 wangbe 43 fr_CH
 44 speakup 45 cs_CZ  46 de_CH  47 sg-lat1 48 fr-bepo 49 colemak 50 de_neo

default choice (US keymap) will be used if no action within 20 seconds
<< Load keymap (Enter for default): _
```

- Yes



```

+-----+
| Installation Device Partitioning |
+-----+
Below is the existing partition layout of your selected device
Error: /dev/sda: unrecognised disk label
Model: ATA UBOX HARDDISK (scsi)
Disk /dev/sda: 151GB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:

All existing partitions have to be deleted in order to continue
THIS MEANS THAT ALL DATA ON THIS DISK WILL BE LOST
Do you want to continue?
yes or no?: yes

```

- Enter enp0s3 and then yes and then again yes for reboot

```

+-----+
| Network Device Selection |
+-----+
Please enter the device which you would like to use
Available ethernet devices: enp0s3
Device #0: _

```

```

+-----+
| Network Device Selection |
+-----+
Please enter the device which you would like to use
Available ethernet devices: enp0s3
Device #0: enp0s3

```

```

+-----+
| Network Configuration |
+-----+
Do you want to use automatic network configuration (via DHCP)?
yes or no?: yes_

```

```

+-----+
| Installation Complete |
+-----+
Congratulation! You have finished the installation of FOSS-Cloud
Now all you need to do is reboot the system and remove the CD-ROM

Do you want to reboot your system?
yes or no?: yes

```

### Step 8: Boot from first Hard disk

```

FOSS-CLOUD INSTALLER, based on SYSTEM-RESOLVE-CD 4.7.2

1) FOSS-Cloud Installer: default boot options
2) FOSS-Cloud Installer: all files cached to memory (docache)
3) FOSS-Cloud Installer: framebuffer console in high resolution
4) FOSS-Cloud Installer: do not ask for keyboard, use US keymap
5) Boot an existing Linux system installed on the disk
6) FOSS-Cloud Installer: alternative kernel with default boot
7) FOSS-Cloud Installer: directly start the graphical environment

A) Run system tools from floppy disk image... >
B) Standard 32bit kernel (rescue32) with more choice... >
C) Standard 64bit kernel (rescue64) with more choice... >
D) Alternative 32bit kernel (altker32) with more choice... >
E) Alternative 64bit kernel (altker64) with more choice... >

*) Boot from first hard disk
*) Boot from second hard disk

Press [TAB] to edit options or <F2>, <F3>, <F4>, <F5>, <F6>, <F7> for help
Boot local OS installed on first hard disk

```

## Step 9: Select FOSS-Cloud



```

:: Loading from sata: sata_promise sata_sil sata_sil24 sata_sou sata_via sata_mu sata_sx4 sata_sis sata_uli sata_usc sata_qst
or ahci libata ata_piix sata_mu sata_inic162x pdc_adma
:: Loading from scsi: sg aic79xx aic7xxx arcmsr BusLogic initio gdth sun53c8xx dmx3191d qla1280 dc395x atp870u mptbase mptscs
ih mptspi mptfc mptsas 3w-xxxx 3w-9xxx cciss hpsa DmC960 sx8 aacraid megaraid megaraid_nbox megaraid_mm megaraid_sas qla2xxx lpf
c scsi_transport_fc aic94xx
:: Loading from usb: ehci-pci ehci-hcd usb-storage uhci-hcd ohci-hcd xhci-hcd s1811-hcd
:: Loading from firewire:
:: Loading from waitscan:
:: Loading from dmraid: dm-mirror dm-crypt
:: Loading from mdadm: dm-mirror dm-crypt
:: Loading from fs: btrfs reiserfs fuse
:: Loading from net: e1000 tg3
:: Loading from iscsi: scsi_transport_iscsi libiscsi iscsi_tcp
:: Loading from crypto: sha256_generic aes-x86_64 aesni-intel xts
>> Determining root device...
>> Detected real_root=/dev/sda3
>> Mounting /dev/sda3 as root...
>> Using mount -t auto -o ro
>> Booting (initramfs)
INIT: version 2.88 booting

OpenRC 0.19.1 is starting up Gentoo Linux (x86_64)

* /proc is already mounted
* Mounting /run ...
* /run/openrc: creating directory
* /run/lock: creating directory
* /run/lock: correcting owner
* Caching service dependencies ... [ ok ]
* Remounting devtmpfs on /dev ... [ ok ]
* Mounting /dev/mqueue ... [ ok ]
* Mounting /dev/shm ... [ ok ]
* Mounting security filesystem ... [ ok ]
* Mounting debug filesystem ... [ ok ]
* Mounting fuse control filesystem ... [ ok ]
* Mounting cgroup filesystem ... [ ok ]
* Setting up tmpfiles.d entries for /dev ... [ ok ]
* Starting udev ... [ ok ]
* Generating a rule to create a /dev/root symlink ... [ ok ]
* Populating /dev with existing devices through uevents ... [ ok ]
* Waiting for uevents to be processed ... [ ok ]
* Setting system clock using the hardware clock (UTC) ... [ ok ]
```

After this IP Address should come



```
*****
This is localhost.unknown_domain (Linux x86_64 4.10.1-gentoo) 08:26:34

localhost login: root
Password:
localhost ~ # fc-node-configuration -n demo-system --password admin
-----+
| Demo-System Installation |
-----+
-----+
| Retrieving local network configuration ... |
-----+
Local network configuration retrieval ok!

Unpacking tarball /usr/share/foss-cloud/predefined-storage.tar.bz2, depending on it's size this action can take some minutes ...
```

```
Starting the daemon dhcpd ...
Executing: /etc/init.d/dhcpd start...
* /var/run/dhcp: creating directory
* /var/run/dhcp: correcting owner
* /var/lib/dhcp: correcting owner
* /var/lib/dhcp/dhcpd.leases: creating file
* /var/lib/dhcp/dhcpd.leases: correcting mode
* /var/lib/dhcp/dhcpd.leases: correcting owner
* Starting dhcpd ... [ ok ]
Started dhcpd successfully!

Adding the daemon dhcpd to the runlevel default ...
Executing: rc-update add dhcpd default ...
Added dhcpd successfully to runlevel default!

Directory /var/virtualization/backup created

Created backup directory symlink: /var/backup --> /var/virtualization/backup, if you want to change the backup directory you may
do so by modifying this symlink.
File /etc/local.d/50-foss-cloud-firstrun.start deleted

Congratulations, you have finished the installation and configuration of this Node!

localhost ~ #
```

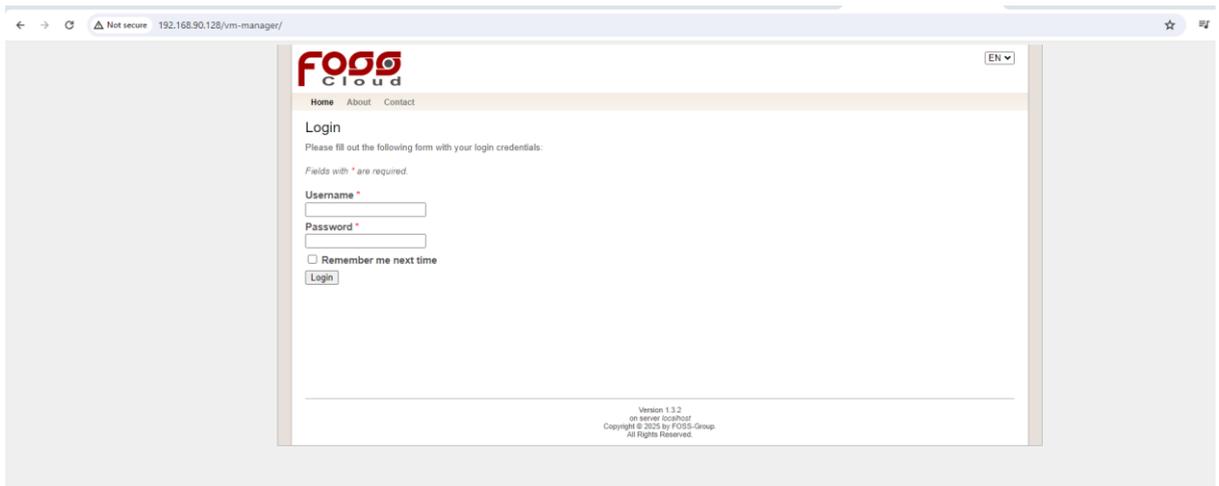
- Ifconfig and then note this ip address

```
localhost ~ # ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.90.128 netmask 255.255.255.0 broadcast 192.168.90.255
    ether 08:00:27:8b:a5:e6 txqueuelen 1000 (Ethernet)
    RX packets 567 bytes 45035 (43.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 126 bytes 11421 (11.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

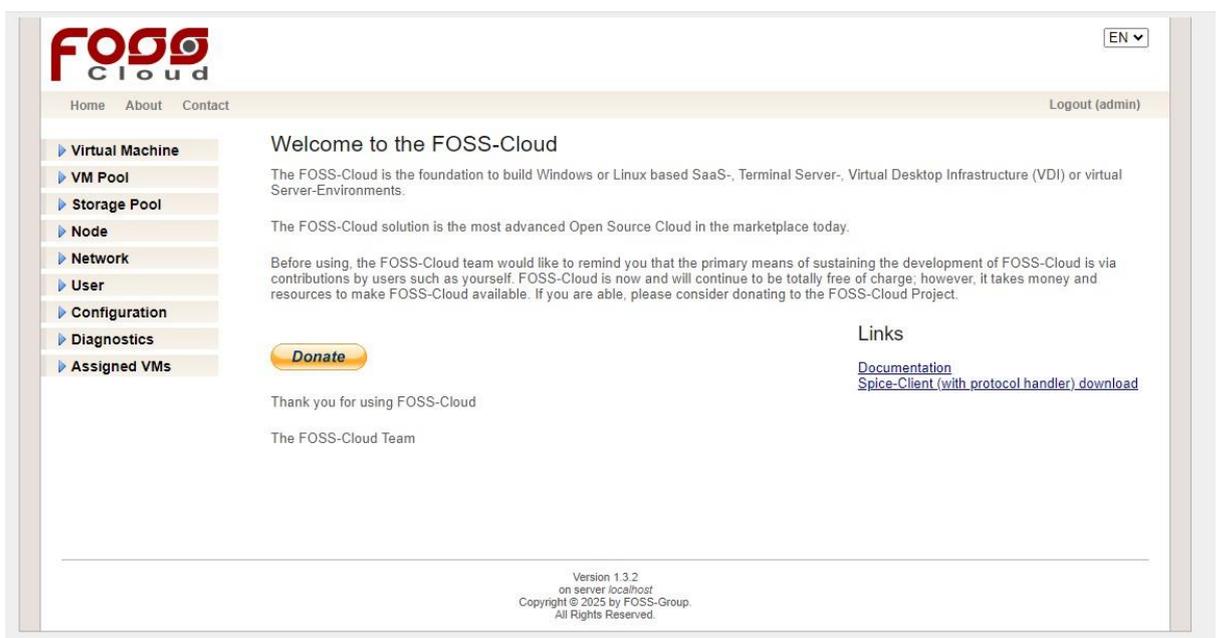
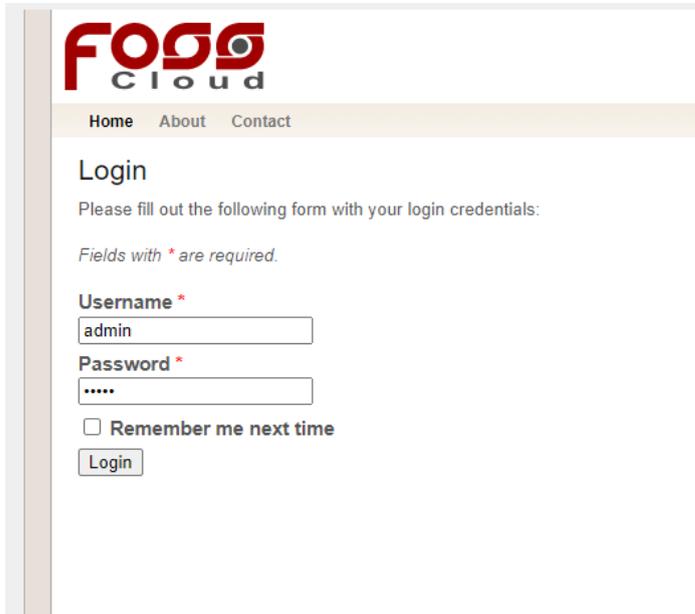
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 1510 bytes 378169 (369.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1510 bytes 378169 (369.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vbr0: flags=4419<UP,BROADCAST,RUNNING,PROMISC,MULTICAST> mtu 1500
    inet 172.31.255.1 netmask 255.255.255.0 broadcast 172.31.255.255
    ether 52:bb:52:d5:2c:4b txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

**Step 12:** Open Browser in your main machine and enter IP address of Foss Virtual Machine



Step 13: Login with username “admin” & password “admin”



# Practical No 8

By Chandan

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

**Solution:**

**Step 1:** Login Foss Cloud

The screenshot shows the FOSS-Cloud web interface. At the top left is the FOSS-Cloud logo. A navigation bar contains 'Home', 'About', and 'Contact'. On the right, there is a language dropdown set to 'EN' and a 'Logout (admin)' link. A left sidebar menu lists various system components: Virtual Machine, VM Pool, Storage Pool, Node, Network, User, Configuration, Diagnostics, and Assigned VMs. The main content area is titled 'Welcome to the FOSS-Cloud' and contains introductory text about the platform's purpose and a 'Donate' button. Below this, there are 'Links' for 'Documentation' and 'Spice-Client (with protocol handler) download'. The footer text reads: 'Version 1.3.2 on server localhost, Copyright © 2025 by FOSS-Group, All Rights Reserved.'

**Step 2:** Expand Virtual Machine Tab

This screenshot shows the same FOSS-Cloud interface as in Step 1, but with the 'Virtual Machine' menu item expanded. The expanded menu reveals sub-items: 'Persistent VMs', 'Dynamic VMs', 'VM Templates' (with a 'Create' button), 'Profiles' (with a 'Create' button and an 'Upload ISO File' button), 'VM Pool', 'Storage Pool', 'Node', 'Network', 'User', 'Configuration', 'Diagnostics', and 'Assigned VMs'. The main content area and footer remain identical to the previous screenshot.

**Step 3:** Upload ISO File of Ubuntu 16

**FOSS Cloud**

Home About Contact

**Virtual Machine**

Persistent VMs  
Dynamic VMs  
VM Templates  
Create  
Profiles  
Create  
Upload ISO File

**VM Pool**

**Storage Pool**

**Node**

**Network**

**User**

**Configuration**

**Diagnostics**

**Assigned VMs**

### Upload ISO File

*Fields with \* are required.*

[Alternative upload method](#)

**Iso File**  
Choose File No file chosen

**File Name**

Upload

Version 1.3.2  
on server local/host  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

- Select & upload

ubuntu-16.04.7-desktop-amd64	18-02-2025 13:12	Disc Image File	16,58,112 KB
foss-cloud-installer-1.3.2	18-02-2025 12:29	Disc Image File	10,05,568 KB

After uploading give File name

**FOSS Cloud**

Home About Contact

**Virtual Machine**

Persistent VMs  
Dynamic VMs  
VM Templates  
Create  
Profiles  
Create  
Upload ISO File

**VM Pool**

**Storage Pool**

**Node**

**Network**

**User**

**Configuration**

**Diagnostics**

**Assigned VMs**

### Upload ISO File

*Fields with \* are required.*

[Alternative upload method](#)

**Iso File**  
Choose File **ubuntu-16.0...p-amd64.iso**

**File Name**

Upload

**FOSS Cloud** EN

Home About Contact Logout (admin)

- Virtual Machine
  - Persistent VMs
  - Dynamic VMs
  - VM Templates
    - Create
  - Profiles
    - Create
    - Upload ISO File
- VM Pool
- Storage Pool
- Node
- Network
- User
- Configuration
- Diagnostics
- Assigned VMs

### Upload ISO File

*Fields with \* are required.*

[Alternative upload method](#)

**ISO File**  
 ubuntu-16.0...p-amd64.iso

**File Name**

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

**FOSS Cloud** EN

Home About Contact Logout (admin)

- Virtual Machine
  - Persistent VMs
  - Dynamic VMs
  - VM Templates
    - Create
    - Upload ISO File
  - Profiles
    - Create
- VM Pool
- Storage Pool
- Node
- Network
- User
- Configuration
- Diagnostics
- Assigned VMs

### Upload ISO File

*Fields with \* are required.*

[Alternative upload method](#)

**ISO File**  
 ubuntu-16.0...p-amd64.iso

Upload finished (1.58 GB) **100%** took 5 seconds

**File Name**

**Finished** x

Upload finished!

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

## Step 4: Create VM Profile

**FOSS Cloud**

Home About Contact

- Virtual Machine
  - Persistent VMs
  - Dynamic VMs
  - VM Templates
    - Create
  - Profiles
    - Create
    - Upload ISO File
- VM Pool
- Storage Pool
- Node
- Network
- User
- Configuration
- Diagnostics
- Assigned VMs

### Create VM Profile

*Fields with \* are required.*

**Step 1**  
Please select a profile first!

**BaseProfile**

linux

windows

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

- Now select BaseProfile [Your OS Windows or Linux] → x86\_64 → en-US then give Name ,Description, memory, Vol capacity, CPU & localtime then create

**FOSS Cloud** [EN]

Home About Contact Logout (admin)

**Virtual Machine**

- Persistent VMs
- Dynamic VMs
- VM Templates
- Create
- Profiles
  - Create
  - Upload ISO File
- VM Pool
- Storage Pool
- Node
- Network
- User
- Configuration
- Diagnostics
- Assigned VMs

### Create VM Profile

*Fields with \* are required.*

**Step I**  
Please select a profile first!

**BaseProfile**

- linux
  - windows
  - default
    - i686
    - x86\_64
      - multi
      - de-DE
      - de-AT
      - de-CH
      - en-US**
      - en-GB
      - fr-CH
      - fr-FR
      - it-CH
      - it-IT

**Step II**  
Override the default values if necessary!

**Isofile \***  
Ubuntu.iso

**Name \***  
Ubuntu VM

**Description \***  
Creating VM on Foss Cloud server

**Memory \***  
256 MB — 128 GB — 512 MB

**Volume Capacity \***  
10 GB — 2048 GB — 40 GB

**CPU \***  
1

**Clock Offset \***  
localtime

Create

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

**FOSS Cloud** [EN]

Home About Contact Logout (admin)

**Virtual Machine**

- Persistent VMs
- Dynamic VMs
- VM Templates
- Create
- Profiles
  - Create
  - Upload ISO File
- VM Pool
- Storage Pool
- Node
- Network
- User
- Configuration
- Diagnostics
- Assigned VMs

### Create VM Profile

*Fields with \* are required.*

**Step I**  
Please select a profile first!

**BaseProfile**

- linux
  - windows
  - default
    - i686
    - x86\_64
      - multi
      - de-DE
      - de-AT
      - de-CH
      - en-US**
      - en-GB
      - fr-CH
      - fr-FR
      - it-CH
      - it-IT

**Step II**  
Override the default values if necessary!

**Isofile \***  
Ubuntu.iso

**Name \***  
Ubuntu VM

**Description \***  
Creating VM on Foss Cloud server

**Memory \***  
256 MB — 128 GB — 2.25 GB

**Volume Capacity \***  
10 GB — 2048 GB — 88 GB

**CPU \***  
1

**Clock Offset \***  
localtime

Create

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

**FOSS Cloud** EN ▾

Home About Contact Logout (admin)

**Virtual Machine**

- Persistent VMs
- Dynamic VMs
- VM Templates
  - Create
  - Upload ISO File
- Profiles
  - Create
  - Upload ISO File

**VM Pool**

**Storage Pool**

**Node**

**Network**

**User**

**Configuration**

**Diagnostics**

**Assigned VMs**

## Create VM Profile

*Fields with \* are required.*

**Step I**  
Please select a profile first!

**BaseProfile**

- linux
  - windows
    - default
      - i686
      - x86\_64
        - multi
        - de-DE
        - de-AT
        - de-CH
        - en-US
        - en-GB
        - fr-CH
        - fr-FR
        - it-CH
        - it-IT

**Step II**  
Overwrite the default values if necessary!

**ISOfile \***  
Ubuntu.iso

**Name \***  
Ubuntu VM

**Description \***  
Creating VM on Foss Cloud server

**Memory \***  
256 MB  2.25 GB

**Volume Capacity \***  
10 GB  88 GB

**CPU \*** 1 ▾ **Clock Offset \*** localtime ▾

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

**FOSS Cloud** EN ▾

Home About Contact Logout (admin)

**Virtual Machine**

- Persistent VMs
- Dynamic VMs
- VM Templates
  - Create
  - Upload ISO File
- Profiles
  - Create
  - Upload ISO File

**VM Pool**

**Storage Pool**

**Node**

**Network**

**User**

**Configuration**

**Diagnostics**

**Assigned VMs**

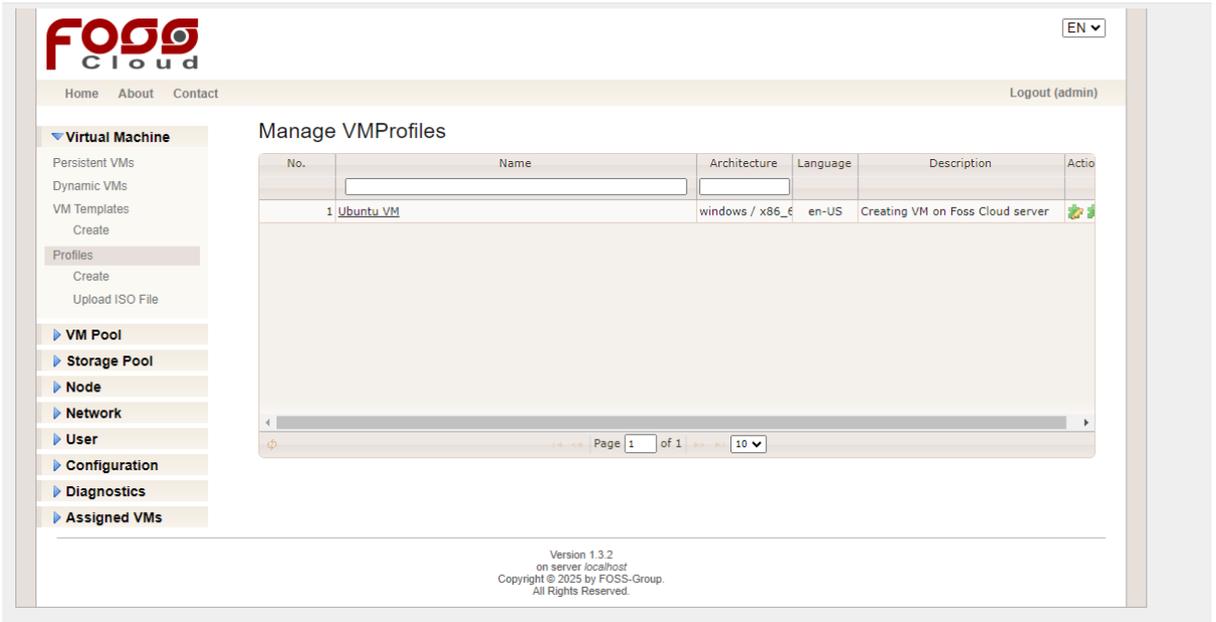
## Manage VMProfiles

No.	Name	Architecture	Language	Description	Action
1	Ubuntu VM	windows / x86_64	en-US	Creating VM on Foss Cloud server	

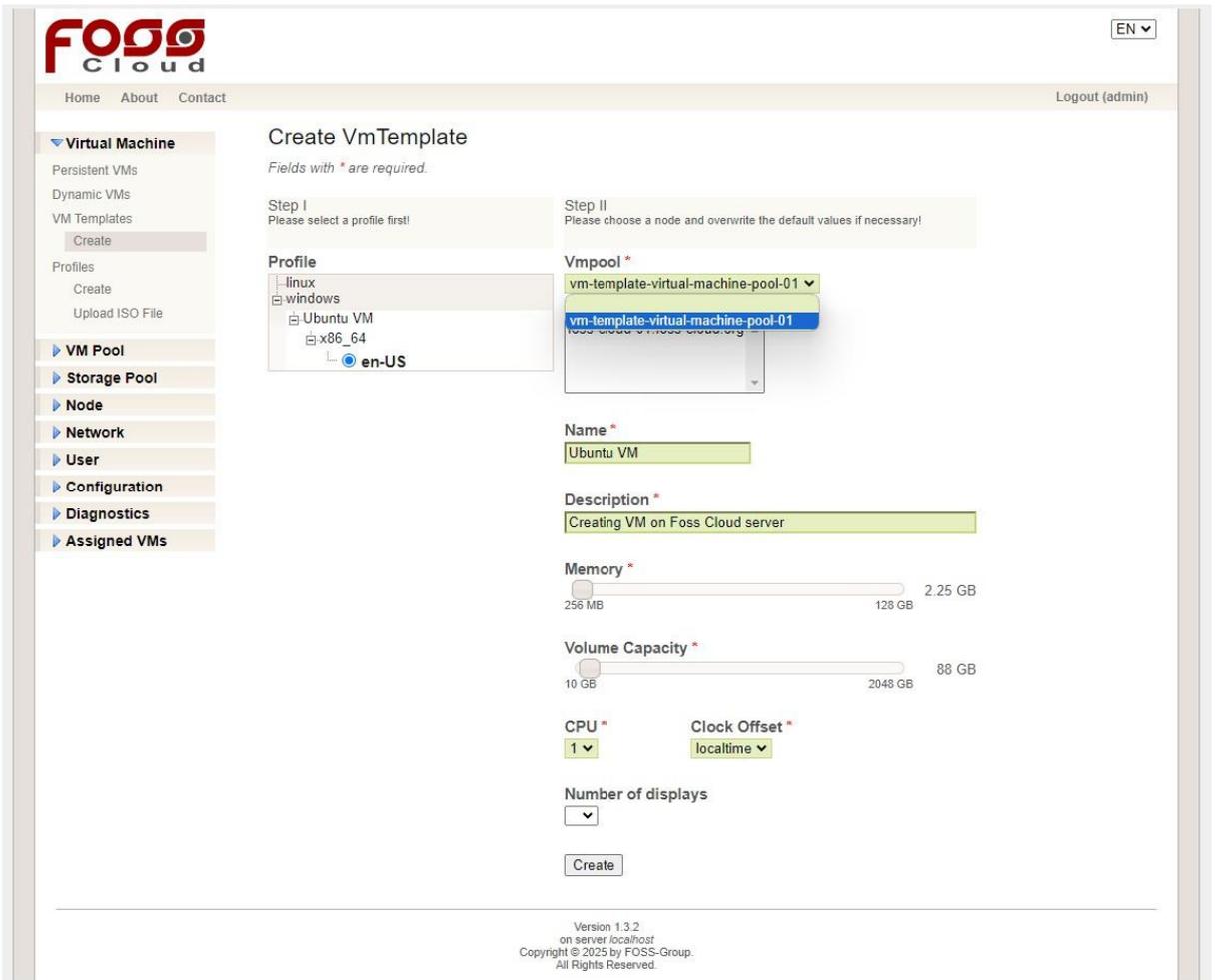
Check ISO Copy

🟢 Finished!

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.



**Step 5:** Now Navigate to VM Template then click on create after that select Profile [os Windows or Linux] → [VM Name] → x86\_64 → en-US and then select Vmpool & number of display 1 and create



**FOSS Cloud** EN

Home About Contact Logout (admin)

**Virtual Machine**

- Persistent VMs
- Dynamic VMs
- VM Templates
  - Create
- Profiles
  - Create
  - Upload ISO File
- VM Pool**
- Storage Pool
- Node
- Network
- User
- Configuration
- Diagnostics
- Assigned VMs

## Create VmTemplate

*Fields with \* are required.*

**Step I**  
Please select a profile first!

**Profile**

- linux
  - windows
    - Ubuntu VM
      - x86\_64
        - en-US

**Step II**  
Please choose a node and overwrite the default values if necessary!

**Vmpool \***  
vm-template-virtual-machine-pool-01

**Node \***  
foss-cloud-01.foss-cloud.org

**Name \***  
Ubuntu VM

**Description \***  
Creating VM on Foss Cloud server

**Memory \***  
256 MB  128 GB 2.25 GB

**Volume Capacity \***  
10 GB  2048 GB 88 GB

**CPU \*** 1 **Clock Offset \*** localtime

**Number of displays** 1

Create

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

**Step 6:** Click on Green Arrow → to start VM

**FOSS Cloud** EN

Home About Contact Logout (admin)

**Virtual Machine**

- Persistent VMs
- Dynamic VMs
- VM Templates
  - Create
- Profiles
  - Create
  - Upload ISO File
- VM Pool**
- Storage Pool
- Node
- Network
- User
- Configuration
- Diagnostics
- Assigned VMs

## Manage VM Templates

Vm Pool: vm-template-virtual-machine-pool-01

No.	DisplayName	Status	Run Action	Memory	Node	Action
+	1 Ubuntu VM	stopped	→ ↓ × ↻	---	foss-cloud-01.foss-cloud.org	

Page 1 of 1  Refresh

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

**FOSS Cloud** EN

Home About Contact Logout (admin)

**Virtual Machine**

- Persistent VMs
- Dynamic VMs
- VM Templates**
  - Create
- Profiles
  - Create
  - Upload ISO File
- VM Pool
- Storage Pool
- Node
- Network
- User
- Configuration
- Diagnostics
- Assigned VMs

**Links**

- Download Spice Client

### Manage VM Templates

Vm Pool: vm-template-virtual-machine-pool-01

No.	DisplayName	Status	Run Action	Memory	Node	Action
+	1 Ubuntu VM	running	→ ↓ × ↺	2.25 GB / 2.25 GB	foss-cloud-01.foss-cloud.org	🌐 🖥️ 📄

Page 1 of 1 | 10 | Refresh 10

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

**FOSS Cloud** EN

Home About Contact Logout (admin)

**Virtual Machine**

- Persistent VMs
- Dynamic VMs
- VM Templates**
  - Create
- Profiles
  - Create
  - Upload ISO File
- VM Pool
- Storage Pool
- Node
- Network
- User
- Configuration
- Diagnostics
- Assigned VMs

**Links**

- Download Spice Client

### Manage VM Templates

Vm Pool: vm-template-virtual-machine-pool-01

No.	DisplayName	Status	Run Action	Memory	Node	Action
+	1 Ubuntu VM	running	→ ↓ × ↺	2.25 GB / 2.25 GB	foss-cloud-01.foss-cloud.org	🌐 🖥️ 📄

Page 1 of 1 | 10 | Refresh 10

Open remote-viewer?

http://192.168.90.128 wants to open this application.

Version 1.3.2  
on server localhost  
Copyright © 2025 by FOSS-Group.  
All Rights Reserved.

# Practical No 9

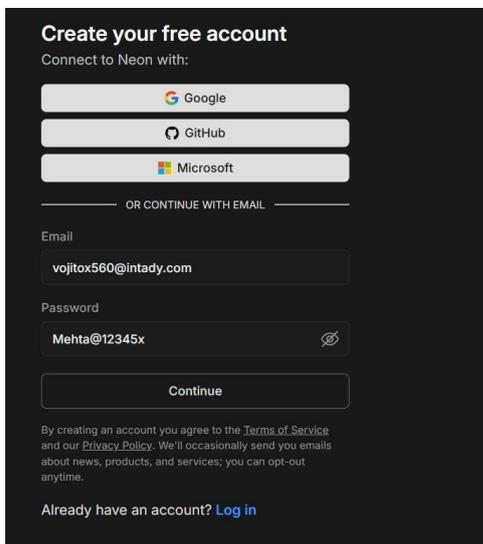
By Chandan

**Aim:** 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

## Solution:

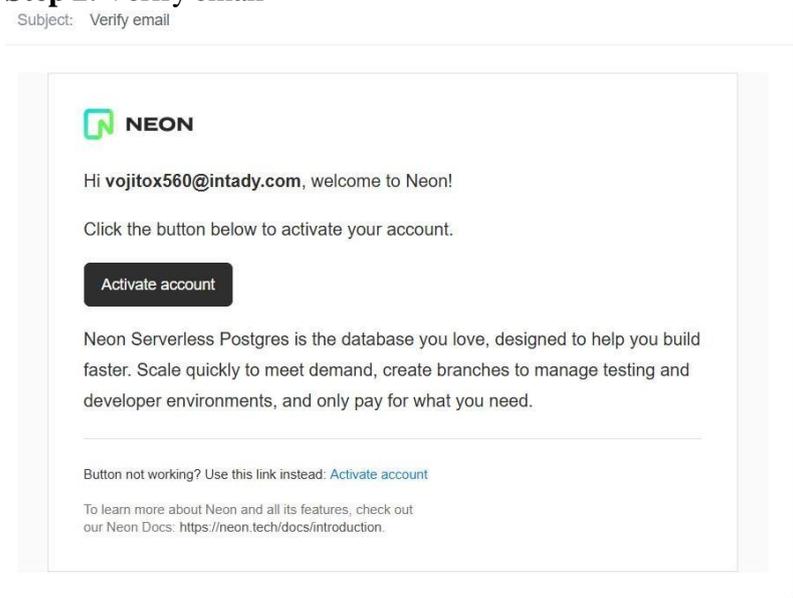
Note: AWS is paid so there we using its kind of 1 type of alternative neon DB which uses AWS Cloud service and give us some free storage which built on AWS

### Step 1: Create a account neon db

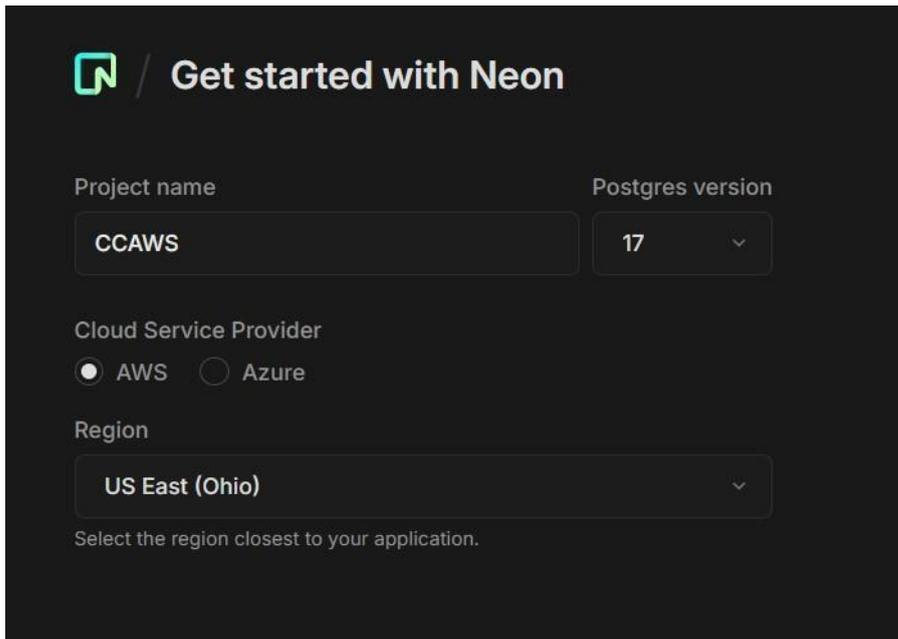


### Step 2: Verify email

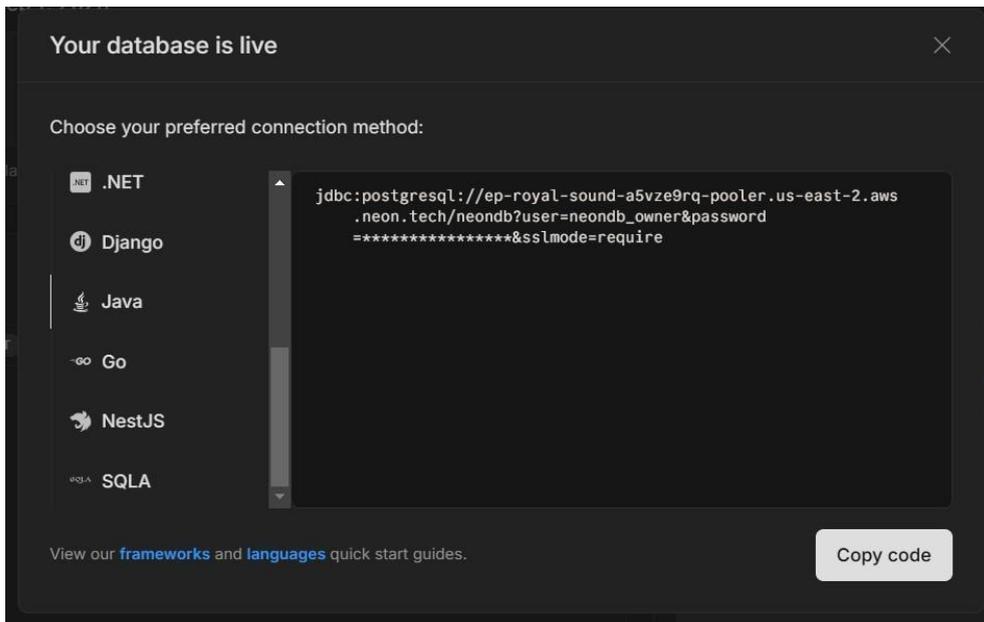
Subject: Verify email



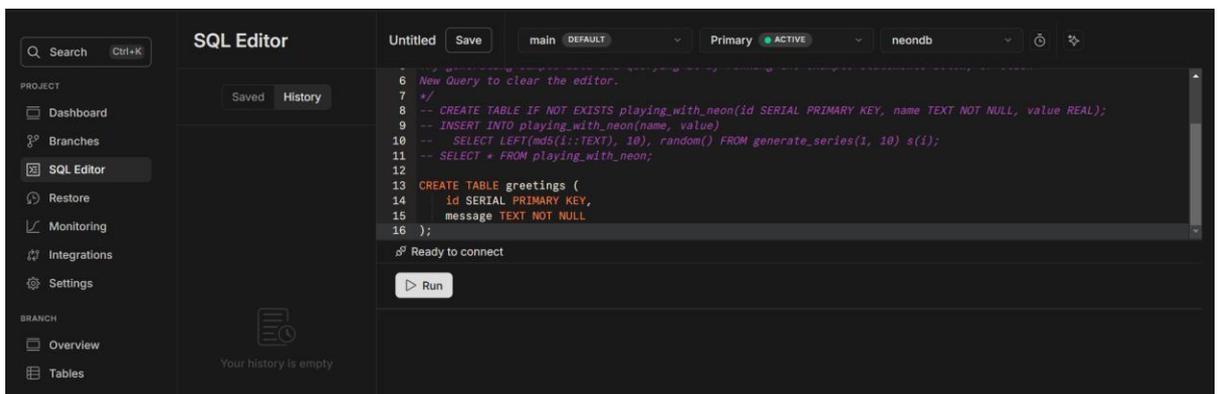
### Step 3: Chose Cloud Service Provider “AWS”



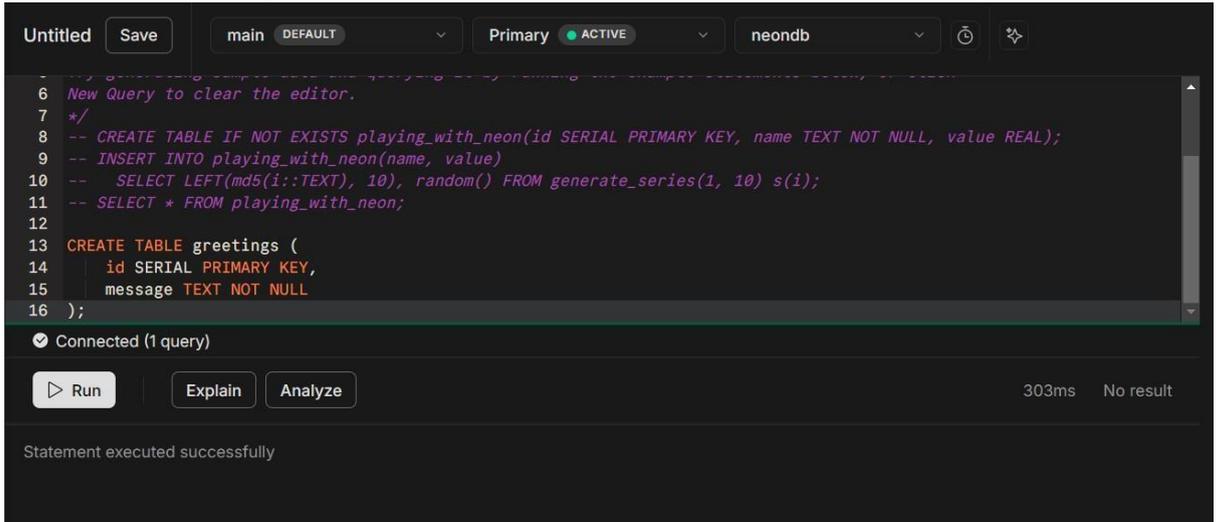
Now we got our connection url



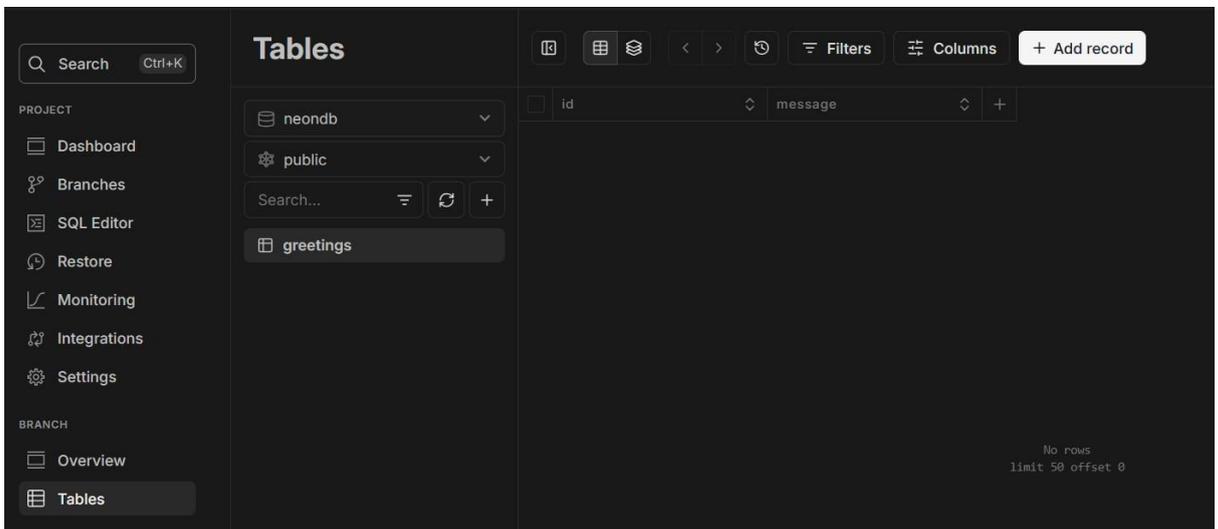
**Step 4:** Go to SQL Editor and create a Table by executing command which is mentioned in screenshot



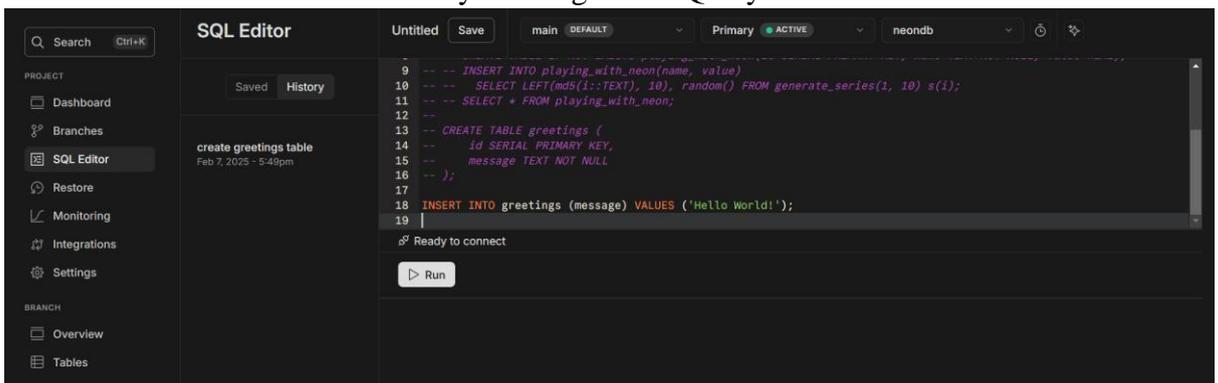
- After writing click on Run

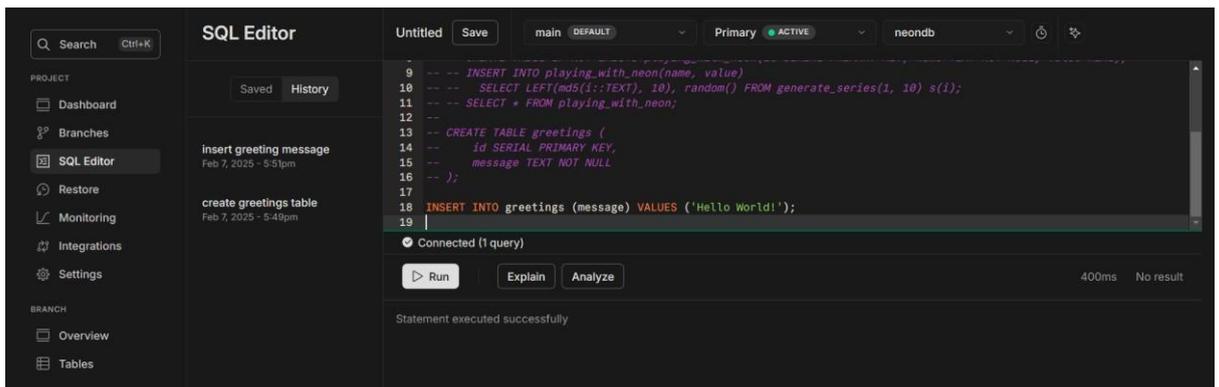


- Now table created

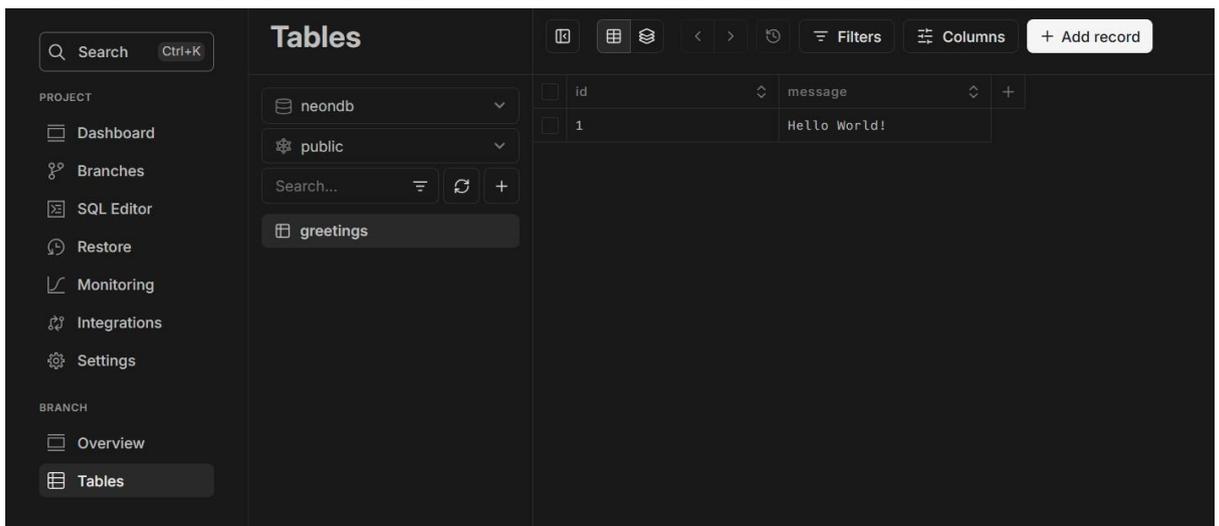


- Now inset a value in able by entering below Query in Screenshot and then run





- Now Value is inserted into table now we will fetch that value in console



**Step 5:** Create a python file and run

**Neondb.py**

```
import psycopg2
```

```
# Replace these with your Neon database credentials
```

```
DB_HOST = "ep-royal-sound-a5vze9rq-pooler.us-east-2.aws.neon.tech"
```

```
DB_NAME = "neondb"
```

```
DB_USER = "neondb_owner"
```

```
DB_PASS = "npg_Ztv1oYdsI3Sq"
```

```
try:
```

```
    # Connect to the Neon database
```

```
    connection = psycopg2.connect(
```

```
        host=DB_HOST,
```

```
        database=DB_NAME,
```

```
        user=DB_USER,
```

```
        password=DB_PASS
```

```
    )
```

```
    # Create a cursor object
```

```
cursor = connection.cursor()

# Query to fetch "Hello World" from the database
cursor.execute("SELECT message FROM greetings WHERE id =
1;") result = cursor.fetchone()

# Print the result
print("Fetched from Neon DB:", result[0])

except Exception as e: print("Error
connecting to the database:", e)
finally: if connection:
cursor.close()
connection.close()
```

**Output:**

```
Fetched from Neon DB: Hello world!
```

# Practical No 10

## Aim:

Implementation of Openstack with user and private network creation.

## Solution:

**Step 1:** sudo snap install microstack --beta

```
cs@cs-iyk:~$ sudo snap install microstack --beta
[sudo] password for cs:
microstack (beta) ussuri from Canonical✓ installed
cs@cs-iyk:~$
```

**Step 2:** snap list microstack

```
cs@cs-iyk:~$ snap list microstack
Name          Version Rev Tracking      Publisher  Notes
microstack    ussuri  245  latest/beta  canonical✓ -
cs@cs-iyk:~$
```

**Step 3:** sudo microstack init --auto --control

```
cs@cs-iyk:~$ sudo microstack init --auto --control
[sudo] password for cs:
Sorry, try again.
[sudo] password for cs:
2025-02-15 08:46:46,852 - microstack_init - INFO - Configuring clustering ...
2025-02-15 08:46:47,297 - microstack_init - INFO - Setting up as a control node.
2025-02-15 08:46:50,064 - microstack_init - INFO - Generating TLS Certificate and Key
2025-02-15 08:46:51,540 - microstack_init - INFO - Configuring networking ...
2025-02-15 08:47:03,491 - microstack_init - INFO - Opening horizon dashboard up to *
2025-02-15 08:47:04,268 - microstack_init - INFO - Waiting for RabbitMQ to start ...
Waiting for 192.168.90.133:5672
2025-02-15 08:47:05,566 - microstack_init - INFO - RabbitMQ started!
2025-02-15 08:47:05,566 - microstack_init - INFO - Configuring RabbitMQ ...
2025-02-15 08:47:07,319 - microstack_init - INFO - RabbitMQ Configured!
2025-02-15 08:47:07,398 - microstack_init - INFO - Waiting for MySQL server to start ...
Waiting for 192.168.90.133:3306
2025-02-15 08:47:08,548 - microstack_init - INFO - Mysql server started! Creating databases ...
2025-02-15 08:47:13,574 - microstack_init - INFO - Configuring Keystone Fernet Keys ...
2025-02-15 08:47:21,693 - microstack_init - INFO - Bootstrapping Keystone ...
2025-02-15 08:47:24,827 - microstack_init - INFO - Creating service project ...
2025-02-15 08:47:26,403 - microstack_init - INFO - Keystone configured!
2025-02-15 08:47:26,474 - microstack_init - INFO - Configuring the Placement service...
2025-02-15 08:47:29,642 - microstack_init - INFO - Running Placement DB migrations...
2025-02-15 08:47:32,526 - microstack_init - INFO - Configuring nova control plane services ...
2025-02-15 08:47:34,159 - microstack_init - INFO - Running Nova API DB migrations (this may take a lot of time)...
2025-02-15 08:47:43,085 - microstack_init - INFO - Running Nova DB migrations (this may take a lot of time)...
Waiting for 192.168.90.133:8774
2025-02-15 08:48:01,500 - microstack_init - INFO - Creating default flavors...
2025-02-15 08:48:12,554 - microstack_init - INFO - Configuring nova compute hypervisor ...
2025-02-15 08:48:12,554 - microstack_init - INFO - Checking virtualization extensions presence on the host
2025-02-15 08:48:12,616 - microstack_init - INFO - Hardware virtualization is supported - KVM will be used for Nova instances
2025-02-15 08:48:17,119 - microstack_init - INFO - Configuring the Spice HTML5 console service...
2025-02-15 08:48:18,285 - microstack_init - INFO - Configuring Neutron
Waiting for 192.168.90.133:9696
2025-02-15 08:49:04,787 - microstack_init - INFO - Configuring Glance ...
Waiting for 192.168.90.133:9292
2025-02-15 08:50:44,126 - microstack_init - INFO - Adding cirros image ...
2025-02-15 08:50:47,791 - microstack_init - INFO - Creating security group rules ...
2025-02-15 08:50:55,580 - microstack_init - INFO - Configuring the Cinder services...
2025-02-15 08:51:34,126 - microstack_init - INFO - Running Cinder DB migrations...
2025-02-15 08:53:21,204 - microstack_init - INFO - restarting Libvirt and virtlogd ...
2025-02-15 08:53:47,224 - microstack_init - INFO - Complete. Marked microstack as initialized!
```

**Step 4:** microstack.openstack --version

```
cs@cs-iyk:~$ microstack.openstack --version
openstack 5.2.0
```

**Step 5:** sudo snap get microstack config.credentials.keystone-password

```
cs@cs-1yc:~$ sudo snap get microstack config.credentials.keystone-password
mU8blRngAvs6uJLb7BBMvN4bZVg5thHT
```

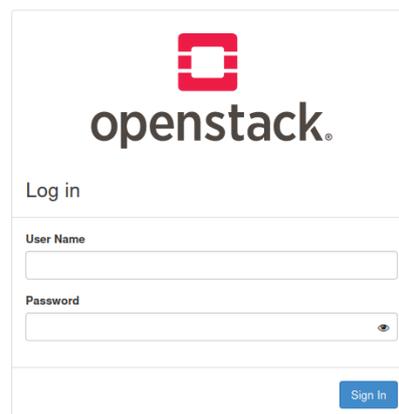
**Step 6:** ip a

```
cs@cs-1yc:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 1c:69:7a:ec:38:d8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.90.133/24 brd 192.168.90.255 scope global dynamic noprefixroute enp1s0
        valid_lft 5523sec preferred_lft 5523sec
    inet6 fe80::1e69:7aff:feec:38d8/64 scope link
        valid_lft forever preferred_lft forever
3: wlo1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether f4:c8:8a:b0:48:3e brd ff:ff:ff:ff:ff:ff
    altname wlp0s20f3
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:ed:5d:12 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
        valid_lft forever preferred_lft forever
5: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether ca:38:26:2c:0c:74 brd ff:ff:ff:ff:ff:ff
6: br-ex: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether aa:3f:51:6c:16:44 brd ff:ff:ff:ff:ff:ff
    inet 10.20.20.1/24 scope global br-ex
        valid_lft forever preferred_lft forever
    inet6 fe80::a83f:51ff:fe6c:1644/64 scope link
        valid_lft forever preferred_lft forever
7: br-int: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 2e:9b:c8:ae:70:49 brd ff:ff:ff:ff:ff:ff
```

**Step 7:** now open Browser and enter <https://yourip>

<https://192.168.90.133>

https://192.168.90.133/auth/login/?next=/



The image shows a web browser window displaying the OpenStack login page. At the top, there is the OpenStack logo (a red square with a white 'O') and the text 'openstack.'. Below the logo, the text 'Log in' is displayed. There are two input fields: 'User Name' and 'Password'. The 'Password' field has a small eye icon to its right, indicating a toggle for password visibility. At the bottom right of the form, there is a blue button labeled 'Sign In'.

**Step 8:** Login user name: admin & password from them terminal which got through previous command



openstack.

Log in

User Name

admin

Password

mU8blRngAvs6uJLb7BBMvN4bZVg5thHT

Sign In