

Practical No. 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.

Cloud_Computing_Practical1 > currency-conversion-service				
	Name	Date modified	Type	Size
	node_modules	12-12-2024 21:21	File folder	
	package	12-12-2024 21:21	JSON Source File	1 KB
	package-lock	12-12-2024 21:21	JSON Source File	31 KB

Now Open with VS Code

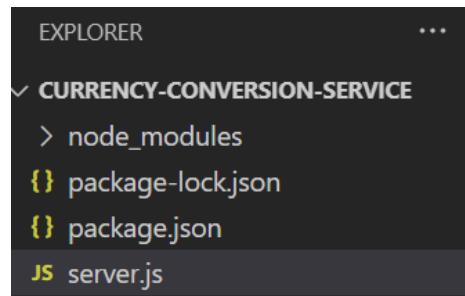
Cloud_Computing_Practical1 > currency-conversion-service			
	Name	Date modified	
	node_modules	12-12-2024 21:21	
	package	12-12-2024 21:21	
	package-lock	12-12-2024 21:21	

Paste
Ctrl+Z
Undo Delete
Open Git GUI here
Open Git Bash here
Open with Code
Give access to >
New >
Properties

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

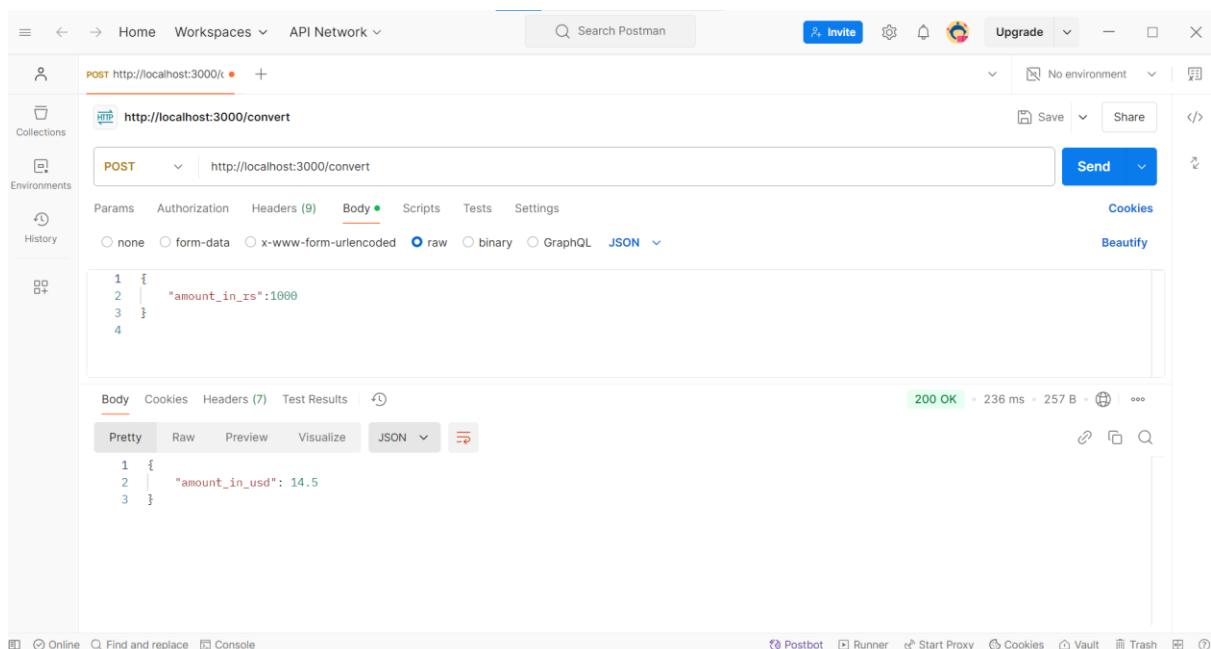
```
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\currency-conversion-service> node server.js
Currency conversion service running on http://localhost:3000
[]
```

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:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections', 'Environments', 'History', and a 'New' button. The main area shows a 'POST' request to 'http://localhost:3000/convert'. The 'Body' tab is selected, showing a raw JSON payload: { "amount_in_rs": 1000 }. The 'Send' button is blue and highlighted. Below the request, the response is shown: a 200 OK status with a response time of 236 ms and a size of 257 B. The response body is { "amount_in_usd": 14.5 }. At the bottom, there are buttons for 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

Successfully output retrieved in Postman to test API.

Java Programming Language

1. Java Client: Implement a Java Client to call the currency conversion service.

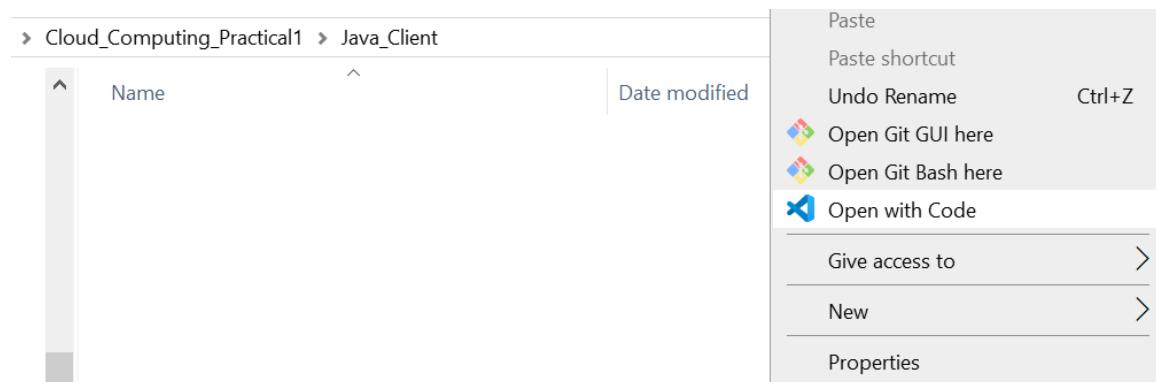
You can use libraries like HttpClient for making HTTP requests.

Create folder name: Java_Client in Cloud_Computing_Practical1

Demonstration:

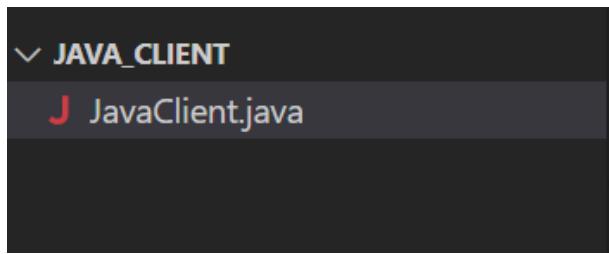
Cloud_Computing_Practical1			
Name	Date modified	Type	Size
currency-conversion-service	12-12-2024 21:47	File folder	
Java_Client	12-12-2024 22:05	File folder	

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:

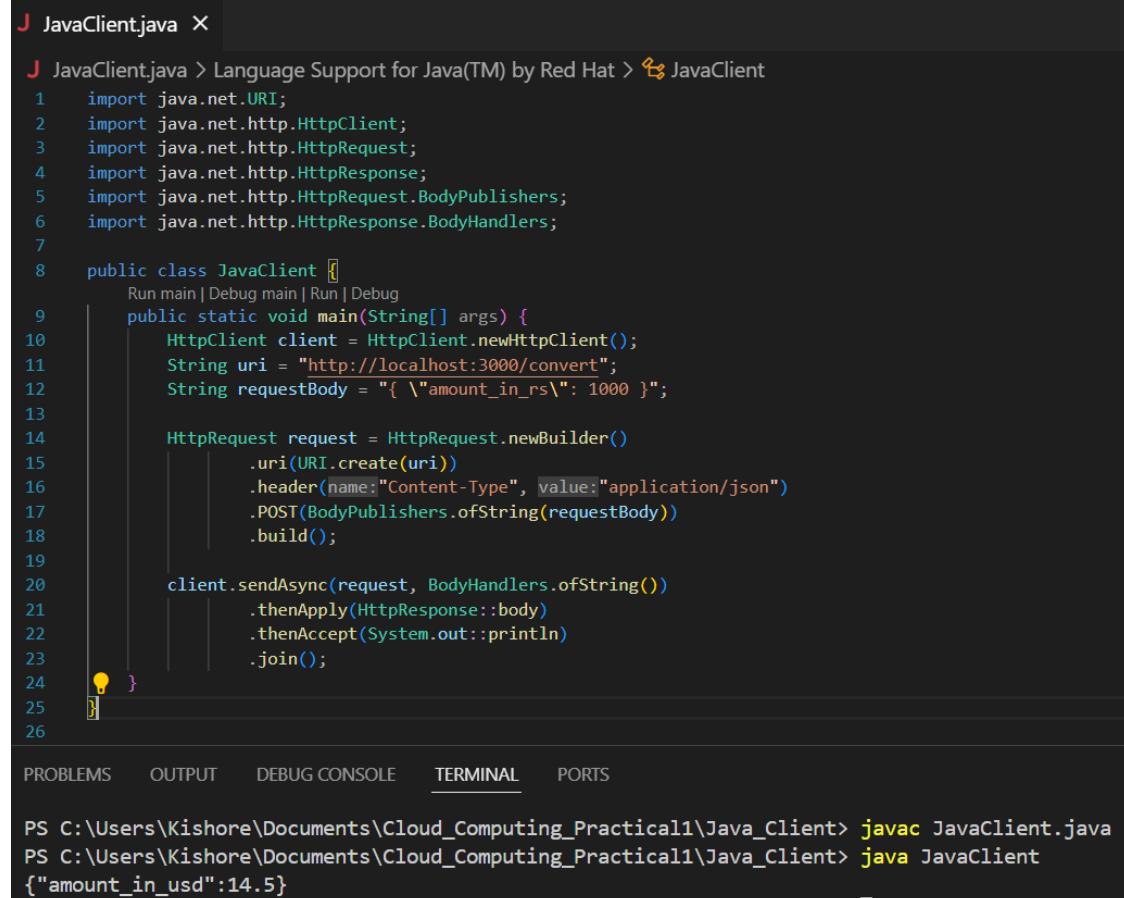
```
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\Java_Client>
javac JavaClient.java
```

```
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:



The screenshot shows a Java code editor with the file `JavaClient.java` open. The code uses the `HttpClient` and `HttpRequest` classes to send a POST request to a local server at `http://localhost:3000/convert` with a JSON body containing `{"amount_in_rs": 1000}`. The response is then printed to the console. The code editor interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. Below the code editor, the terminal window shows the command `javac JavaClient.java` being run, followed by the output `{"amount_in_usd":14.5}`.

```
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}
```

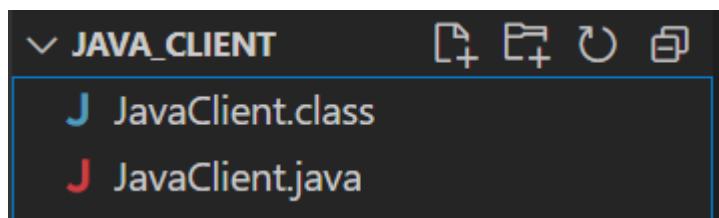
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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.

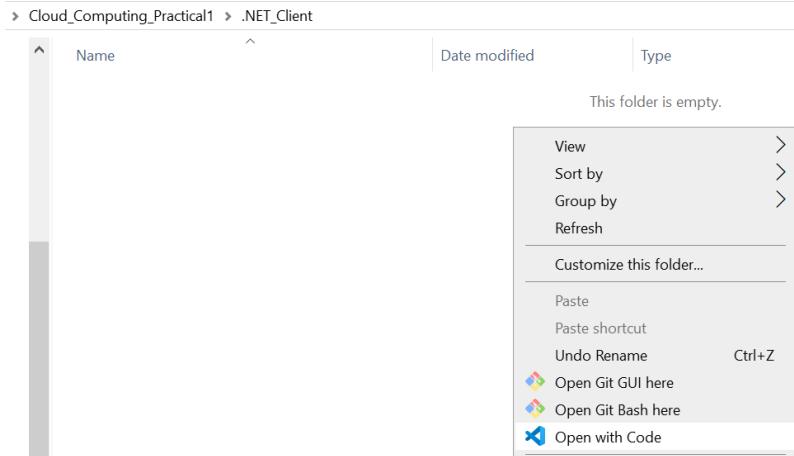
You can use HttpClient in C#.

Create folder name: .NET_Client in Cloud_Computing_Practical1

Demonstration:

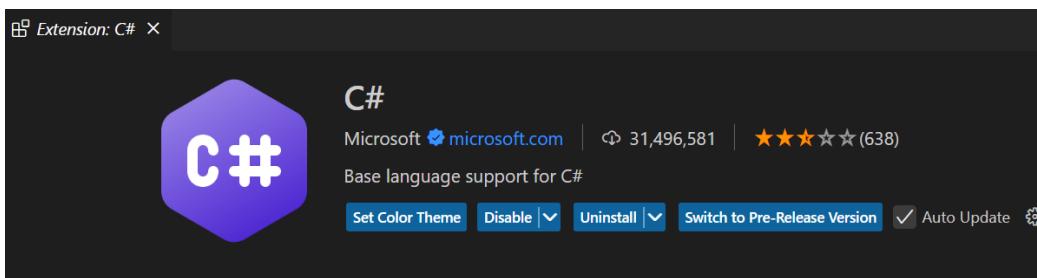
A screenshot of a file explorer window titled 'Cloud_Computing_Practical1'. Inside the folder, there are three items: '.NET_Client', 'currency-conversion-service', and 'Java_Client'. The '.NET_Client' item is highlighted with a blue selection bar. The table has four columns: Name, Date modified, Type, and Size. The '.NET_Client' item is listed with a date of 12-12-2024 22:32, a file folder type, and no size information.

Open .NET_Client and Open with VS Code



Extension: 4 Extensions needed to work in C#

C#



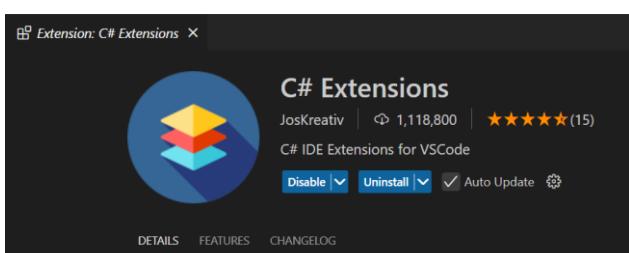
C# Snippet



C# Dev Kit



C# Extensions

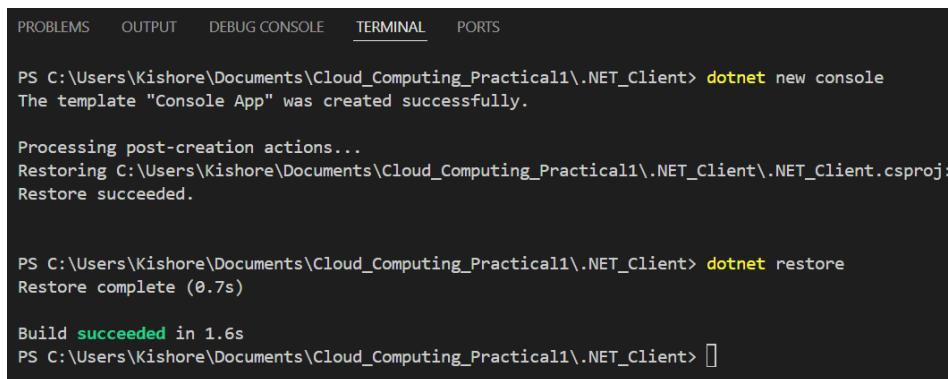


File Architecture:

Click on View > Terminal and Type the following command

- dotnet new console
- dotnet restore

Demonstration:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

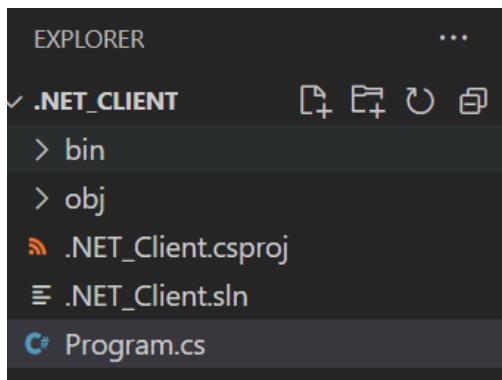
PS C:\Users\Kishore\Documents\Cloud_Computing_Practical1\.NET_Client> dotnet new console
The template "Console App" was created successfully.

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:

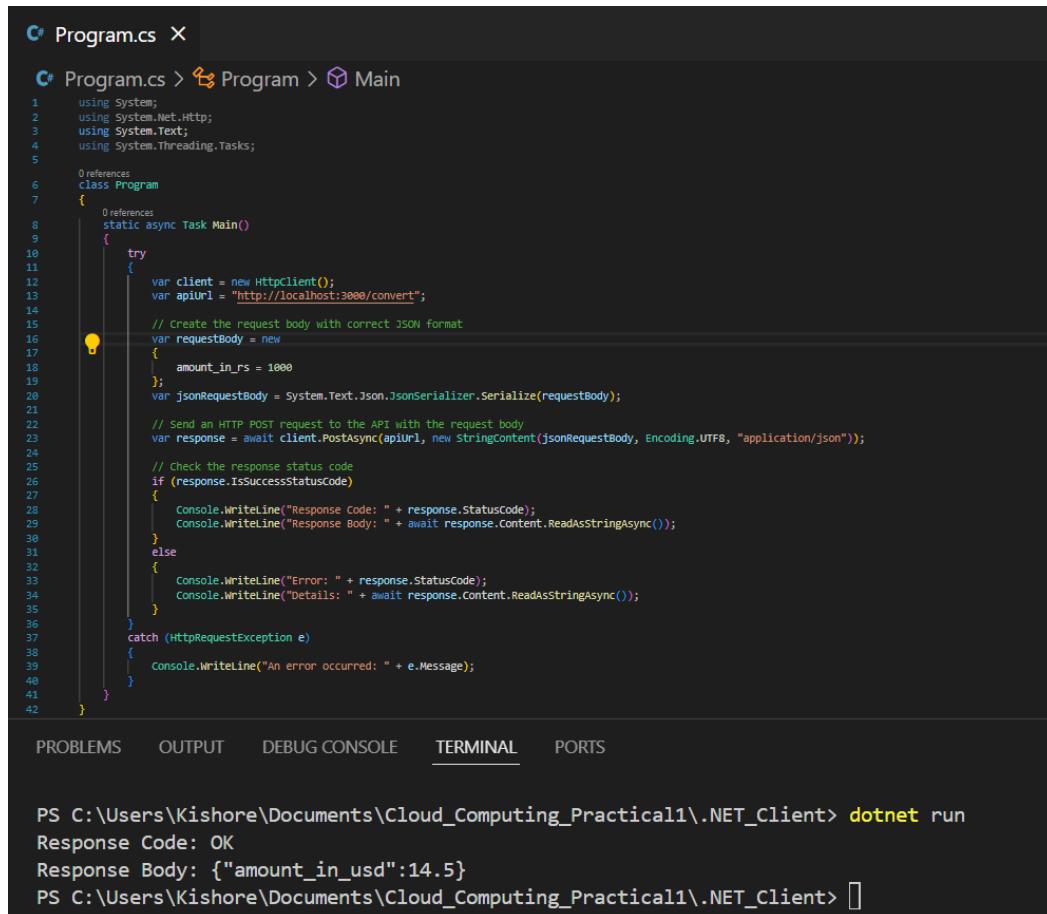
Click View > Terminal:

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 interface. The top part displays the code for `Program.cs` in C#:

```

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

```

The bottom part shows the terminal output:

```

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 No. 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
```

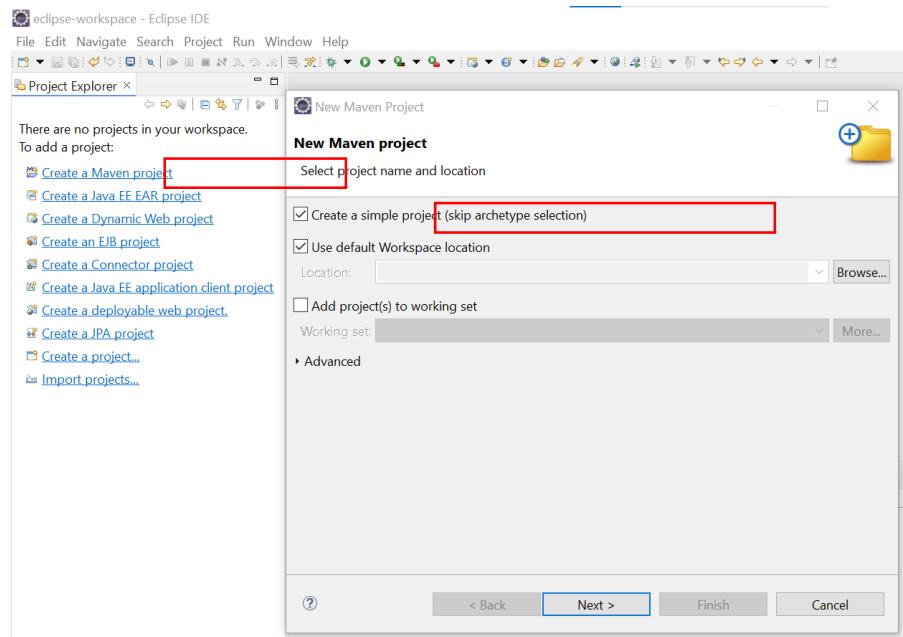
Version checked in command prompt

```
C:\Users\Kishore>mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcfdc97d260186937)
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"

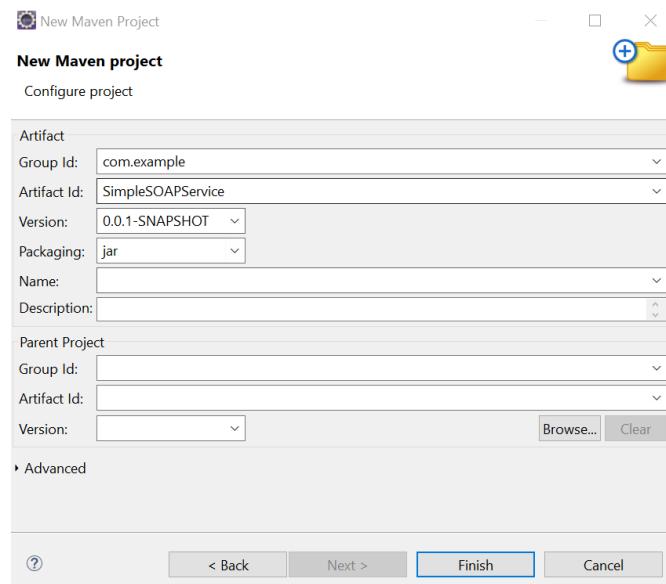
C:\Users\Kishore>java --version
java 23.0.1 2024-10-15
Java(TM) SE Runtime Environment (build 23.0.1+11-39)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.1+11-39, mixed mode, sharing)
```

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>cxfrt-frontend-jaxws</artifactId>
      <version>3.4.5</version>
    </dependency>

    <!-- Apache CXF HTTP Transport -->
    <dependency>
      <groupId>org.apache.cxf</groupId>
      <artifactId>cxfrt-transports-http</artifactId>
      <version>3.4.5</version>
    </dependency>
  
```

<!-- Apache CXF HTTP Jetty Transport (for embedded HTTP server)

-->

```
<dependency>
  <groupId>org.apache.cxf</groupId>
  <artifactId>cxfrt-transports-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 -->

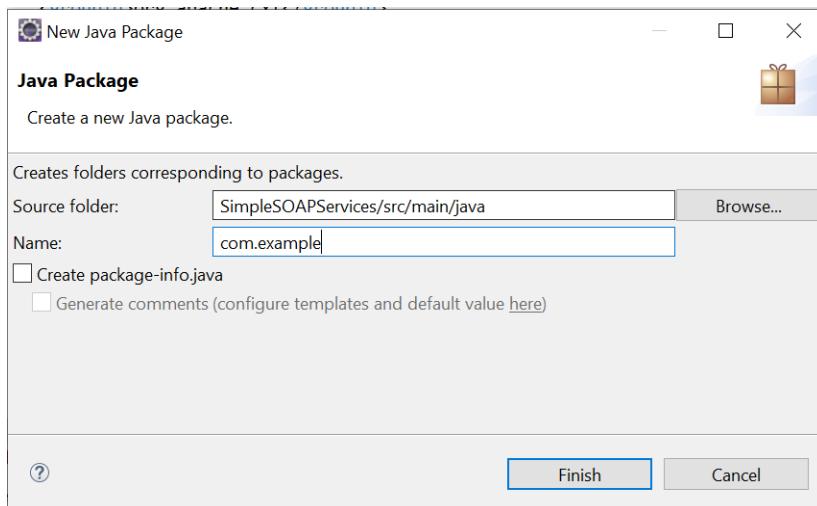
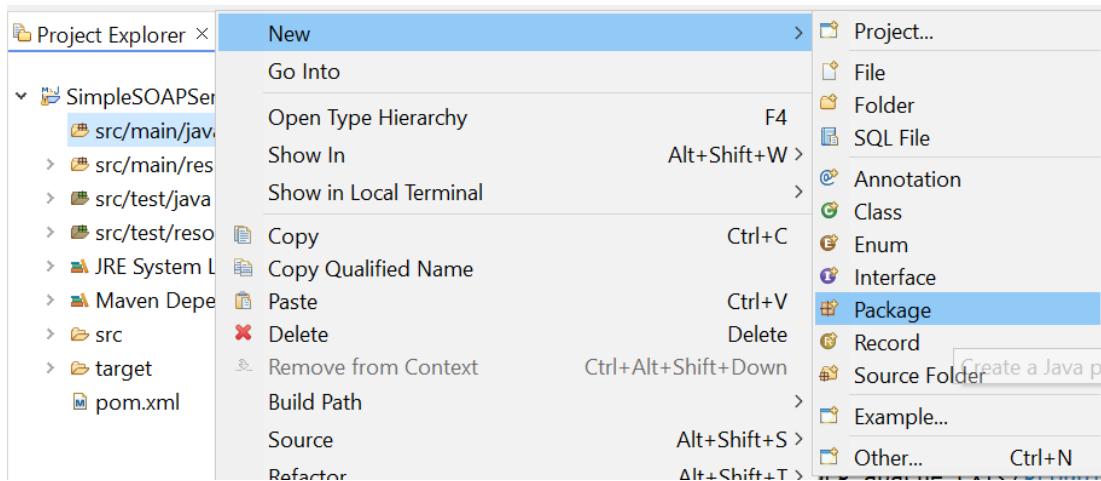
```
<dependency>
  <groupId>javax.annotation</groupId>
  <artifactId>javax.annotation-api</artifactId>
  <version>1.3.2</version>
</dependency>
</dependencies>
</project>
```

After adding dependencies Save it by Ctrl+S

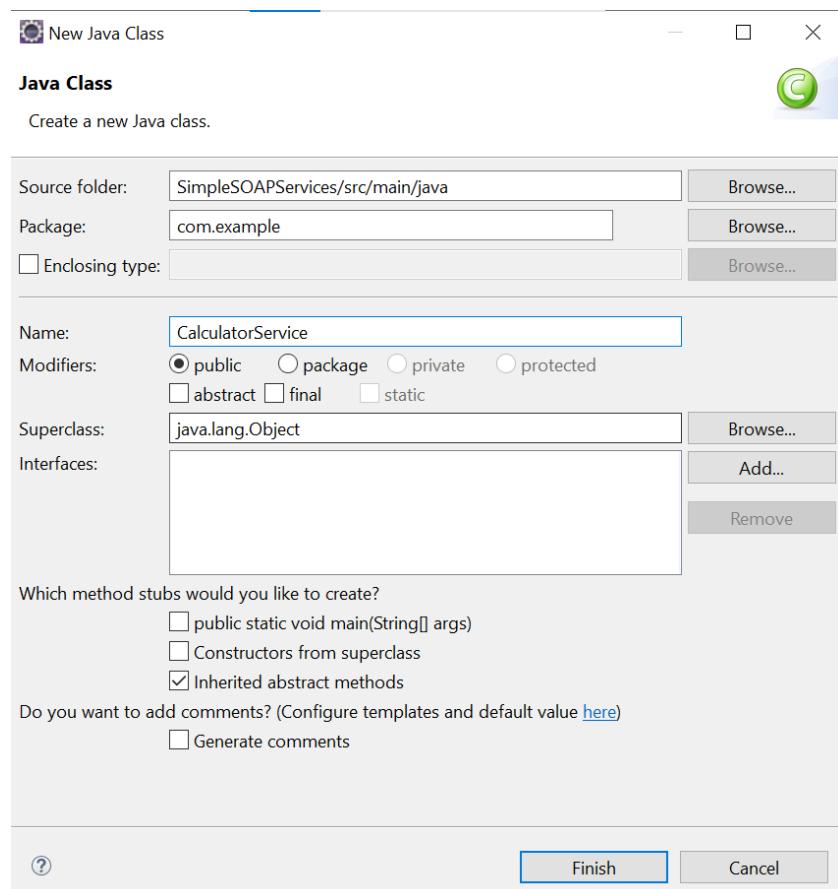
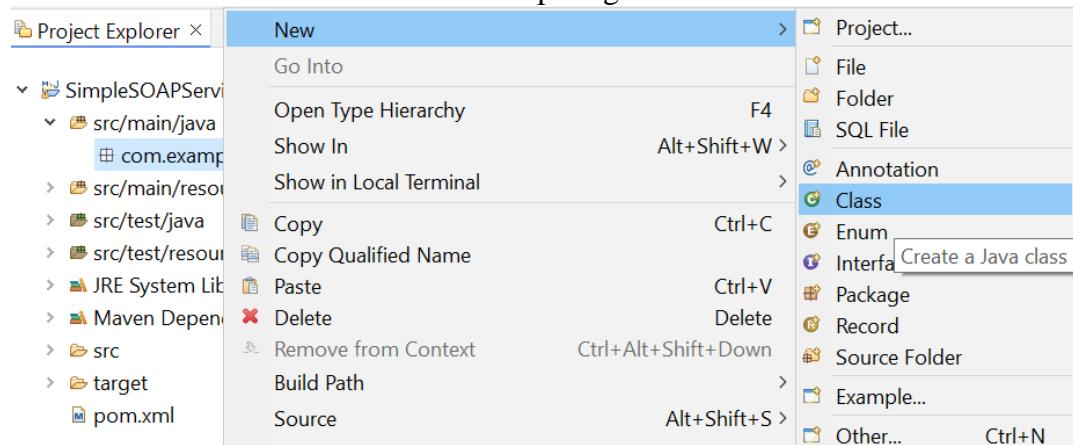
Step 4: Create the Service Interface

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`.



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.jws.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);
```

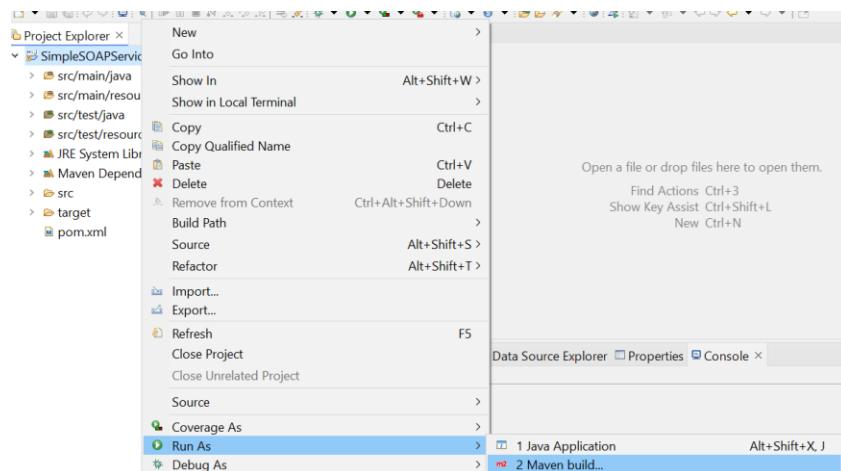
```
}
```

```
}
```

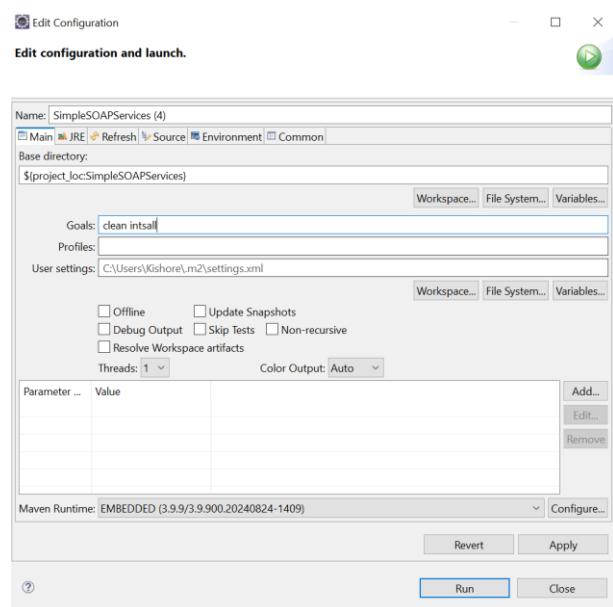
Save It

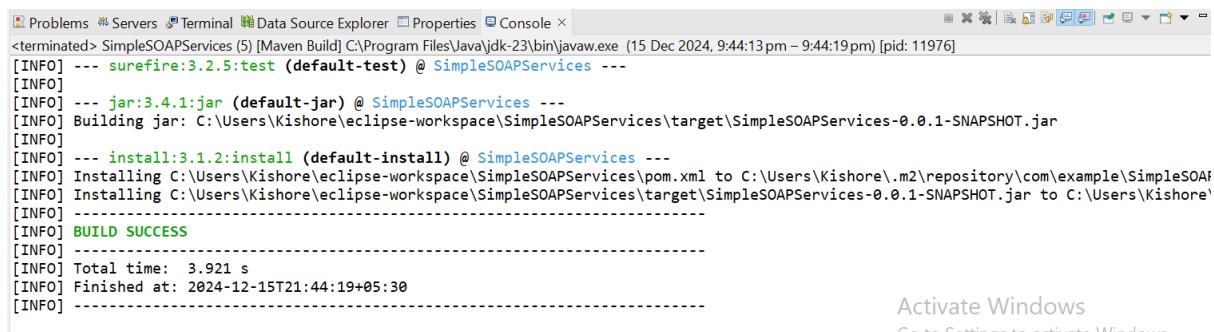
Output:

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





```

Problems Servers Terminal Data Source Explorer Properties Console ×
<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\SimpleSOAPServices\0.0.1-SNAPSHOT\SimpleSOAPServices-0.0.1-SNAPSHOT.jar
[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] -----

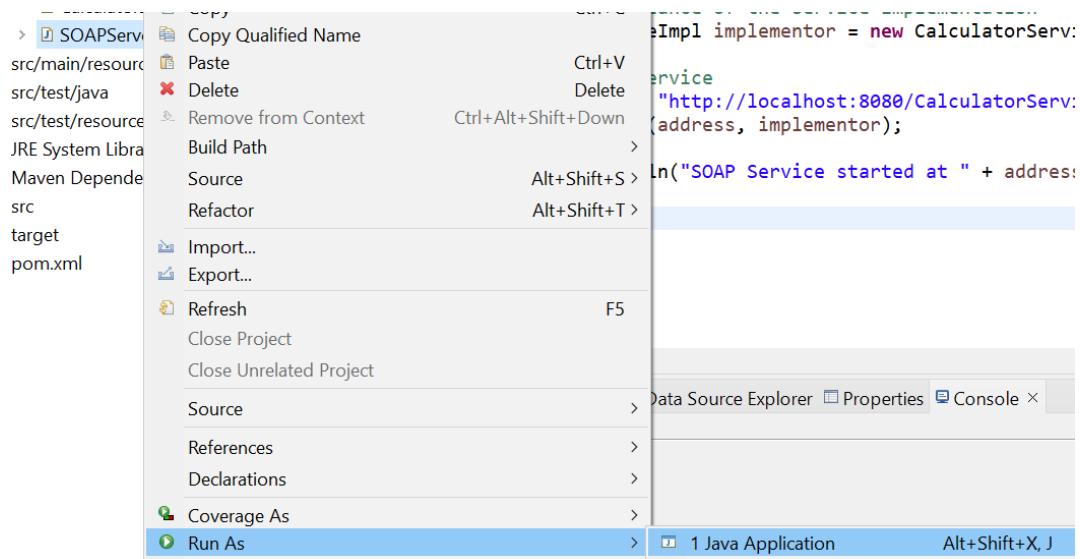
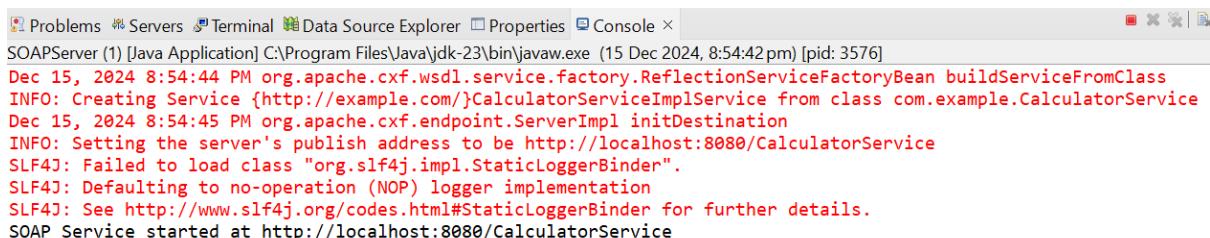
```

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>

```

Problems Servers Terminal Data Source Explorer Properties Console ×
SOAPServer (1) [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (15 Dec 2024, 8:54:42 pm) [pid: 3576]
Dec 15, 2024 8:54:44 PM org.apache.cxf.wsdl.service.factory.ReflectionServiceFactoryBean buildServiceFromClass
INFO: Creating Service {http://example.com/}CalculatorServiceImplService from class com.example.CalculatorService
Dec 15, 2024 8:54:45 PM org.apache.cxf.endpoint.ServerImpl initDestination
INFO: Setting the server's publish address to be http://localhost:8080/CalculatorService
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
SOAP Service started at http://localhost:8080/CalculatorService

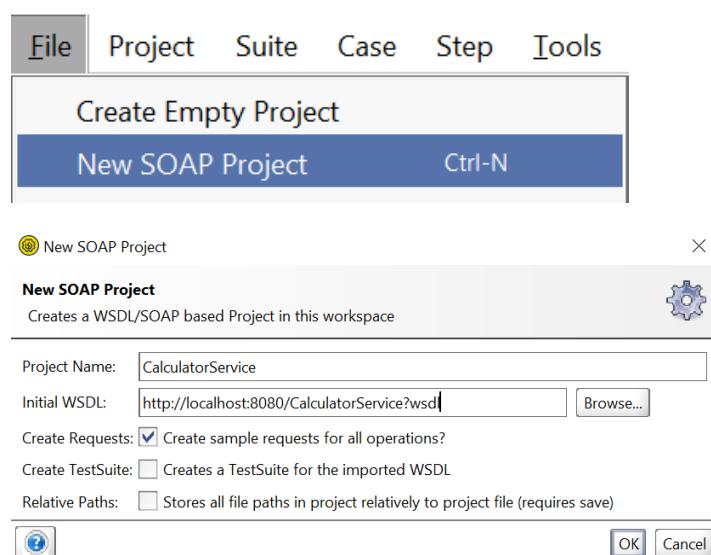
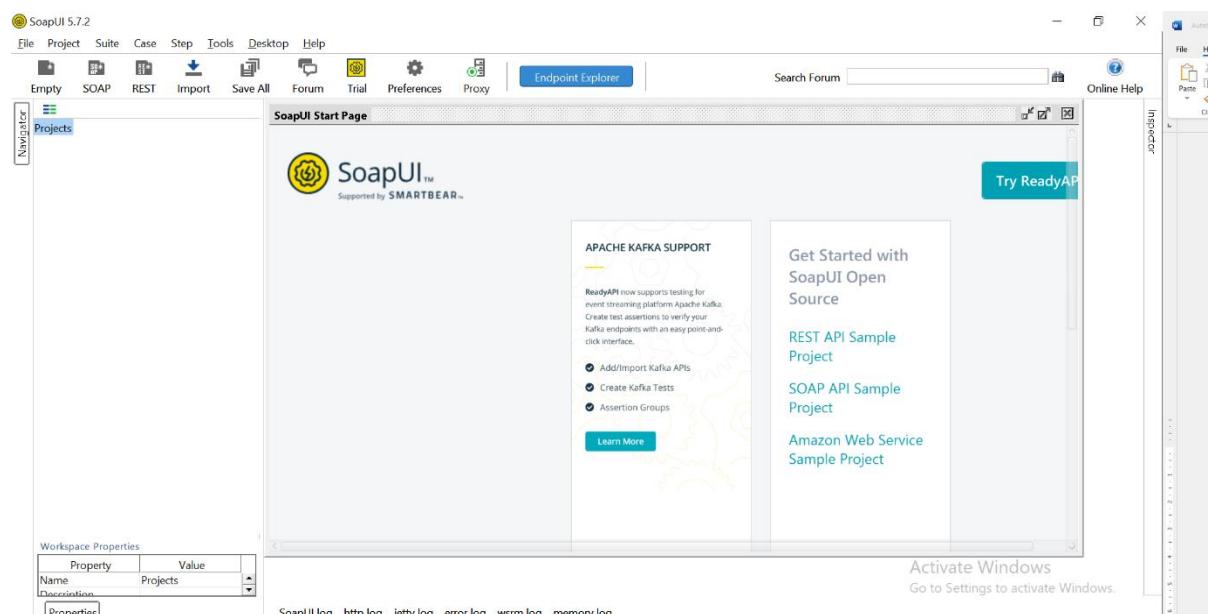
```

Step 7. Test the SOAP Service Using SOAPUI

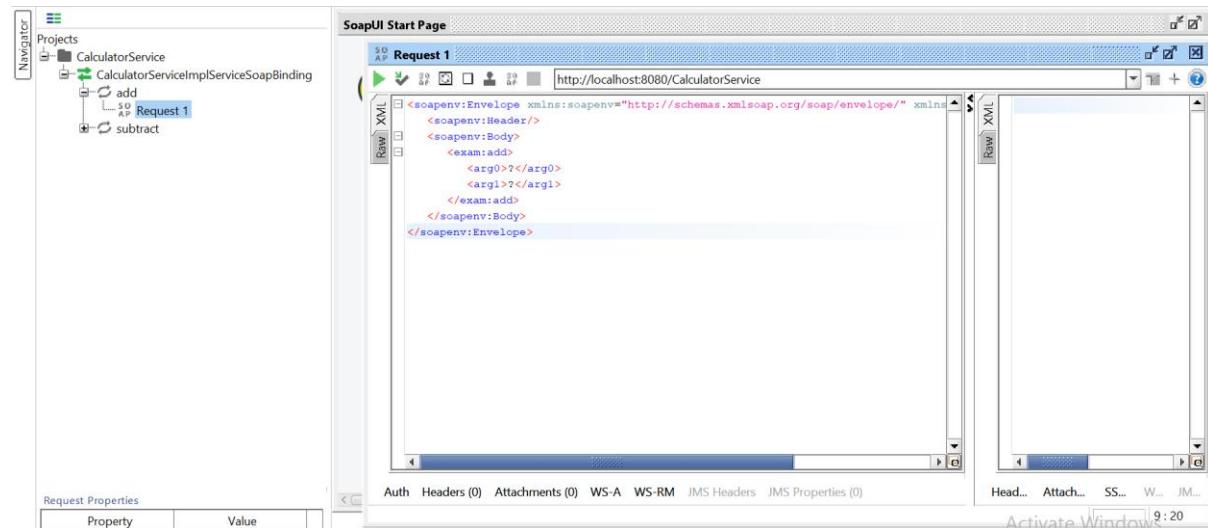
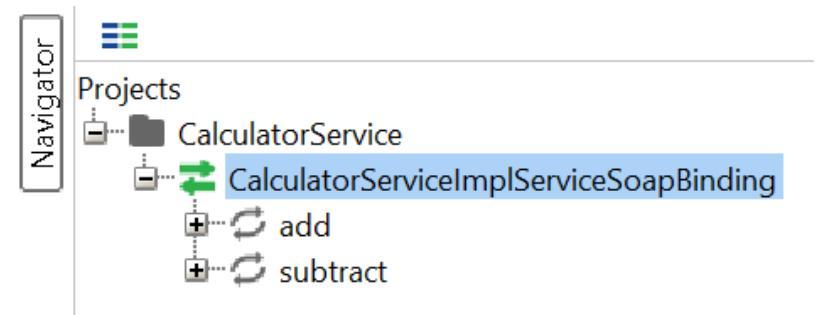
1. Open SOAPUI and create a new project:

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:



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 No. 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/)
- Postman: [Download Postman | Get Started for Free](https://www.getpostman.com/)
(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\

Check Version:

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

C:\Users\Kishore>
```

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

pip install Flask

Demonstration:

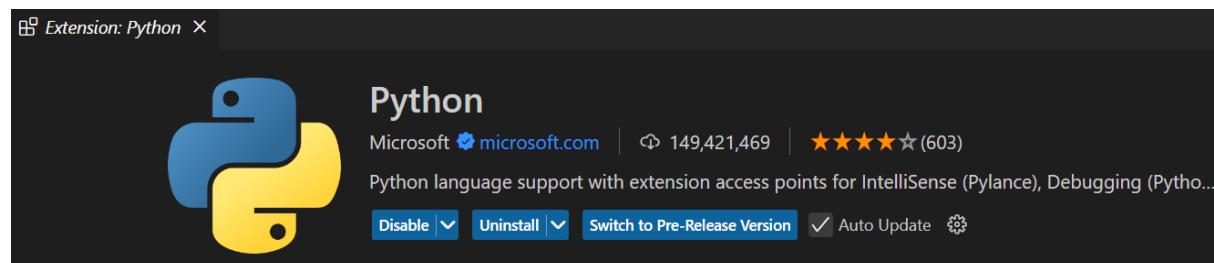
```
C:\Users\Kishore>pip install Flask
Requirement already satisfied: Flask in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (3.0.3)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (from Flask) (3.0.3)
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: MarkupSafe>=2.0 in c:\users\kishore\appdata\local\programs\python\python312\lib\site-packages (from 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



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)
```

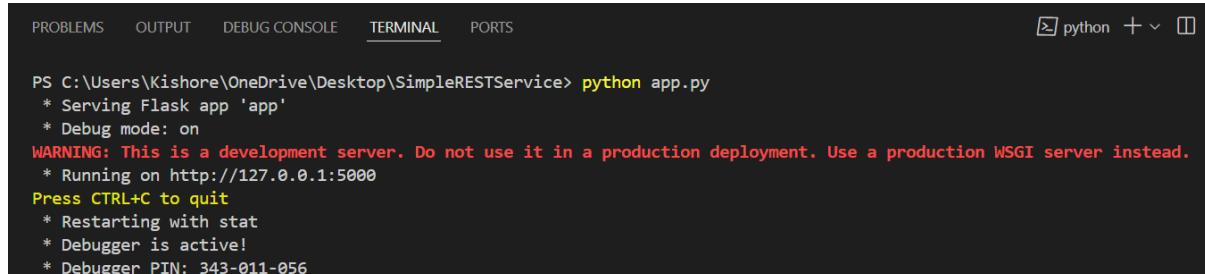
Output:

View>Terminal

Type

Path> **python app.py**

Demonstration:



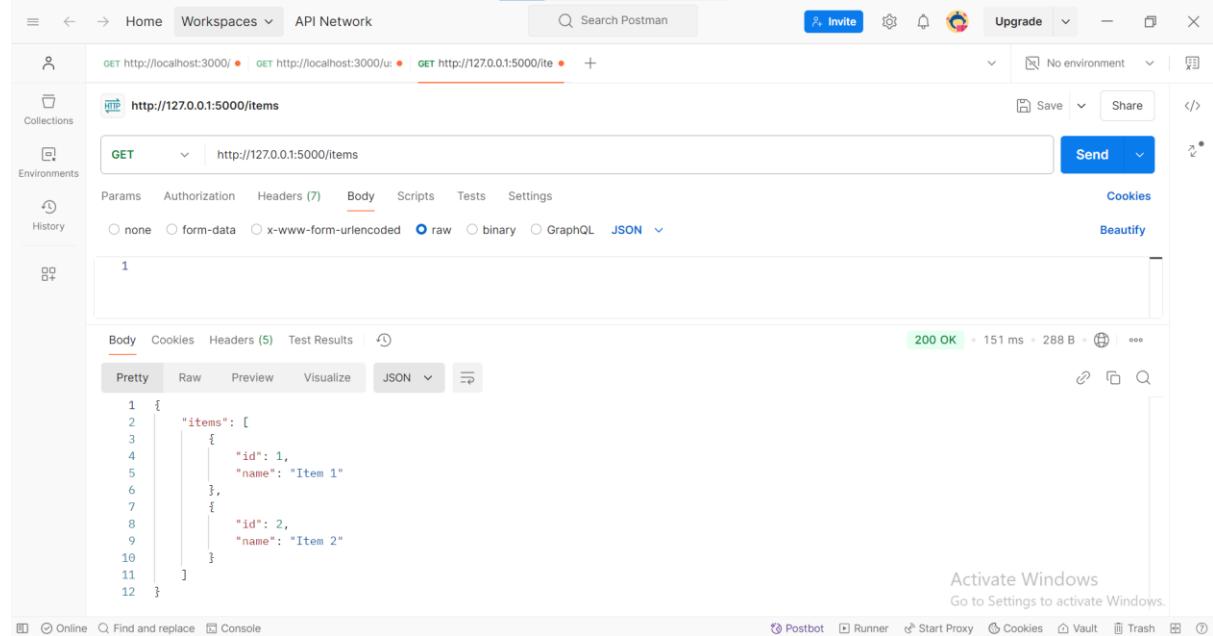
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
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
```

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 shows the Postman application interface. A GET request is made to <http://127.0.0.1:5000/items>. The response is a 200 OK status with a response time of 151 ms and a response size of 288 B. The response body is a JSON object with two items:

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

Endpoint to get a specific item by ID

Send a GET Request:

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

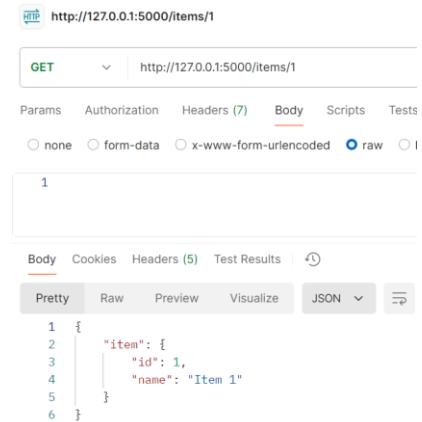
Response:

```
{
  "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:



```

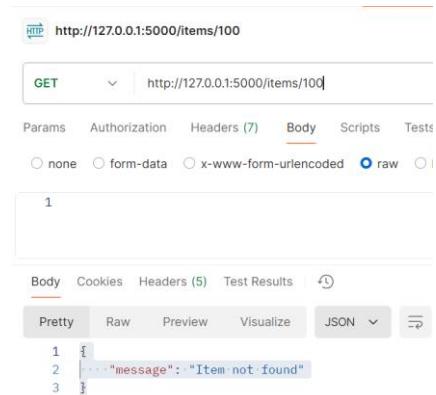
1 {
2   "item": {
3     "id": 1,
4     "name": "Item 1"
5   }
6 }

```

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

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

Demonstration:



```

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"}
```

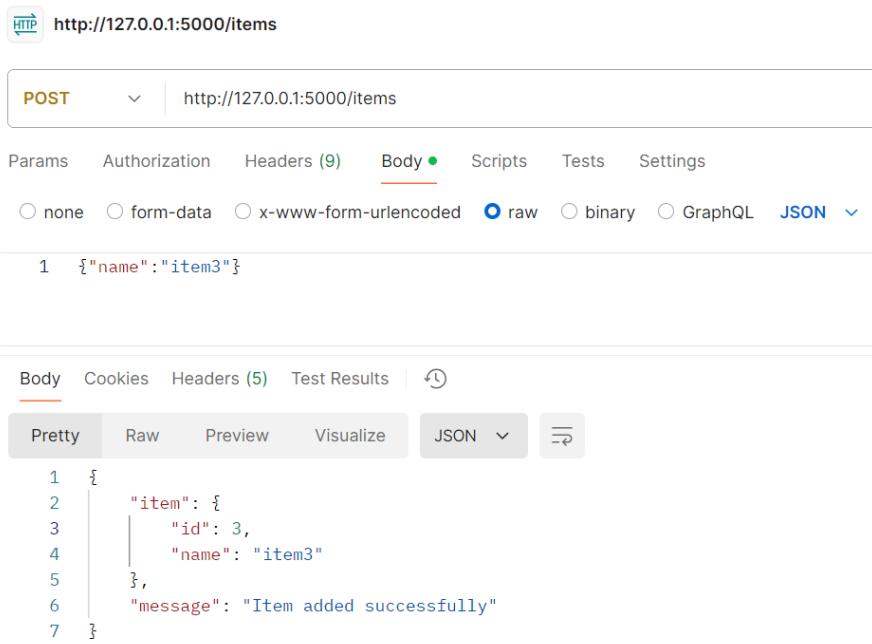
Response:

```
{
  "item": {
    "id": 3,
    "name": "item3"
  }
}
```

"message": "Item added successfully"

}

Demonstration:



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

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

Params Authorization Headers (9) **Body** Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

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

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize **JSON**

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

Practical No. 4

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

To create a simple application that searches for a place using the Google Places API and retrieves the details from the Google Maps Geocoding API.

Steps are as follows:

Software Tools Required

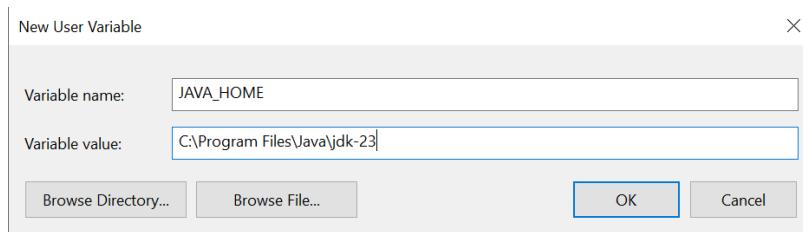
- **Code Editor:** VS Code
- **Gmail:** Before Starting this Practical, your personal Gmail account must be logged in the web browser.
- **Google Cloud SDK:** For managing Google APIs. [Free Trial and Free Tier Services and Products | Google Cloud](#)

Downloads Required:

- **JDK:** [Java Downloads | Oracle India](#) (x64 MSI Installer)
- **Apache Tomcat:** [Apache Tomcat Downloads](#). (x64 bit Windows zip)

After Downloading JDK, Set the Path in Environment Variables in User Variables > Path > New and Provide the Variable name and values to set the Path for JDK as shown below:

Demonstration:



Variable	Value
JAVA_HOME	C:\Program Files\Java\jdk-23

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

Now check the versions that shows path is set for Java JDK

```
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)
```

Step 1: Download and Extract apache-tomcat-9.0.98 zip file into folder, and open apache-tomcat-9.0.98 folder till bin and select the directory path, cut (press backspace) and type **cmd**.

Demonstration:

Directory Path (Your directory path depends where you place your apache-tomcat folder)

Select directory path

C:\> This PC > Documents > Cloud_Computing_Practicals_Solutions_2025 > Cloud_Computing_Practical4_2025 > apache-tomcat-9.0.98 > bin

Type cmd and command prompt will open to the same directory path

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5371]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\bin>
```

Now type: **startup.bat**

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5371]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\bin>startup.bat
Using CATALINA_BASE:  "C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98"
Using CATALINA_HOME:  "C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98"
Using CATALINA_TMPDIR:  "C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\temp"
Using JRE_HOME:  "C:\Program Files\Java\jdk-23"
Using CLASSPATH:  "C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\bin\bootstrap.jar;C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\bin\tomcat-juli.jar"Using CATALINA_OPTS:  ""
```

We will get this tomcat loaded

```

Tomcat
n directory [C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\webapps\examples] has finished in [736] ms
18-Jan-2025 22:40:45.600 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\webapps\host-manager]
18-Jan-2025 22:40:45.676 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\webapps\host-manager] has finished in [76] ms
18-Jan-2025 22:40:45.678 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\webapps\manager]
18-Jan-2025 22:40:45.768 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\webapps\manager] has finished in [88] ms
18-Jan-2025 22:40:45.772 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\webapps\myjsp]
18-Jan-2025 22:40:45.835 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\webapps\myjsp] has finished in [63] ms
18-Jan-2025 22:40:45.835 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\webapps\ROOT]
18-Jan-2025 22:40:45.903 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\Users\Kishore\Documents\Cloud_Computing_Practicals_Solutions_2025\Cloud_Computing_Practical4_2025\apache-tomcat-9.0.98\webapps\ROOT] has finished in [68] ms
18-Jan-2025 22:40:45.919 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
18-Jan-2025 22:40:45.966 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [1694] milliseconds

```

Note:

This Tomcat Server must be kept running in the background by minimizing this window. Otherwise, the we won't get the output.

Another way to load Tomcat

Also, in bin folder we will have startup as Windows Batch File, double click Run and it will load Tomcat.

This PC > Documents > Cloud_Computing_Practicals_Solutions_2025 > Cloud_Computing_Practical4_2025 > apache-tomcat-9.0.98 > bin			
Name	Date modified	Type	Size
configtest	05-12-2024 19:50	Shell Script	2 KB
configtest	05-12-2024 19:50	Shell Script	9 KB
daemon	05-12-2024 19:50	Windows Batch File	3 KB
digest	05-12-2024 19:50	Shell Script	2 KB
digest	05-12-2024 19:50	Shell Script	2 KB
makebase	05-12-2024 19:50	Windows Batch File	4 KB
makebase	05-12-2024 19:50	Shell Script	4 KB
service	05-12-2024 19:50	Windows Batch File	9 KB
setclasspath	05-12-2024 19:50	Windows Batch File	4 KB
setclasspath	05-12-2024 19:50	Shell Script	5 KB
shutdown	05-12-2024 19:50	Windows Batch File	2 KB
shutdown	05-12-2024 19:50	Shell Script	2 KB
startup	05-12-2024 19:50	Windows Batch File	2 KB

Step 2: After completing Step 1, Now open apache-tomcat-9.0.98 folder, open a folder name **webapps** which is already created in apache-tomcat-9.0.98 folder

This PC > Documents > Cloud_Computing_Practicals_Solutions_2025 > Cloud_Computing_Practical4_2025 > apache-tomcat-9.0.98 >

Name	Date modified	Type	Size
bin	05-12-2024 19:50	File folder	
conf	17-01-2025 12:19	File folder	
lib	05-12-2024 19:50	File folder	
logs	18-01-2025 22:40	File folder	
temp	05-12-2024 19:50	File folder	
webapps	17-01-2025 12:23	File folder	

Open **webapps** folder, create a new folder and name it as: **myjsp**

Name	Date modified	Type	Size
docs	05-12-2024 19:50	File folder	
examples	05-12-2024 19:50	File folder	
host-manager	05-12-2024 19:50	File folder	
manager	05-12-2024 19:50	File folder	
myjsp	17-01-2025 12:20	File folder	

Step 3: Open **myjsp** folder and open it with VS Code

Name	Date modified	Type	Size
This folder is empty.			
View > Sort by > Group by > Refresh <hr/> Customize this folder... Paste Paste shortcut Undo Delete Ctrl+Z Open Git GUI here Open Git Bash here Open with Code			

Create 2 jsp files with name **myindex.jsp** and **myinput.jsp**

1. Filename: myindex.jsp : **Display Map with Google Maps API**

Code:

```

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

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Google Maps Location</title>

<!-- Google Maps JavaScript API -->

```

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

<style>
/* Set the size of the map container */
#map {
  height: 400px;
  width: 100%;
}

</style>

</head>

<body>
<%
// Fetch latitude and longitude from form inputs
String latParam = request.getParameter("t1");
String longParam = request.getParameter("t2");
// Default values in case parameters are not provided
double lati = 40.7128; // Default latitude (New York)
double longi = -74.0060; // Default longitude (New York)
// If parameters exist, parse them to doubles
if (latParam != null && !latParam.trim().isEmpty()) {
  lati = Double.parseDouble(latParam);
}
if (longParam != null && !longParam.trim().isEmpty()) {
  longi = Double.parseDouble(longParam);
}
%>
<h3>Google Maps Location</h3>
```

```
<div id="map"></div>
```

```
<script>
```

```
    // Initialize the map
```

```
    function initMap() {
```

```
        var location = { lat: <%= lati %>, lng: <%= longi %> }; // Use JSP values  
        for lat and lng
```

```
        // Create a new map centered on the location
```

```
        var map = new google.maps.Map(document.getElementById('map'), {
```

```
            zoom: 10, // Zoom level
```

```
            center: location // Center the map on the provided coordinates
```

```
        });
```

```
        // Add a marker at the location
```

```
        var marker = new google.maps.Marker({
```

```
            position: location,
```

```
            map: map
```

```
        });
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

2. Filename: myinput.jsp : Form to Get Latitude and Longitude

Code:

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

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

<title>Enter Latitude and Longitude</title>

</head>

<body>

<!-- Form to accept input values for latitude and longitude -->

<form action="myindex.jsp" method="get">

<pre>

Enter latitude: <input type="text" name="t1" />

Enter longitude: <input type="text" name="t2" />

<input type="submit" value="Show" />

</pre>

</form>

</body>

</html>

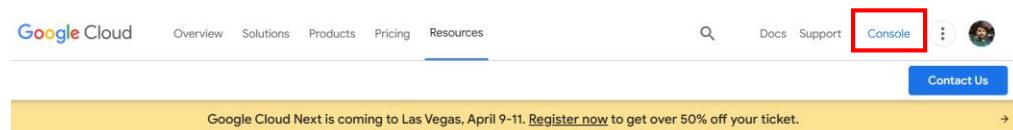
Your **myjsp** folder will look like this

> This PC > Documents > Cloud_Computing_Practicals_Solutions_2025 > Cloud_Computing_Practical4_2025 > apache-tomcat-9.0.98 > webapps > myjsp				
	Name	Date modified	Type	Size
	myindex	18-01-2025 22:55	Java Server Pages So...	2 KB
	myinput	16-01-2025 01:29	Java Server Pages So...	1 KB

Step 4: Open Google Cloud Platform Console,

[Free Trial and Free Tier Services and Products | Google Cloud](https://cloud.google.com/)

Click on Console



Build what's next in generative AI

Try Gemini 2.0 models, the latest and most advanced multimodal models in Vertex AI. See what you can build with up to a 2M token context window, starting as low as \$0.0001.

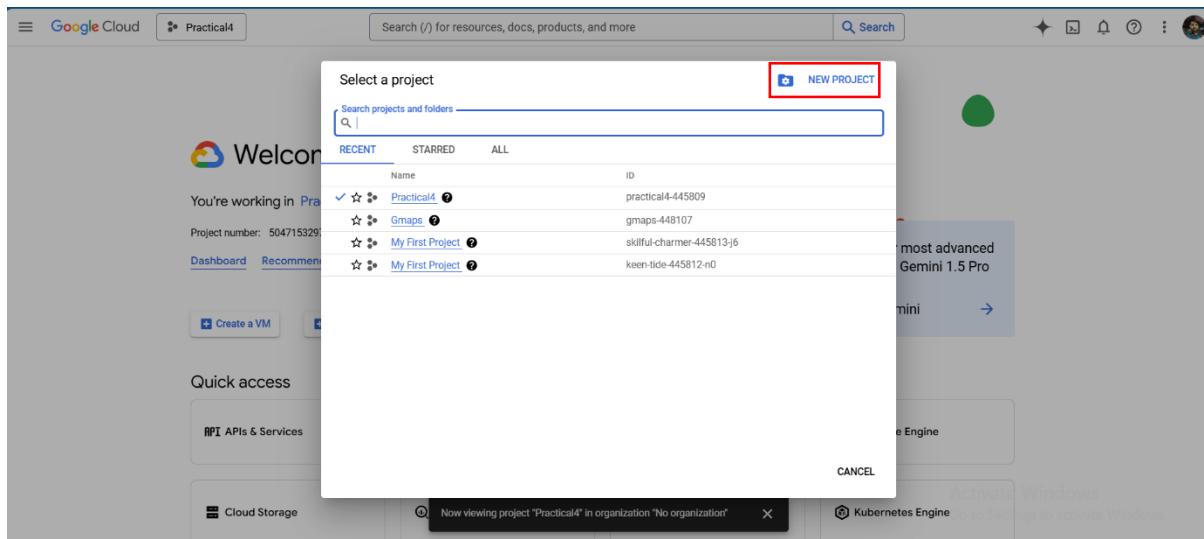
[Try it in console](#)
[Contact sales](#)
[Activate Windows](#)

Step-by-Step Guide to Implement Google Maps API with JSP

1. Get Your Google Maps API Key

- First, you need to obtain an API key from **Google Cloud Console** to use the **Google Maps JavaScript API**. Follow these steps:
 - Go to [Google Cloud Console](#).
 - Create a new project (or select an existing one).
 - Enable the **Maps JavaScript API**. (Instead of Places API and Geocoding API)
 - Under **Credentials**, create an **API Key** and copy it.

Demonstration: Here we have existing projects, now click on **NEW PROJECT**



Project Name: GoogleMapsAPI and click **CREATE**

New Project

⚠ You have 16 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name *

GoogleMapsAPI

?

Project ID: confident-totem-448217-g0. It cannot be changed later. [EDIT](#)

Location *

No organization

[BROWSE](#)

Parent organization or folder

[CREATE](#)

[CANCEL](#)

Now we can see our new project GoogleMapsAPI and Notifications of new project

Search, Click and Enable the Maps JavaScript API.

After Enabling Maps JavaScript API, it will show notifications and also redirect to billing method (Ignore billing method)

Create a new billing account

Name * The name of this billing account is only used to help you remember what it is.

Country *

Currency

Continue **Cancel**

Notifications

- Enable service: maps-backend.googleapis.com GoogleMapsAPI Just now
- Create Project: GoogleMapsAPI 5 minutes ago [Select Project](#)

Now click on Navigation Menu

Navigation menu (.)

Create a new billing account

Name *

APIs & Services > Credentials

Cloud overview >

Solutions >

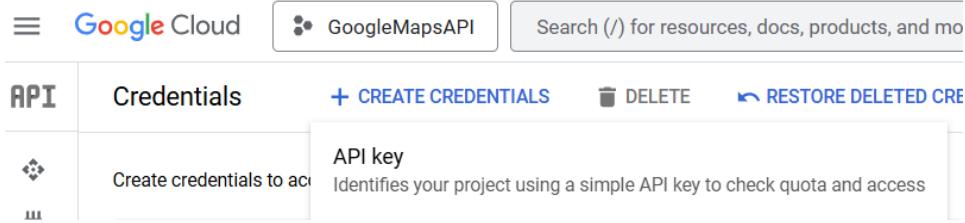
Pinned products

API APIs & Services > **Enabled APIs & services**

Marketplace >

Vertex AI > **Credentials**

Click + CREATE CREDENTIALS > API Key



The screenshot shows the Google Cloud API Credentials page. The 'GoogleMapsAPI' project is selected. A modal window is open, titled 'Creating API key...', with the sub-section 'API key' selected. The text inside the modal says: 'Identifies your project using a simple API key to check quota and access'. At the bottom of the modal are 'CLOSE' and 'CREATE API KEY' buttons.

Creating API key...



CLOSE

We got the API Key Created, Copy the API Key and paste it in **myindex.jsp** file.

API key created

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

Your API key 

 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

Now, open **myjsp** folder and open with VS Code and paste the `API_KEY` in **myindex.jsp** script tag:

Shown below:

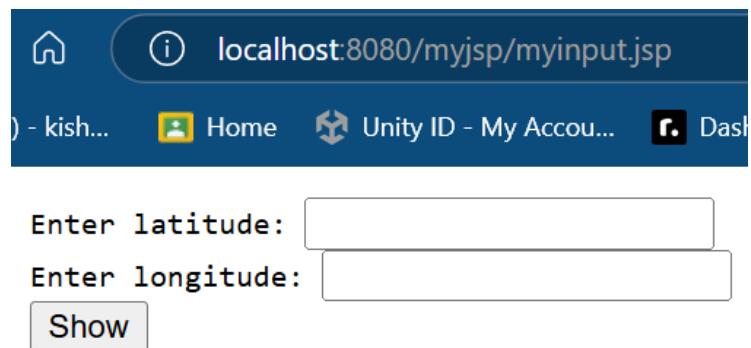
Before: <!-- Google Maps JavaScript API -->

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

After: <!-- Google Maps JavaScript API --><script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAiofE1T6K975oNisc37nvURK7c87XR7e4 &callback=initMap" async defer></script>

Step 5: Now to check the output, open a web browser Google Chrome or Microsoft Edge Ctrl + Click to follow the Link, URL:
<http://localhost:8080/myjsp/myinput.jsp>

Demonstration:



localhost:8080/myjsp/myinput.jsp

Enter latitude:

Enter longitude:

Show

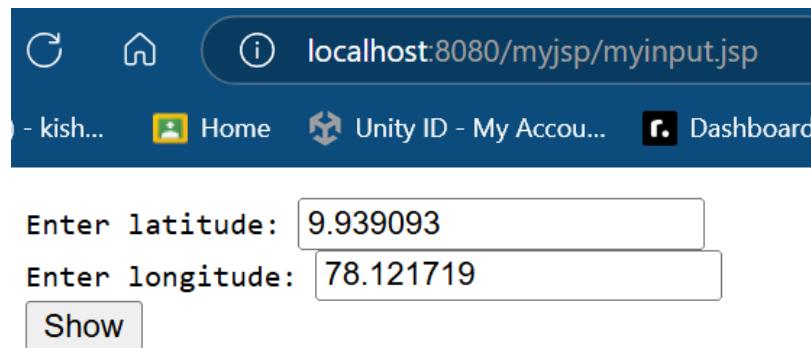
Enter Example latitude and longitude of any place

Here, latitude and longitude coordinates of city: Madurai

Latitude: 9.939093

Longitude: 78.121719

Output: Enter co-ordinates and click on Show



localhost:8080/myjsp/myinput.jsp

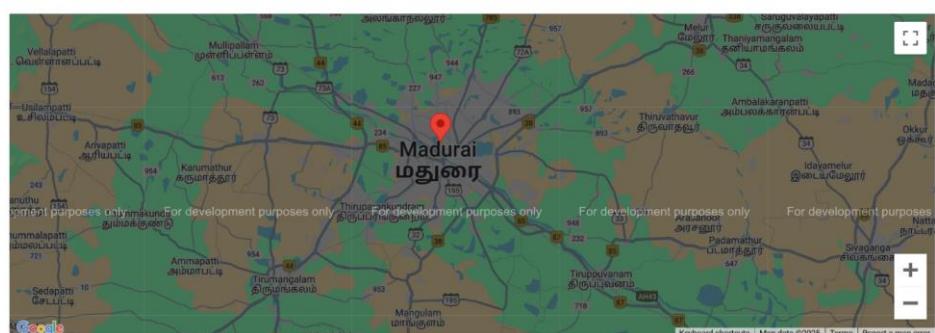
Enter latitude: 9.939093

Enter longitude: 78.121719

Show

It will redirect to the Google Maps: Exact Coordinates Location

Google Maps Location



Practical No.5

Installation and Configuration of virtualization using KVM.

Software Tools Required

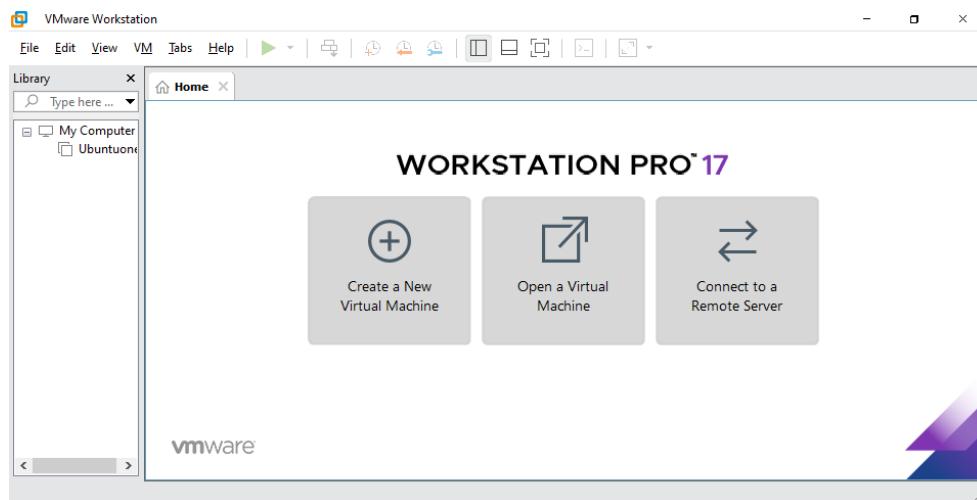
- **Virtual Machine Monitor:** VMware Workstation
- **Operating System:** Linux

Downloads Required:

- Virtual Machine Monitor: [Download VMware Workstation Pro 17.6.0 Build 24238078 for Windows | Uptodown.com](https://www.uptodown.com/windows-software/VMware-Workstation-Pro-17.6.0-Build-24238078)
- Operating System: [Download Ubuntu Desktop | Ubuntu](https://www.ubuntu.com/download/desktop) (for Linux distribution of your choice)

Open Vmware Workstation and start the steps

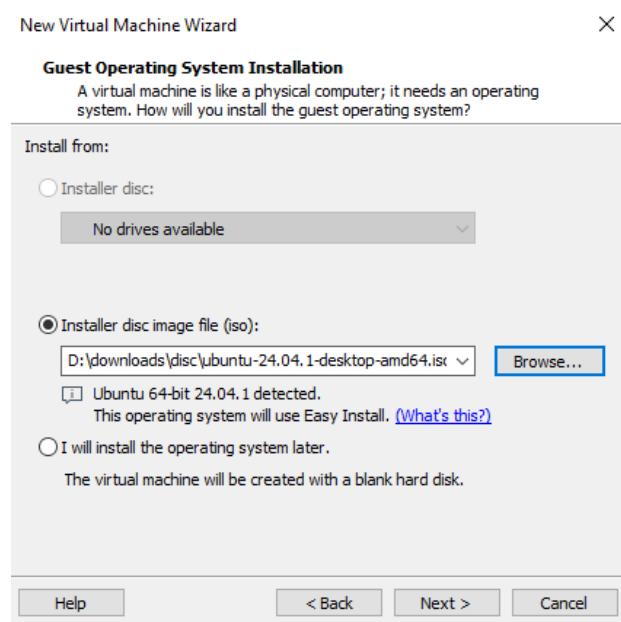
Demonstration:



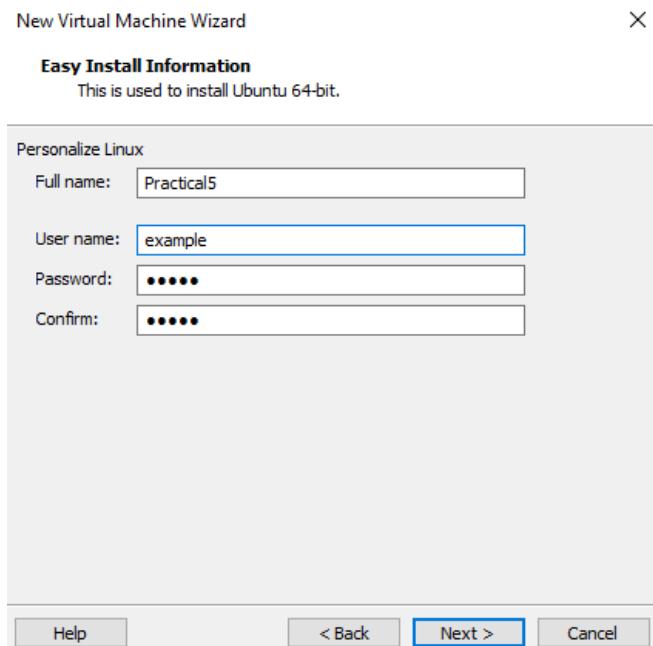
Create a New Virtual Machine >



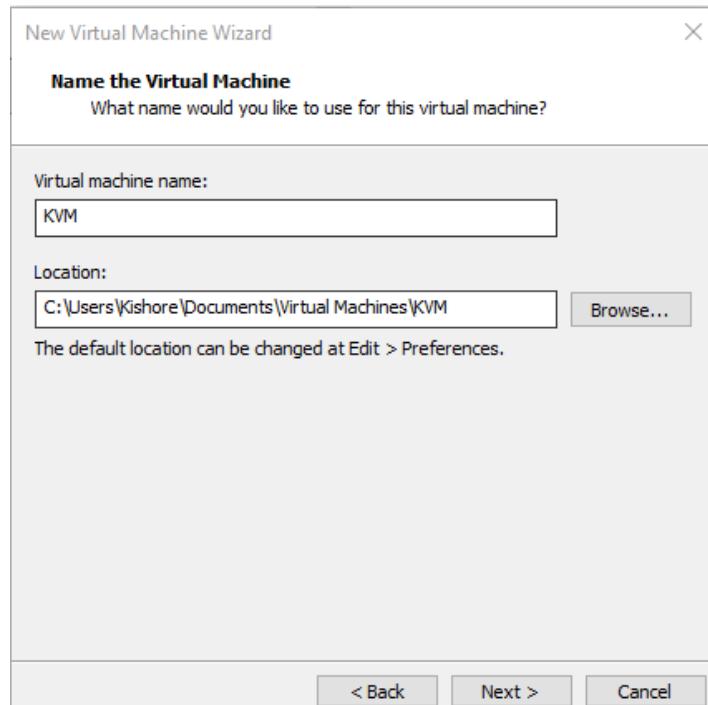
Browse and place the ISO File



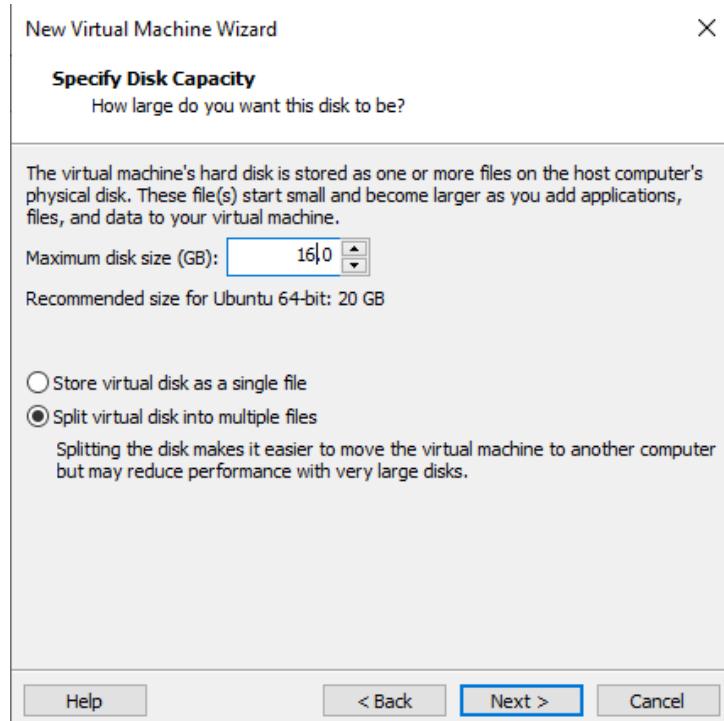
Provide full name as **Practical5** and user name as **example** and password and confirm password as **linux**

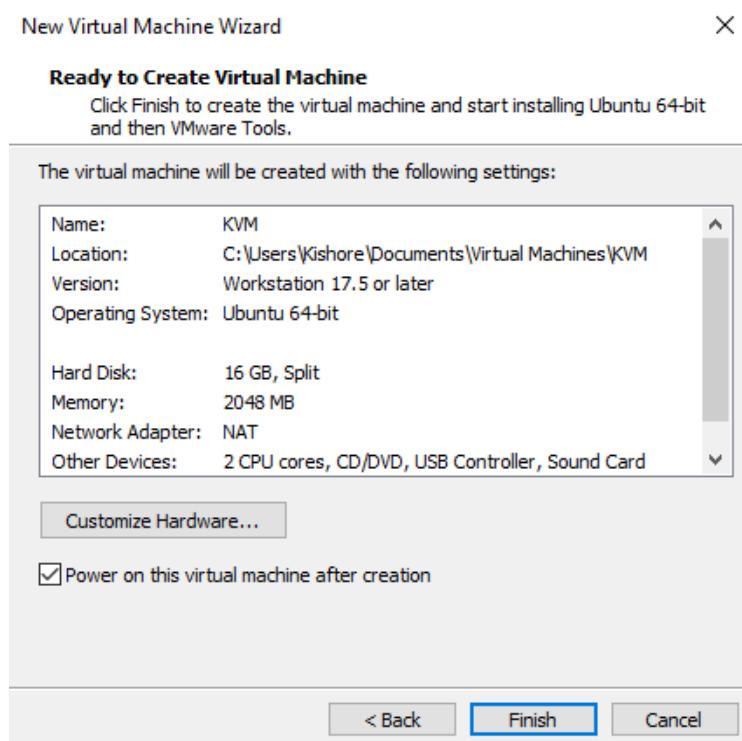


Give virtual machine name as **KVM**



Provide disk size: 16.0GB then click Next





To enable KVM

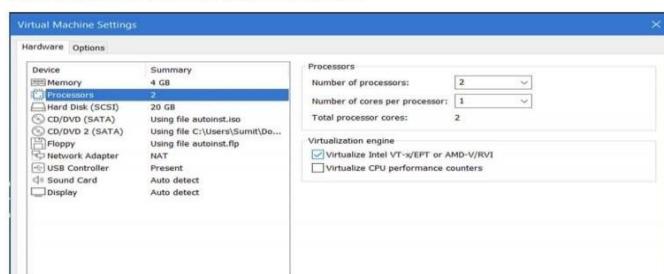
Check the Virtualize Intel VT

For enabling KVM

1. Edit virtual machine settings



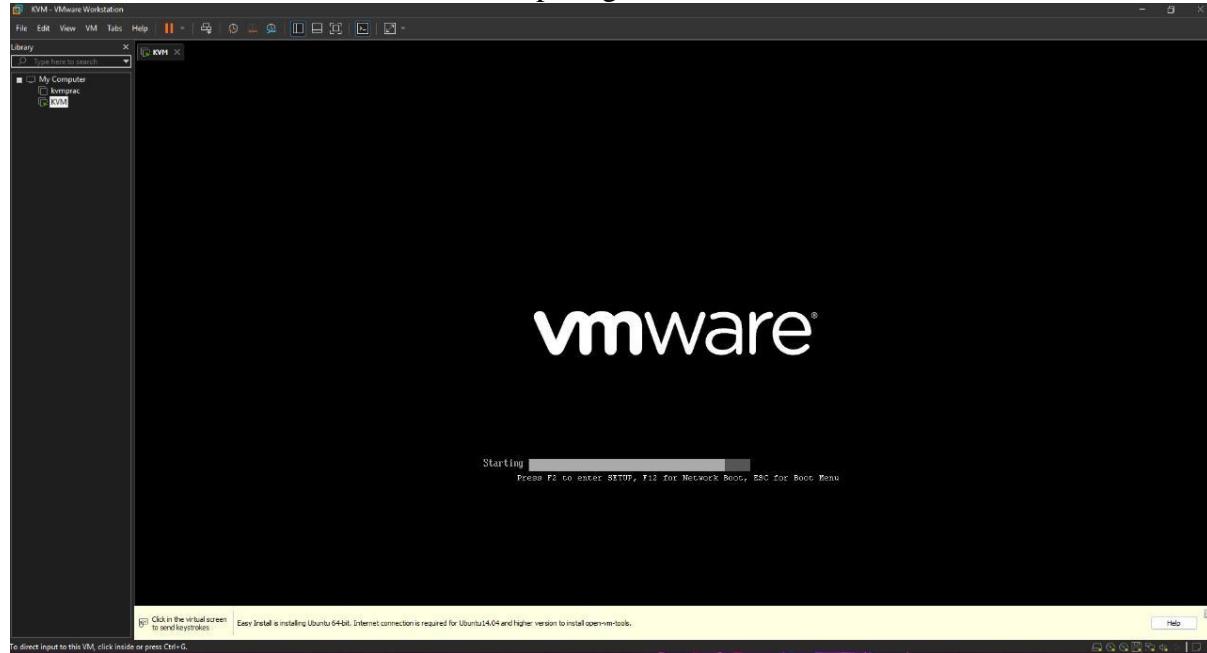
2. Select processor then tick the Virtualize Intel VT



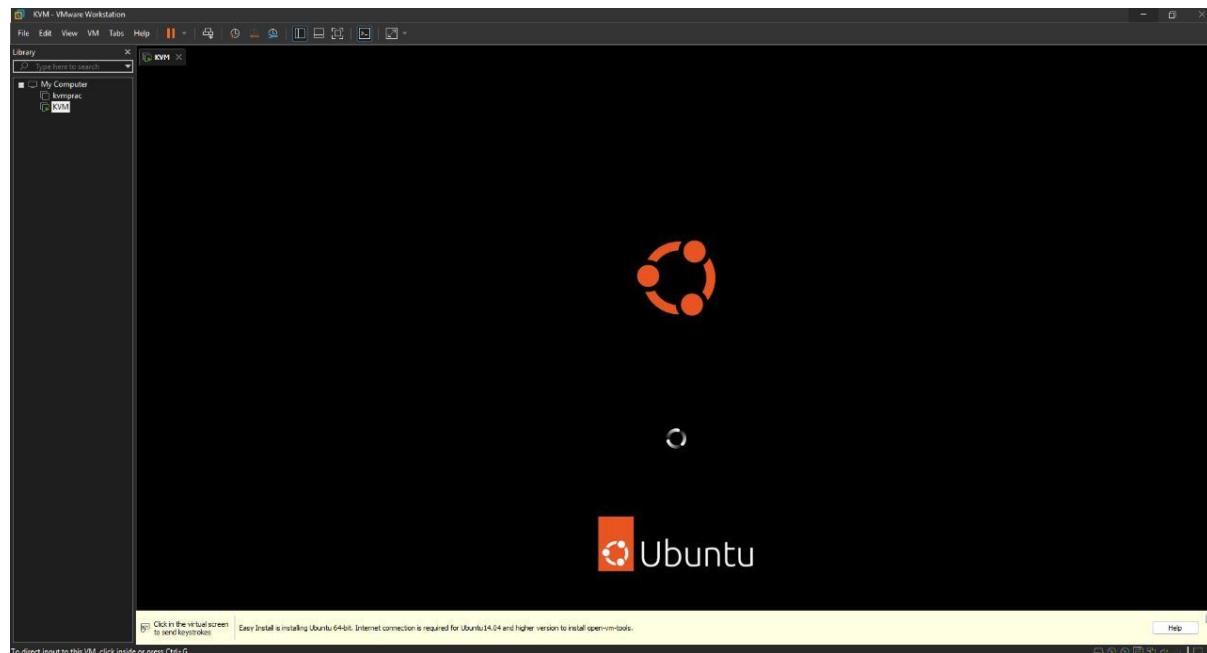
Select OK

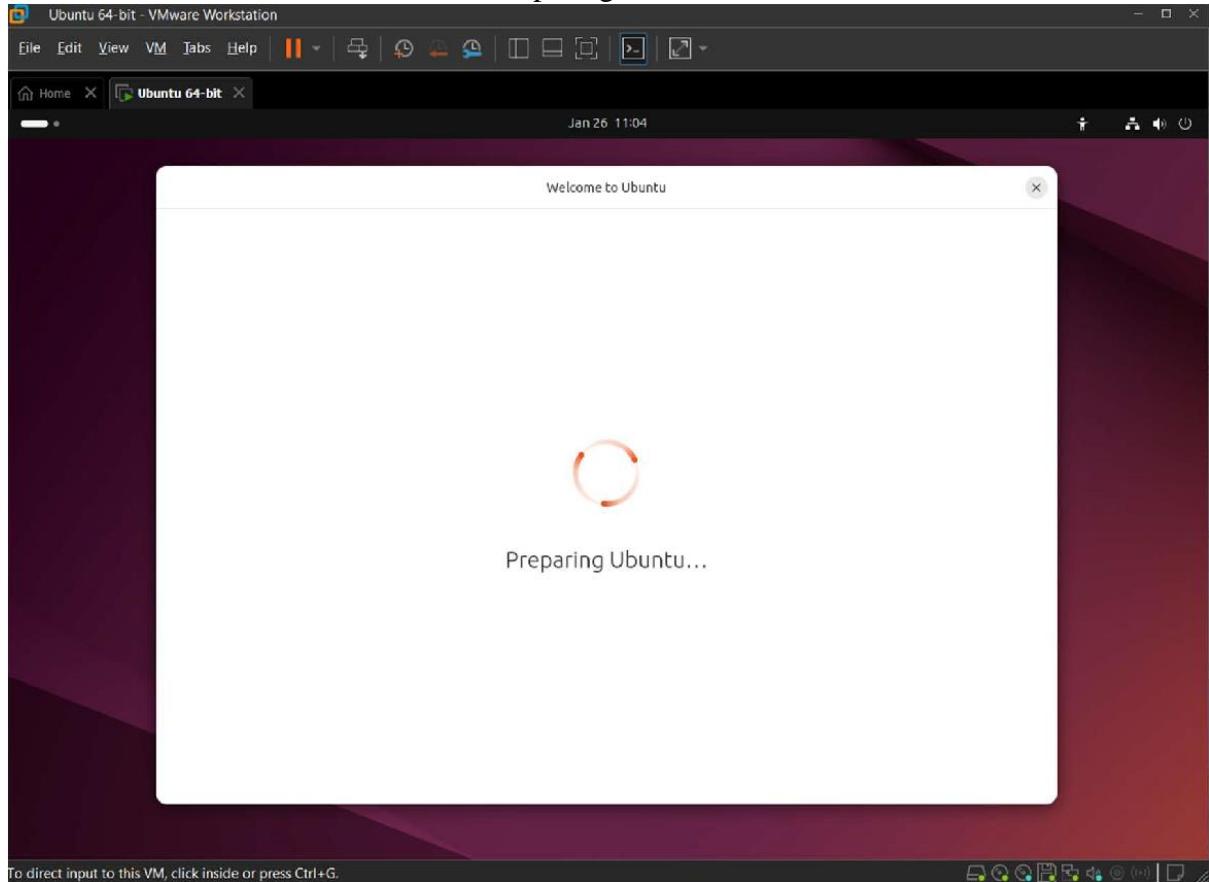
(In case You get an error like Intel VT not supported, Turn off the Memory Integrity from core isolation)





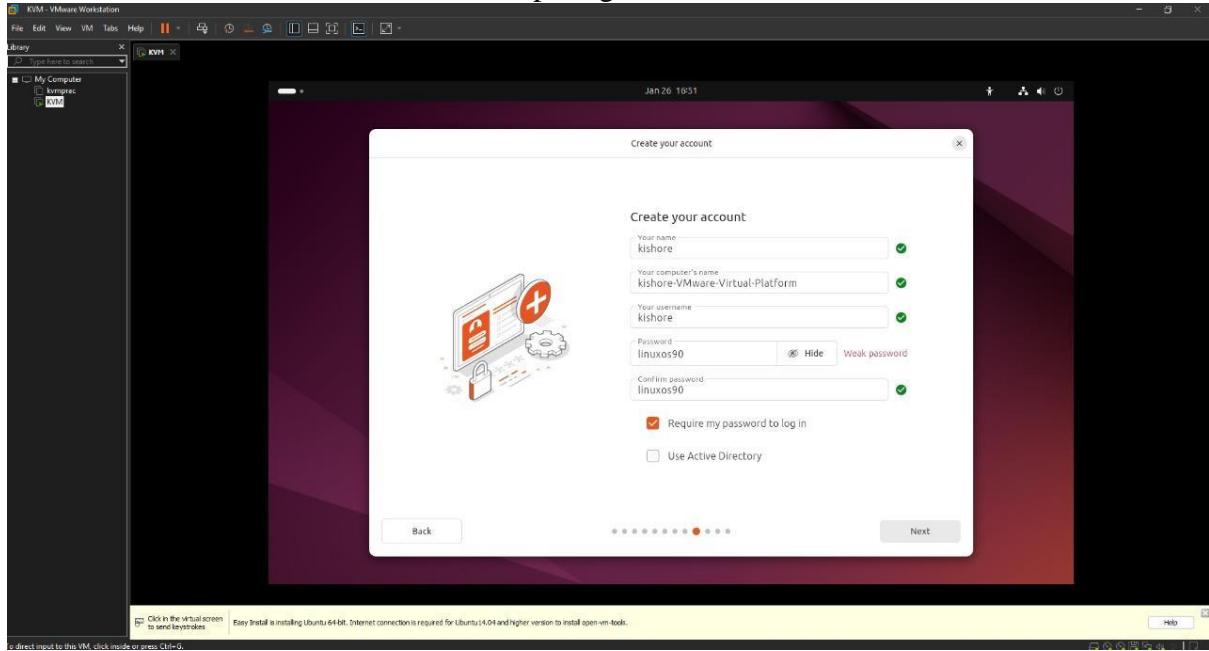
Ubuntu, a linux Distribution system will load.





Ubuntu Setup:

1. Choose Language then click next
2. Then in Accessibility Tab Click next
3. Keyboard Layout As English US then next
4. Connect to Internet then next
5. Tick Install ubuntu then next
6. Interactive then Default Installation
7. Install Proprietary Software (tick both options)
8. Then Tick Erase disk and install ubuntu
9. Then Fill up the following details:

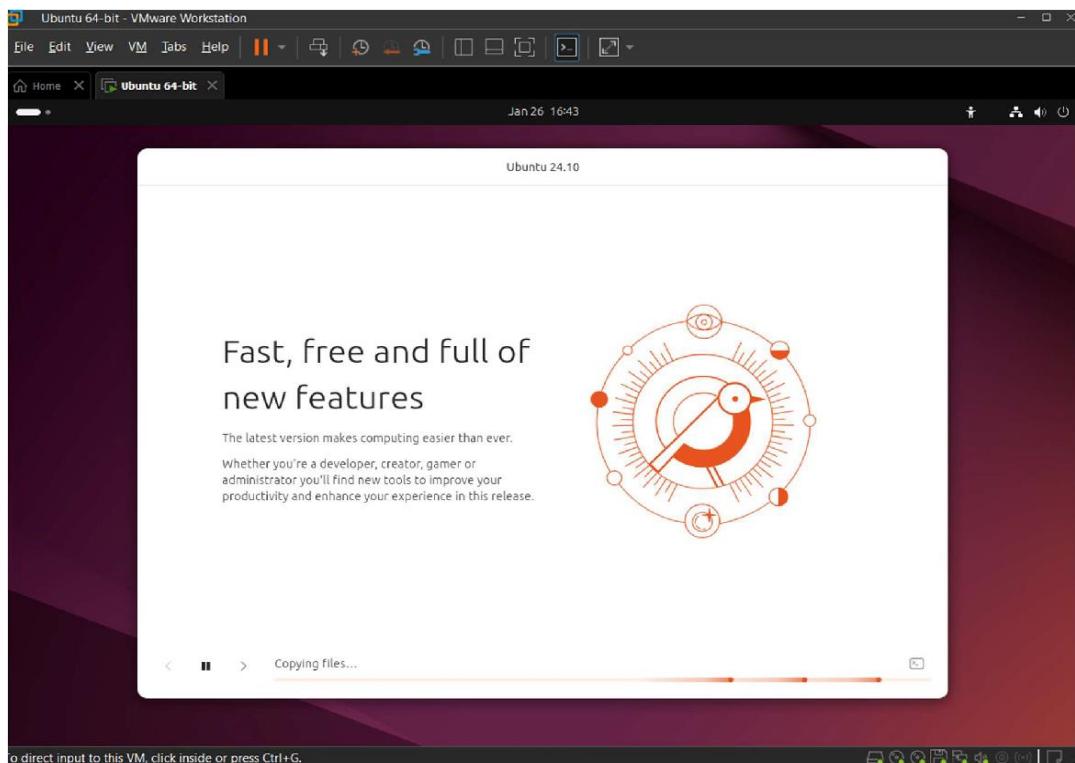


10. Then set the time zone and click next

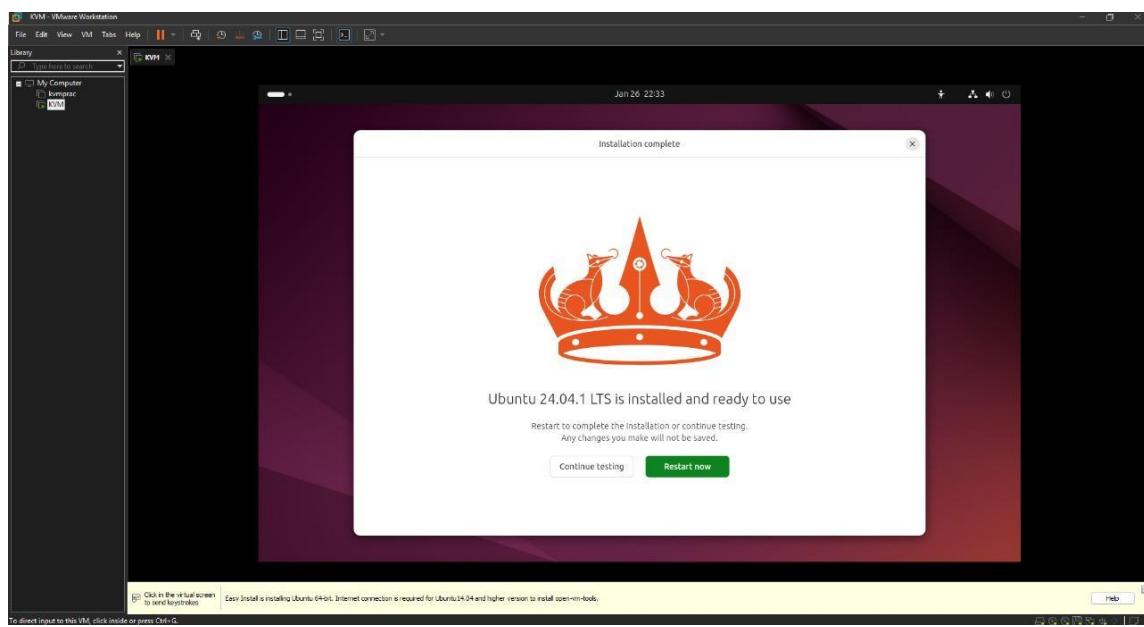
Location: Mumbai

Timezone: Asia/Kolkata

11. Then Following window appears, Click Install



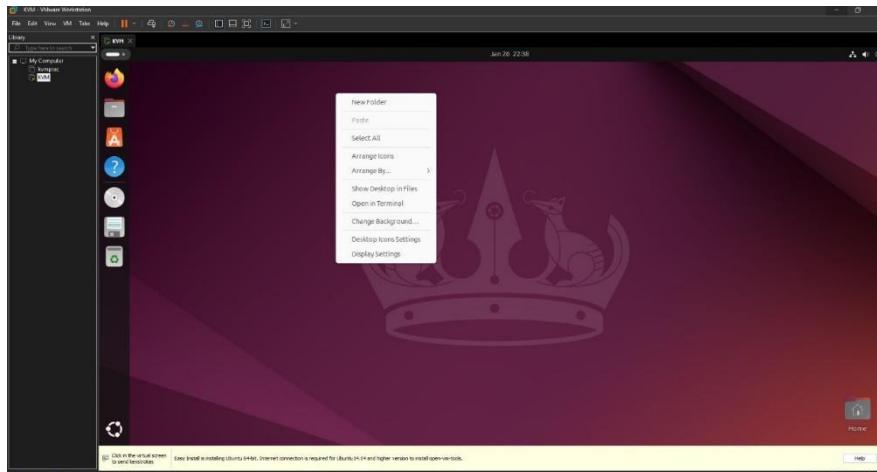
We get the following screen,



Click Restart.

Now login using your password.

II) KVM Setup Steps: Open Terminal



1. Update the System

sudo apt update && sudo apt upgrade -y

2. Check if Virtualization is Enabled:

sudo grep -c "svm|vmx" /proc/cpuinfo

3. Verify KVM Virtualization

Check if KVM virtualization is enabled by running command:

kvm-ok

If the kvm-ok command is not found, install the CPU

checker tool: **sudo apt install cpu-checker**

then type kvm-ok

Output should include:

INFO: /dev/kvm exists

KVM acceleration can be used.

4. Install KVM and Required Packages

```
sudo apt install qemu-kvm virt-manager libvirt-daemon-system libvirt-clients  
bridge-utils -y
```

5. Enable the Virtualization

Daemon Start and enable the
libvirt daemon:

```
sudo systemctl enable libvirtd
```

```
sudo systemctl start libvirtd
```

6. Check the Status of the Libvirt Daemon

Verify that the daemon is running:

```
sudo systemctl status libvirtd
```

7. Add Your User to KVM and Libvirt Groups

Replace your-username with your actual username and run the following
commands:

```
sudo usermod -aG kvm your-  
username
```

```
sudo usermod -aG libvirt your-  
username
```

```
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ sudo apt update && sudo apt upgrade -y
[sudo] password for cloud:
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://in.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:4 http://in.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:5 http://in.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [783 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [976 kB]
Fetched 1,885 kB in 2s (795 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
8 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following package was automatically installed and is no longer required:
python3-netifaces
Use 'sudo apt autoremove' to remove it.
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:
 libb64json1 libb64postproc57 libb64avcodec60 libb64util58 libb64wscale7 libb64resample4
 libb64avformat68 libb64vfilter9
Learn more about Ubuntu Pro at https://ubuntu.com/pro
The following upgrades have been deferred due to phasing:
 python3-distupgrade ubuntu-release-upgrader.core ubuntu-release-upgrader-gtk
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ sudo grep -c "svm\|vmx" /proc/cpuinfo
0
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ 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
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ sudo apt install cpu-checker
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
cpu-checker is already the newest version (0.7-1.3build2).
```

```
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ sudo apt install qemu-kvm virt-manager libvirt-daemon-system libvirt-clients bridge-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'qemu-system-x86' instead of 'qemu-kvm'
qemu-system-x86 is already the newest version (1:8.2.2+ds-0ubuntu1.5).
virt-manager is already the newest version (1:4.1.0-3ubuntu0.1).
libvirt-daemon-system is already the newest version (10.0.0-2ubuntu8.5).
libvirt-clients is already the newest version (10.0.0-2ubuntu8.5).
bridge-utils is already the newest version (1.7.1-1ubuntu2).
The following package was automatically installed and is no longer required:
python3-netifaces
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ 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
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ sudo apt install libvirt-daemon-system libvirt-clients libvirt-daemon
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
● libvirtd.service - libvirt legacy monolithic daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-01-26 19:42:17 IST; 17min ago
     Docs: man:libvirtd(8)
           https://libvirt.org/
   Main PID: 1467 (libvirtd)
      Tasks: 24 (limit: 32768)
        Memory: 25.4M (peak: 95.5M swap: 10.9M swap peak: 10.9M)
        CPU: 2.504s
      CGroup: /system.slice/libvirtd.service

```

```
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ sudo apt install qemu-kvm virt-manager libvirt-daemon-system libvirt-clients bridge-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'qemu-system-x86' instead of 'qemu-kvm'
qemu-system-x86 is already the newest version (1:8.2.2+ds-0ubuntu1.5).
virt-manager is already the newest version (1:4.1.0-3ubuntu0.1).
libvirt-daemon-system is already the newest version (10.0.0-2ubuntu8.5).
libvirt-clients is already the newest version (10.0.0-2ubuntu8.5).
bridge-utils is already the newest version (1.7.1-1ubuntu2).
The following package was automatically installed and is no longer required:
python3-netifaces
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ sudo systemctl enable libvirtd
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ sudo systemctl start libvirtd
cloud@cloud-VMware-Virtual-Platform:~/Desktop$ sudo systemctl status libvirtd
● libvirtd.service - libvirt legacy monolithic daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-01-26 19:42:17 IST; 17min ago
     Docs: man:libvirtd(8)
           https://libvirt.org/
   Main PID: 1467 (libvirtd)
      Tasks: 24 (limit: 32768)
        Memory: 25.4M (peak: 95.5M swap: 10.9M swap peak: 10.9M)
        CPU: 2.504s
      CGroup: /system.slice/libvirtd.service

```

8. Log Out and Re-login

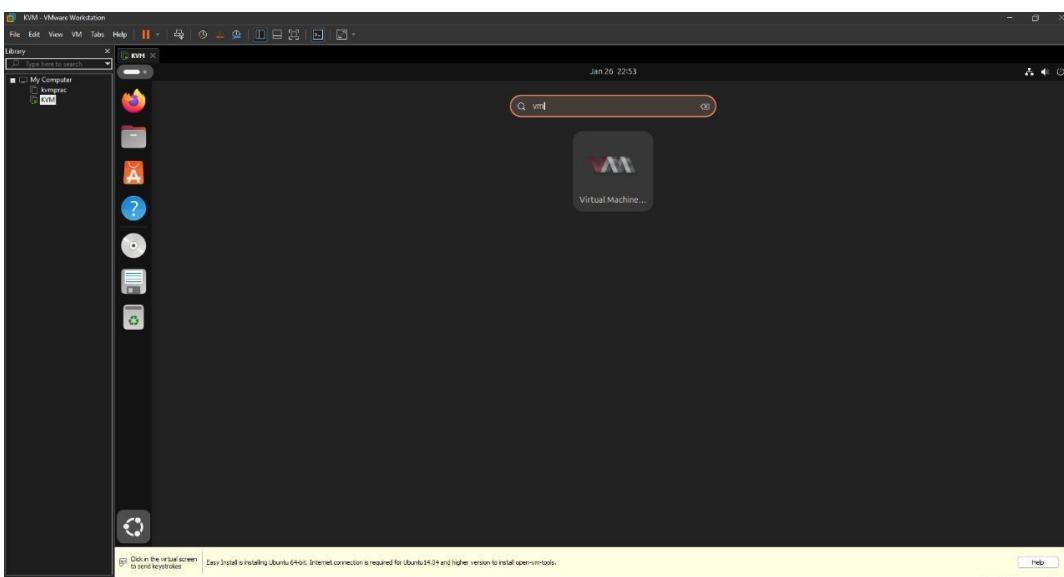
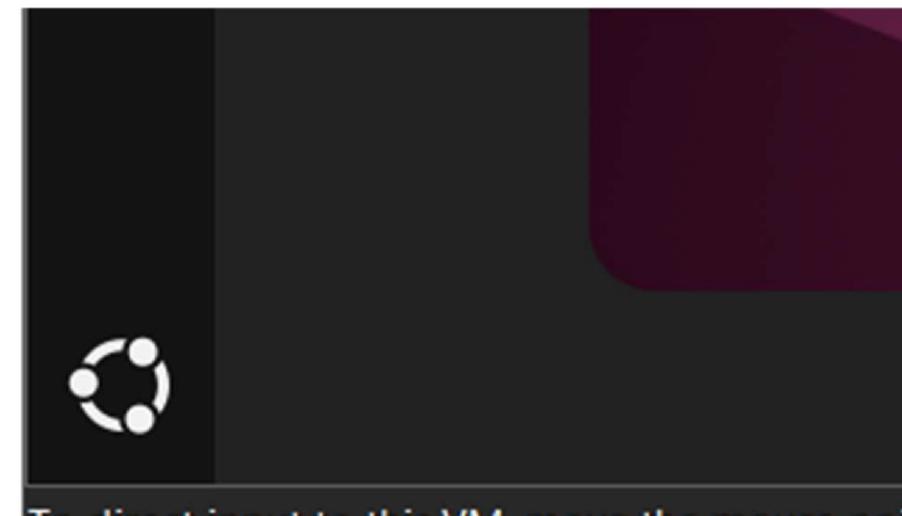
Log out from your system and log back in to apply group changes.

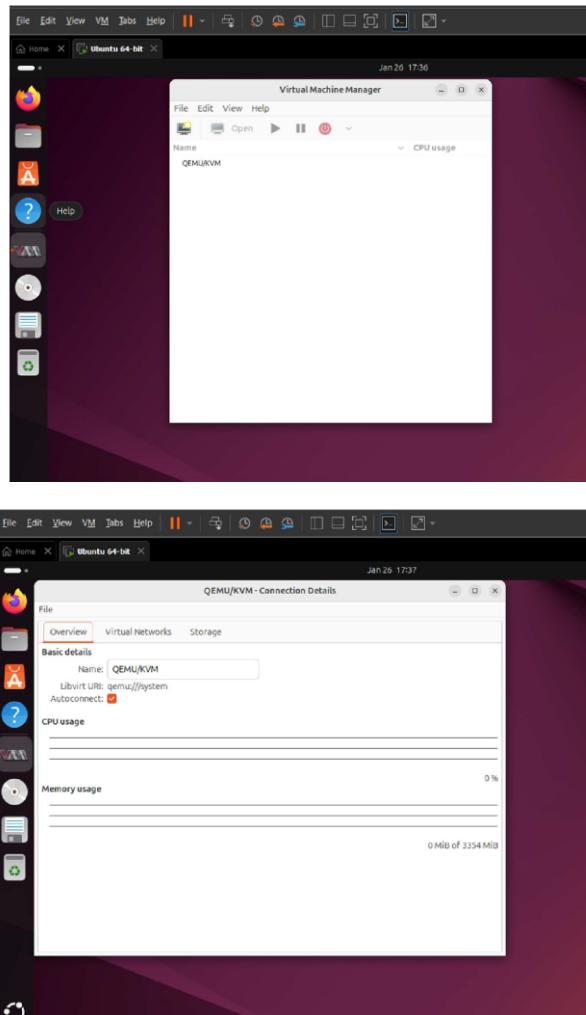
9. Run KVM Virtual Machine Manager

Search for "Virtual Machine Manager" in your system applications and launch it.

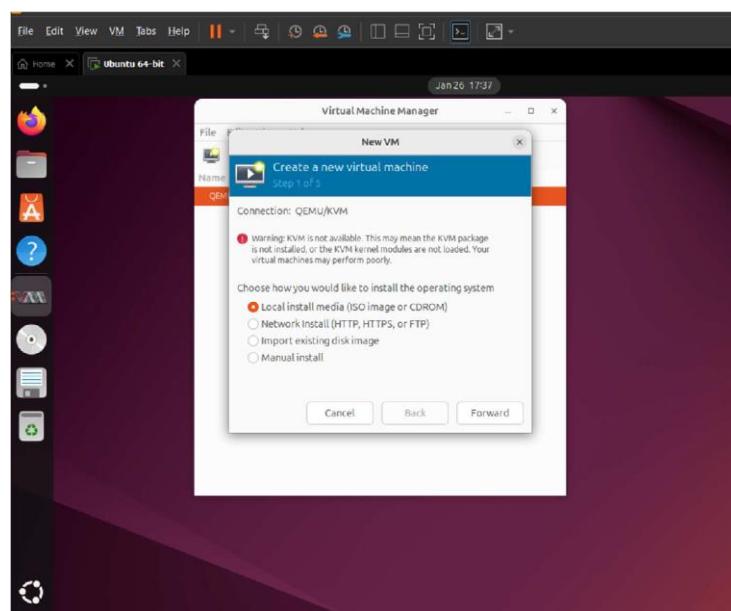
10. Prepare to Create Virtual Machines

In Virtual Machine Manager, click Create a New Virtual Machine to begin setting up virtual environments.

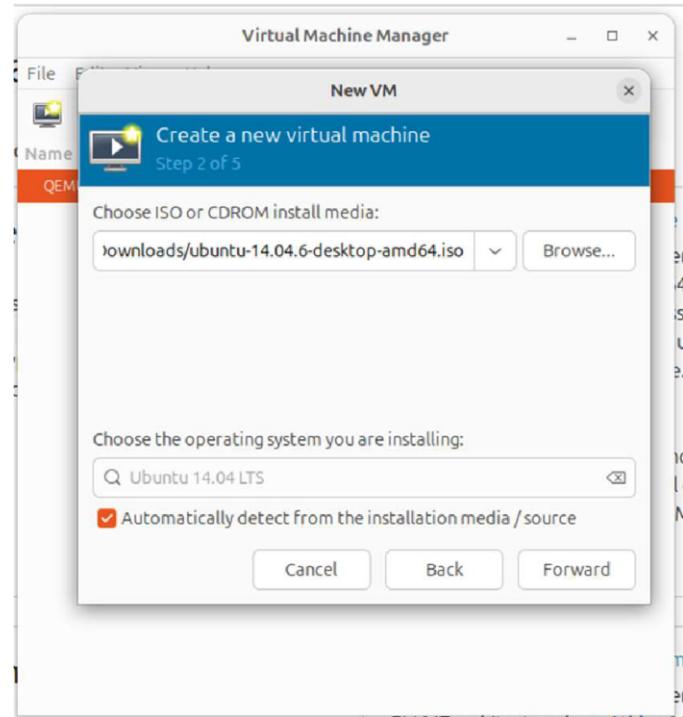




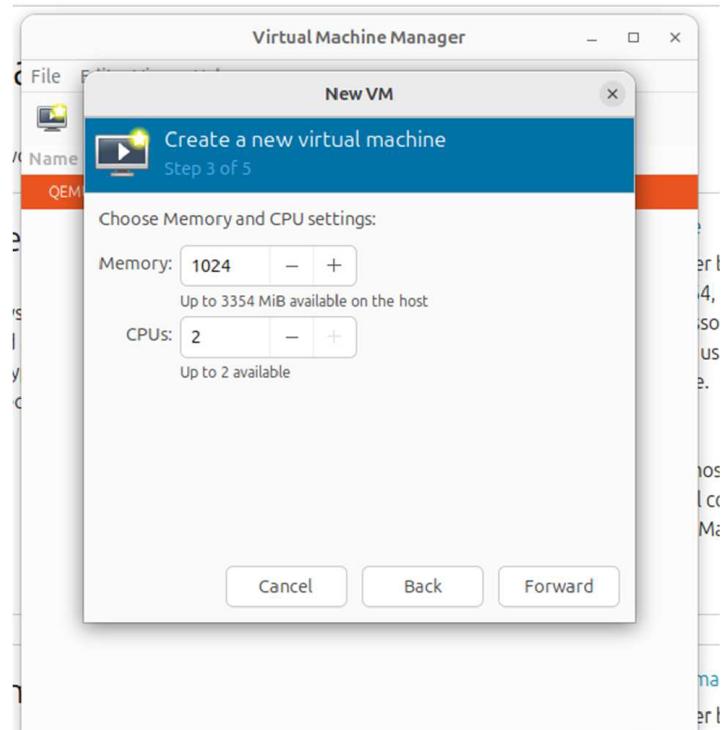
Now we can create new virtual machine from File>Create New Virtual Machine



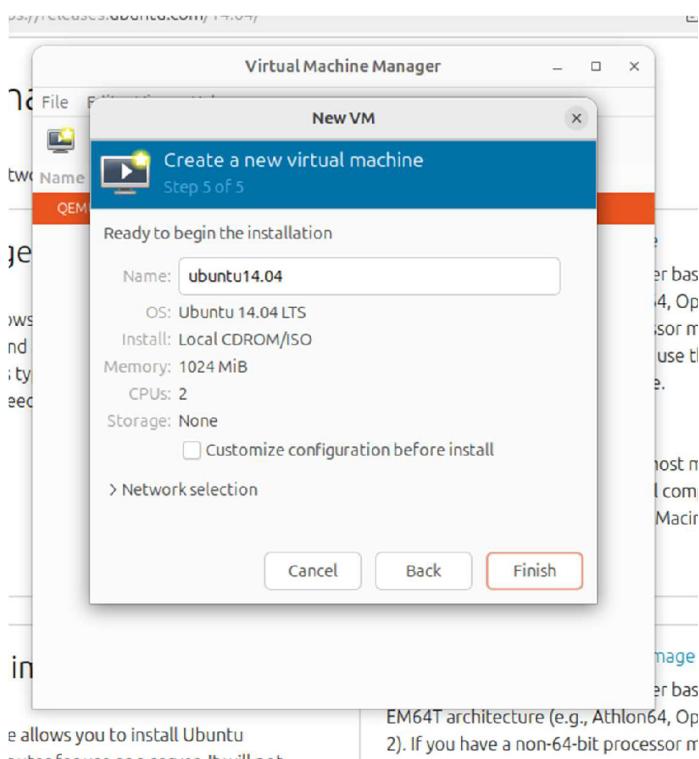
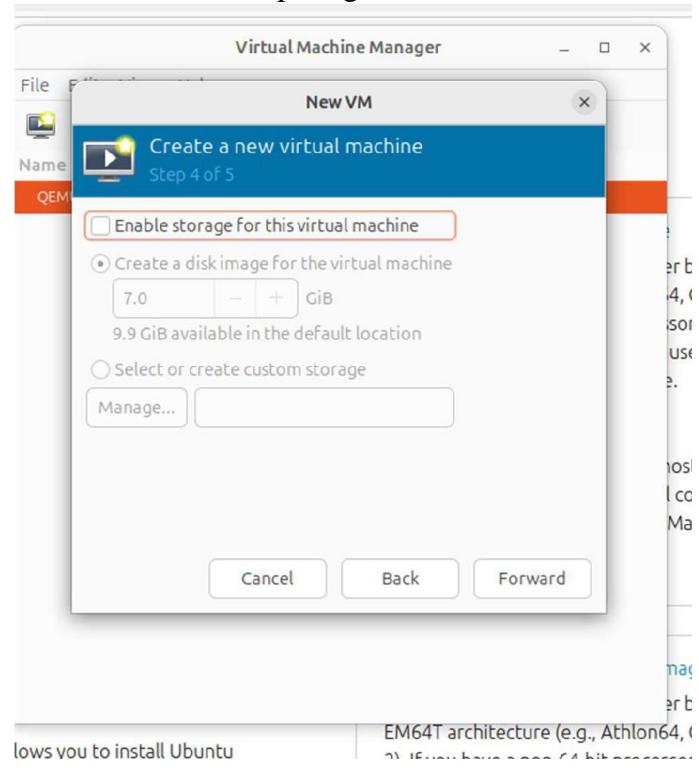
Then, Select the OS to install (eg:/windows, ubuntu etc)

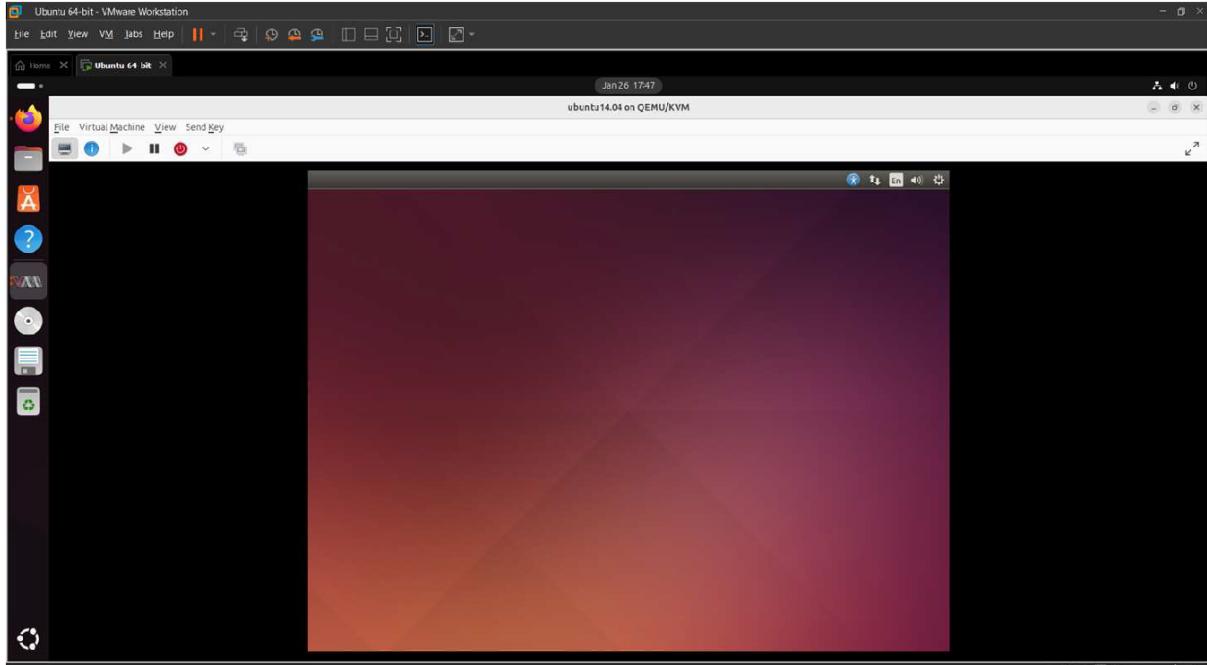


Then



Then





Practical No. 6

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

Software Tools Required

- **IDE or Code Editor:** Netbeans version 8.0.2
- **JDK:** java jdk version 8

Downloads Required:

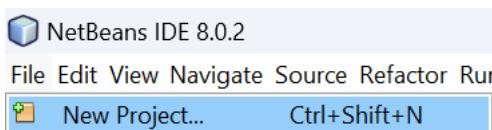
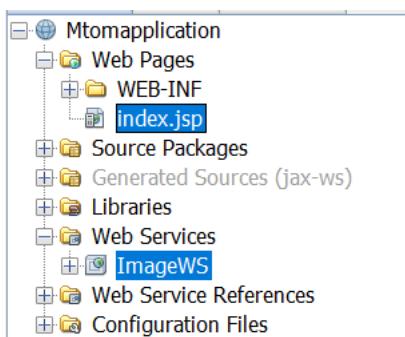
- Netbeans 8.0.2: <https://dlc-cdn.sun.com/netbeans/8.0.2/final/bundles/netbeans-8.0.2-windows.exe>
- Java JDK 8:
<https://www.oracle.com/in/java/technologies/javase/javase8u211-later-archive-downloads.html#license-lightbox>



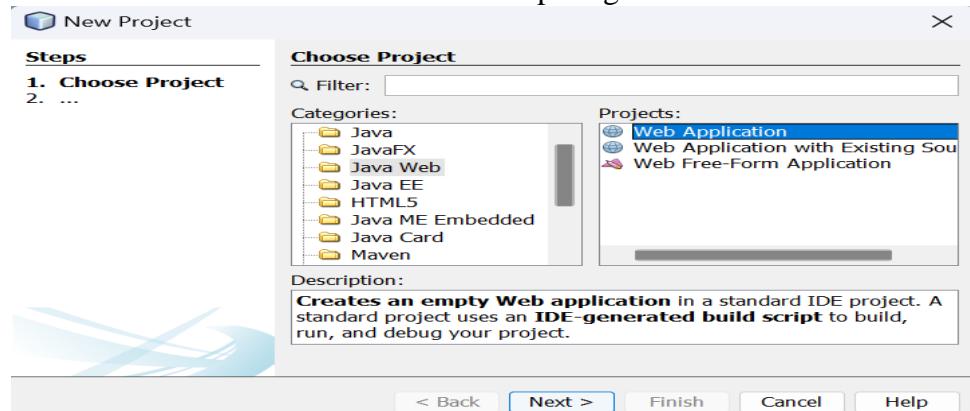
jdk1.8.0_431-202503
05T110059Z-001.zip

Project Name: Mtomapplication

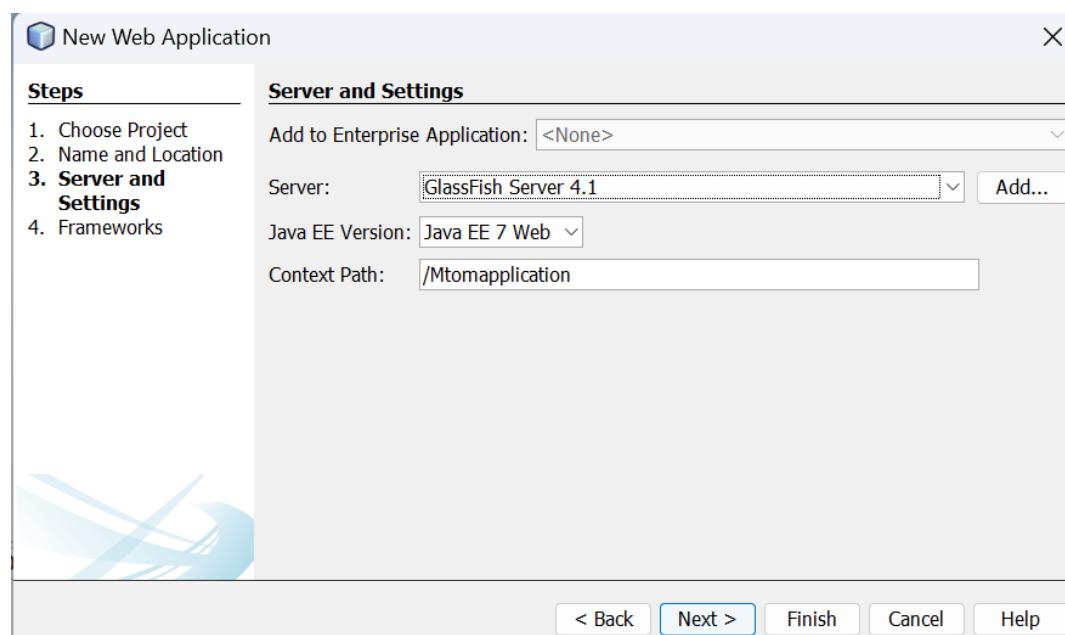
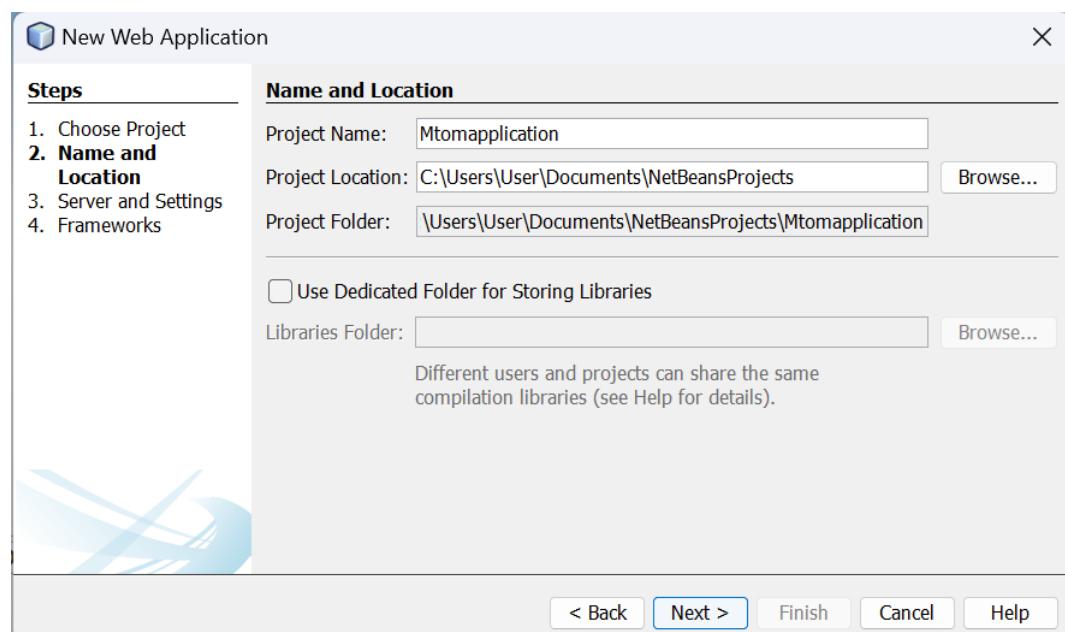
Project Architecture:

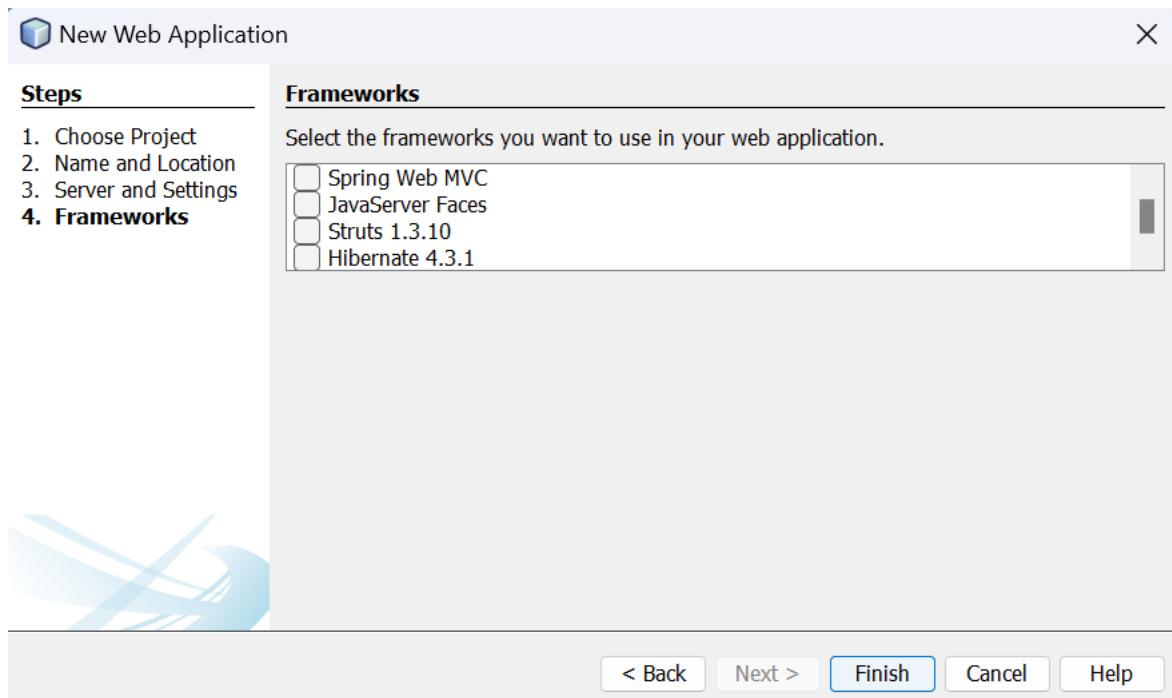


Select Java Web >Web Application



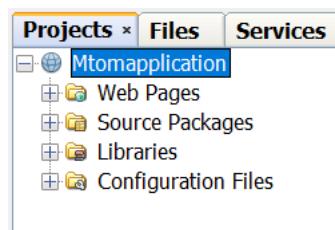
Click Next



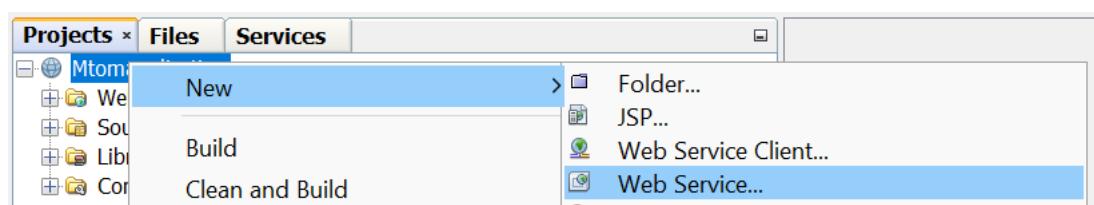


Click Finish

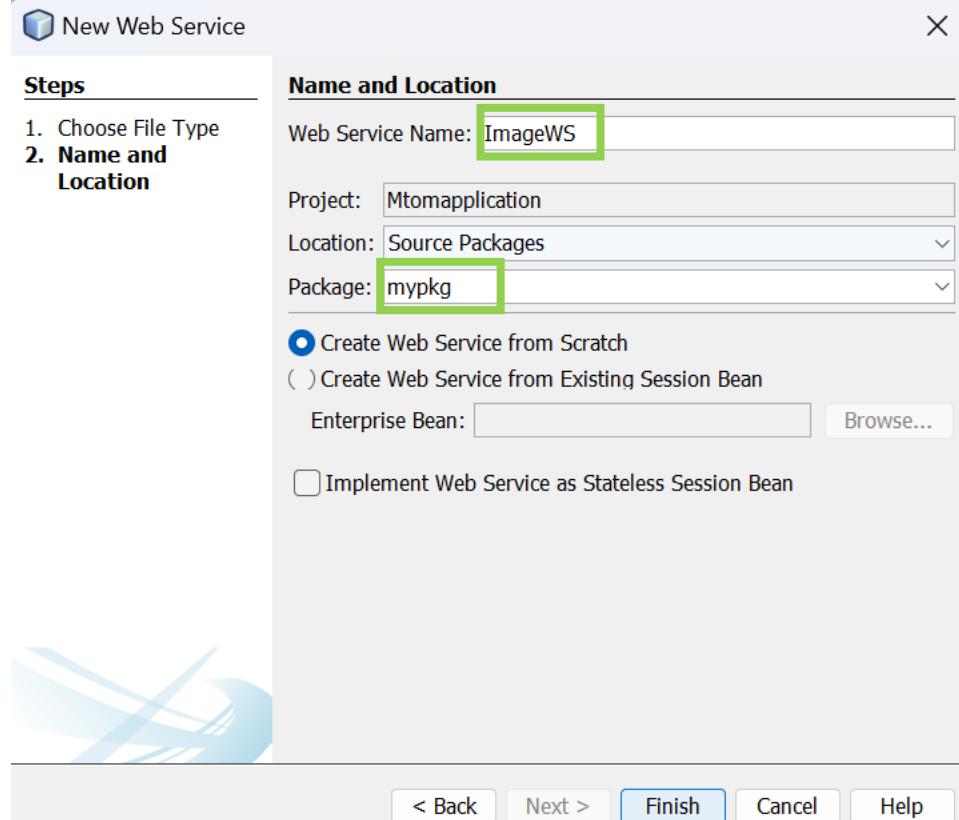
Expand Project Architecture



Right Click on Mtommapplication Project New > Web Service



Name: ImageWS and Package: mypkg and Finish



Filename: ImageWS.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package mypkg;
```

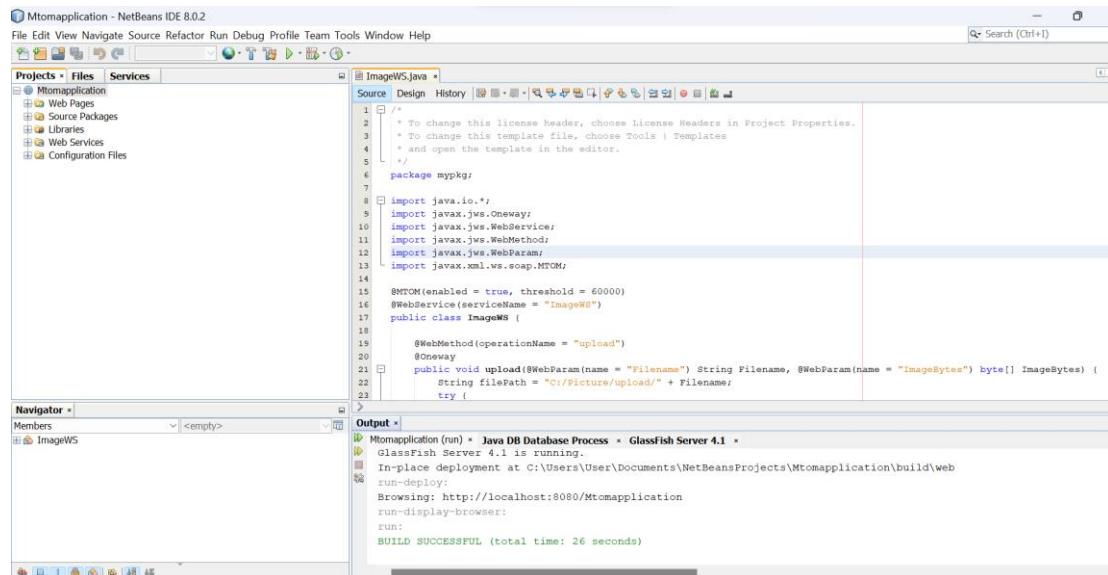
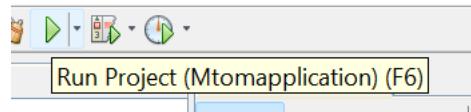
```
import java.io.*;
import javax.jws.Oneway;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.xml.ws.soap.MTOM;
```

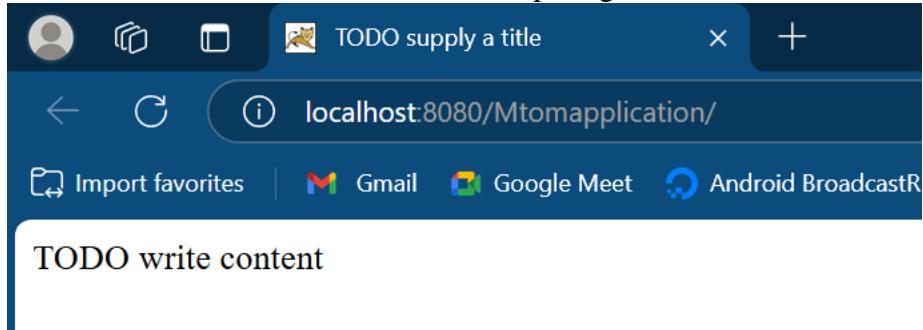
```
@MTOM(enabled = true, threshold = 60000)  
@WebService(serviceName = "ImageWS")  
public class ImageWS {  
  
    @WebMethod(operationName = "upload")  
    @Oneway  
    public void upload(@WebParam(name = "Filename") String Filename,  
    @WebParam(name = "ImageBytes") byte[] ImageBytes) {  
        String filePath = "C:/Picture/upload/" + Filename;  
        try {  
            FileOutputStream fos = new FileOutputStream(filePath);  
            BufferedOutputStream bos = new BufferedOutputStream(fos);  
            bos.write(ImageBytes);  
            bos.close();  
            System.out.println("Received file: " + filePath);  
        } catch (Exception ex) {  
            ex.printStackTrace();  
        }  
    }  
    @WebMethod(operationName = "download")  
    public byte[] download(@WebParam(name = "Filename") String Filename) {  
        String filePath = "C:/Picture/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) {
        ex.printStackTrace();
        return null;
    }
}
```

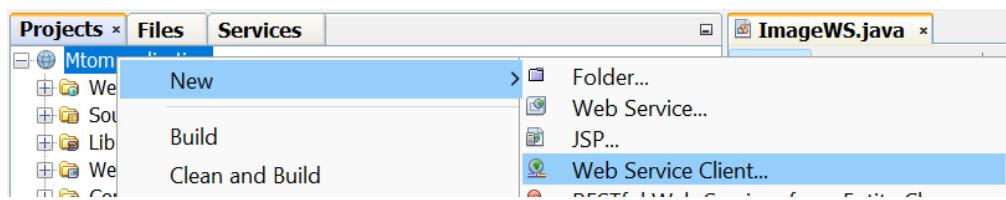
Save File.

Run Project

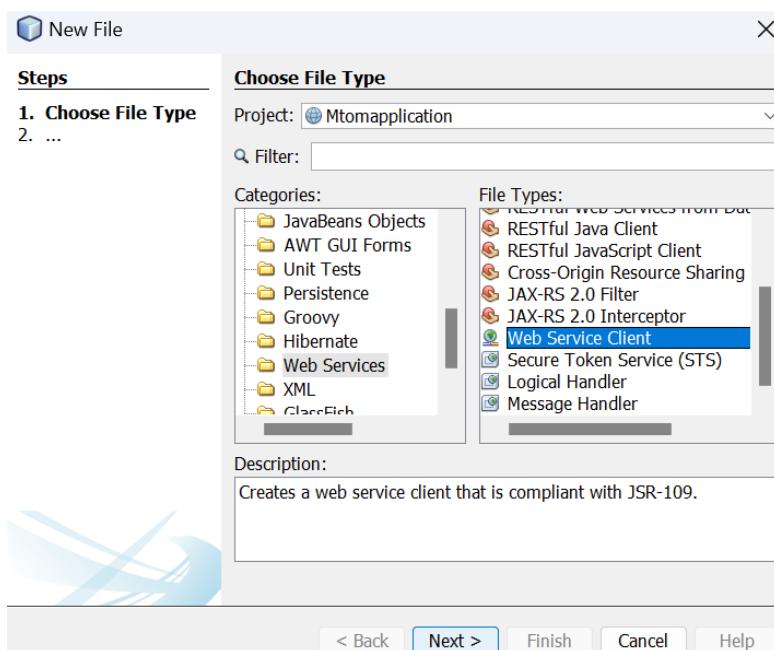


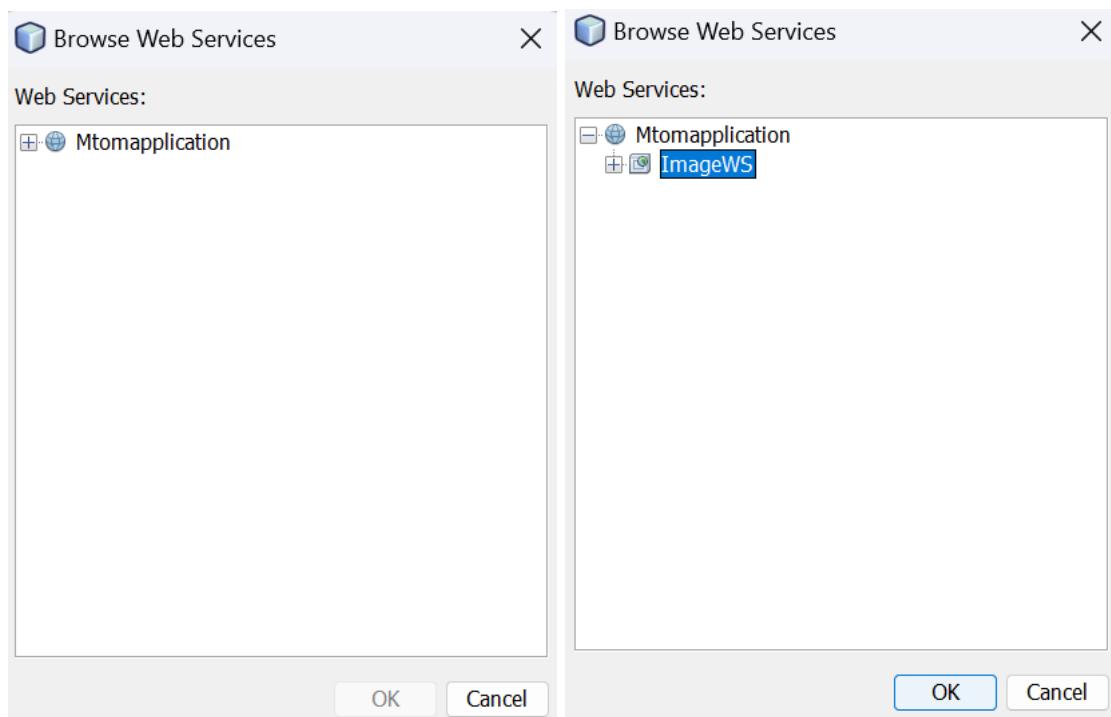
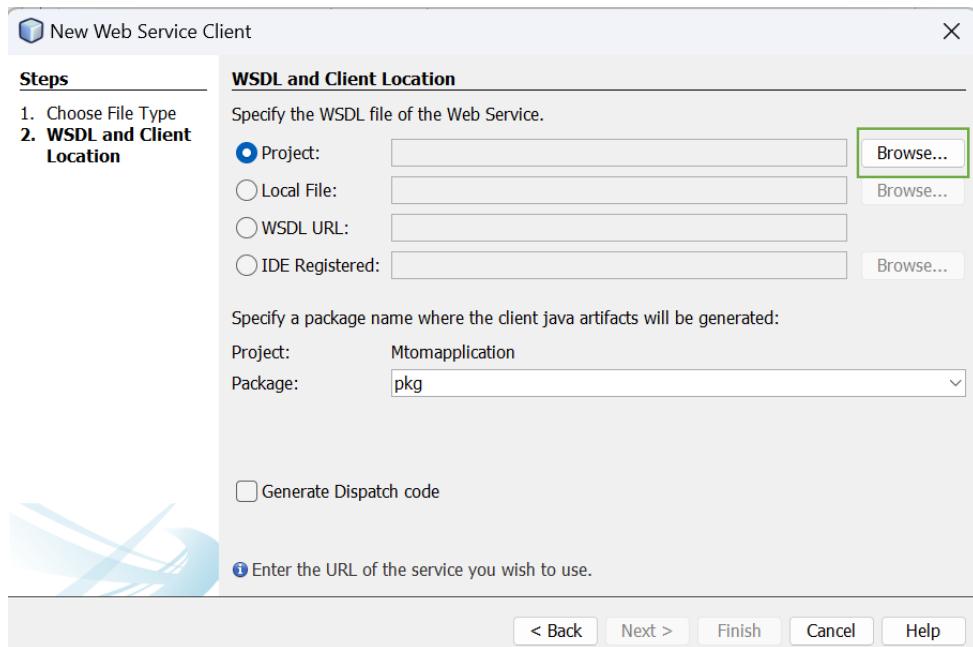


Right Click on Mtomapplication > New > Web Service Client.

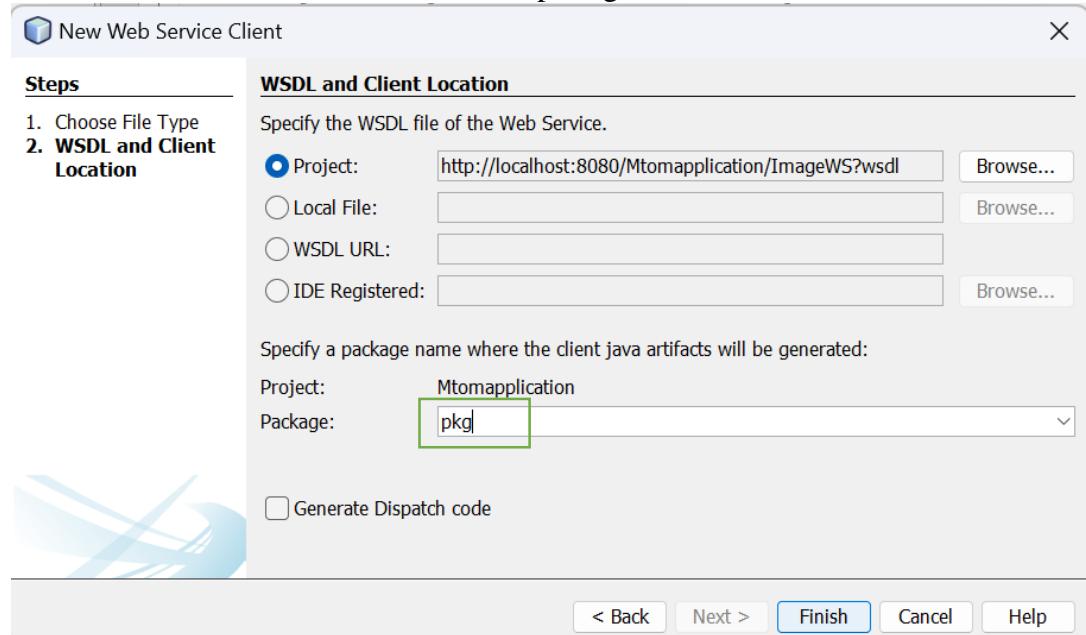


If not available click on Others



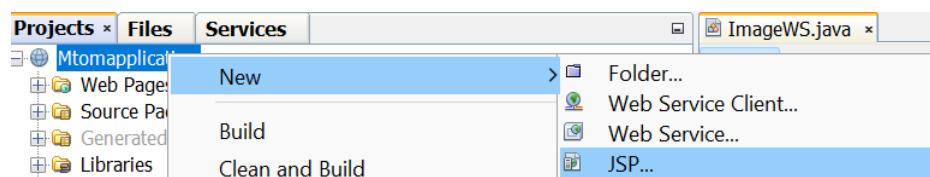


Package: pkg

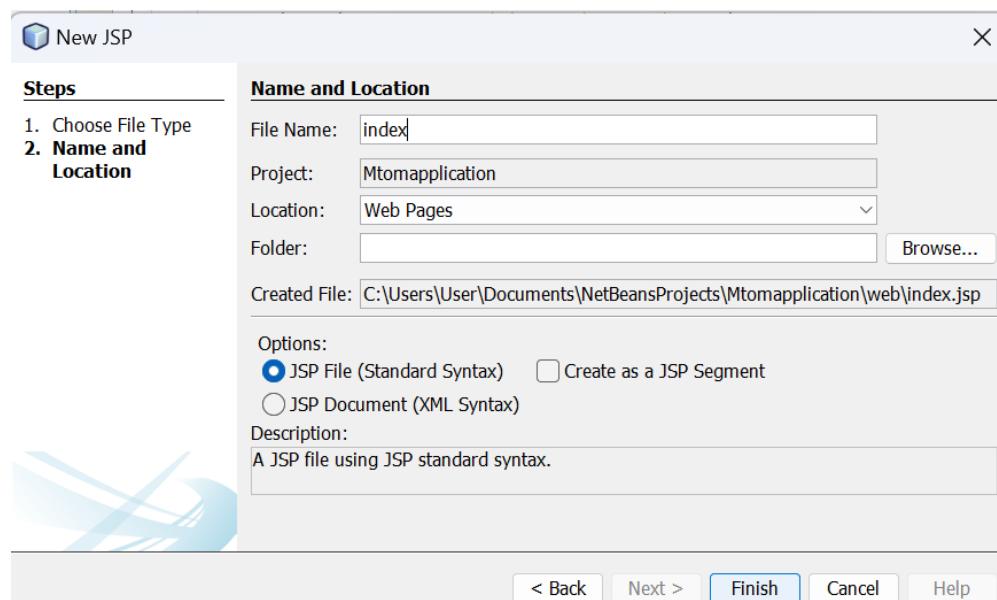


Finish

New > JSP



Name: index and click Finish



Filename: index.jsp

<%--

Document : index

Created on : Feb 3, 2025, 1:25:10 PM

Author : User

```
--%>
```

```
<%@page import="java.io.BufferedOutputStream"%>
<%@page import="java.io.FileOutputStream"%>
<%@page import="java.io.FileInputStream"%>
<%@page import="java.io.BufferedInputStream"%>
<%@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:/Picture/abcd2.jpg"; // Assuming this is the correct file
path
```

```
File file=new File(filePath);

FileInputStream fis=new FileInputStream(file);

BufferedInputStream bis=new BufferedInputStream(fis);

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

ex.printStackTrace();

}

%>

<%-- 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

String filename = "abcd2.jpg"; // Assuming this is the correct file name
```

```
// Initialize filePath

String filePath = "C:/Picture/download/" + filename; // Assuming this is the
correct download path

// Invoke the download method and get the file bytes

byte[] fileBytes = port.download(filename);

// Write the downloaded bytes to a file

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

ex.printStackTrace();

}

%>

<%-- end web service invocation --%><hr/>

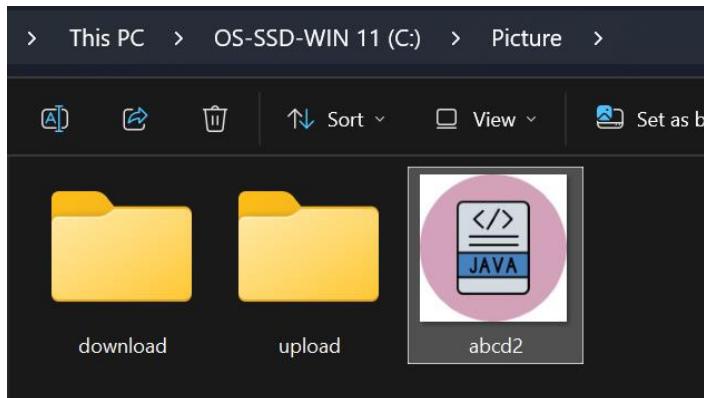
</body>

</html>
```

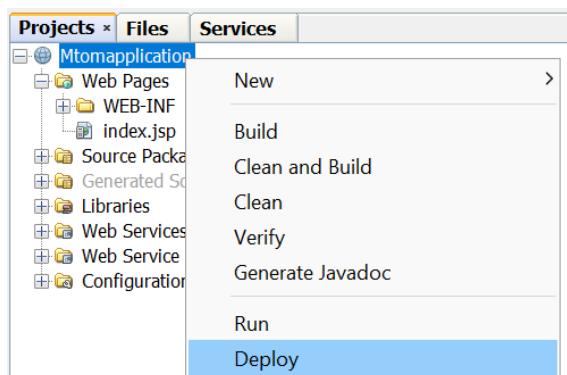
Create a Folder Name: Picture inside

Create 2 Folders name: download and upload and an image with name abcd2 or any name with proper extension

Here image name and extension: abcd2.jpg



Deploy Project



Then Run Project or F6

