SHRI G.P.M. DEGREE COLLEGE OF SCIENCE & COMMERCE

SHRI G.P.M. DEGREE COLLEGE OF SCIENCE & COMMERCE.

(COMMITED TO EXCELLENCE IN EDUCATION)

CERTIFICATE

This is to certify that Mr._____

a student of TY**.BSC-CS** Roll no: _____ has completed the required number of practicals in the subject of _____

as prescribed by the UNIVERSITY OF MUMBAI under my supervision during the academic year 2024-2025.

……..……..…..

Prof. Incharge

……

Principal

……..……

External Examiner

……...…..……

Course Co-ordinator

……..

Date

……

College Stamp

Professor Name: Prof. Vivek Maurya	Class : TY.BSC-CS Semester : Sem VI (2024-2025)
Course code : USCSP602	Subject: Cloud Computing and Web Services

Index

****

Theory-1

Introduction to Web Services

Web Service:
A web service is a software system designed to support interoperable machine-to-machine interaction over a network.
Essentially, it's a way for different applications to communicate with each other over the internet.
Web services can use various protocols for communication, such as SOAP (Simple Object Access Protocol) or REST.

API (Application Programming Interface):
An API is a set of rules and specifications that allow software applications to communicate and interact with each other.
It defines how different software components should interact.
APIs enable developers to access and use the functionality of other applications or services.
Web services utilize APIs. So any web service provides an API.

REST (Representational State Transfer):
REST is an architectural style for web communication that utilizes URLs to identify resources and standard HTTP methods (GET, POST, PUT, DELETE) to manipulate them. Each request is stateless, meaning it contains all necessary information for processing.
REST is based on request response model

REST API (Representational State Transfer Application Programming Interface):
A REST API is a specific type of web API that adheres to the architectural principles of REST.
REST is an architectural style that emphasizes:
Statelessness: Each request from a client to a server must contain all the information needed to understand and process the request.
Client-server architecture: Client and server are seperate.
Use of standard HTTP methods: Such as GET, POST, PUT, and DELETE, to perform operations on resources.
Resource identification through URLs: Each resource is uniquely identified by a URL.
Use of representations: Data is transferred in various formats, such as JSON or XML.
REST APIs are widely used because they are lightweight, scalable, and easy to use.

In summary:
A web service is a way for applications to communicate over a network.
An API defines how applications interact.
REST is a stateless web communication style using URLs and HTTP methods for resource interaction.
A REST API is a specific type of web API that follows the REST architectural style.

Practical-1: Define a simple services like Converting Rs into Dollar and Call it from different platform

Software Tools Required:

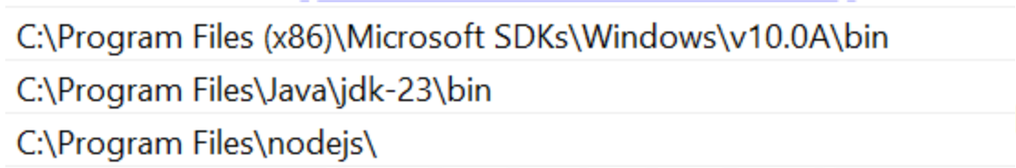
- **Code Editor**: Visual Studio Code (VS Code)
- **Framework**: Express.js (web framework)
- **Runtime Environment**: Node.js
- **Programming Languages**: JavaScript (for Node.js and Express.js), Python, Java, C# (for .NET)

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
- Python: Download Python | Python.org

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:



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\Saud>dotnet --version
9.0.101

C:\Users\Saud>java --version
java 21.0.6 2025-01-21 LTS
Java(TM) SE Runtime Environment (build 21.0.6+8-LTS-188)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.6+8-LTS-188, mixed mode, sharing)

C:\Users\Saud>node -v
v22.14.0
```

Here all versions are shown so path is set by checking versions for Nodejs, Java JDK and Dotnet SDK.
Then run this command in powershell:

Set-ExecutionPolicy -Scope CurrentUser RemoteSigned

To Implement the Service:

a. Create a folder `rest-service` and cd to rest-service
b. Run this in `rest-service` folder: npm i express axios
c. Create file: `server.js` and write the code in it

Source code: To create a simple API that takes an amount in Indian Rupees and returns the equivalent amount in US Dollars

Server side: server.js

```
const express = require('express')
const app = express()
```

```
app.get('/convert/:amount', (req , res) => {
    const inr = +req.params.amount;
    let usd = inr * 0.0116650131;
    usd = usd.toPrecision(4);
    res.json({ 'USD' : usd })
})

app.listen(3000, ()=>{
    console.log('server is running at port 3000')
})
```

Run the code: **node server.js**

Output:

```
PS C:\Users\Saud\node_proj\rest-service> node server
server is running at port:  3000
```

Client side: **Python:** create a python file in the same directory i.e. `rest-service`

```
import requests as req
res = req.get('http://localhost:3000/convert/100')
d = res.json()
print(d)
```

Output:

```
PS C:\Users\Saud\node_proj\rest-service> py .\p_client.py
{'USD': '1.167'}
```

Client side: **Java:** create a JAVA file in the same directory

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

public class JavaClient {
    public static void main(String[] args) {
        double inrAmount = 100;
        String url = "http://localhost:3000/convert/" + inrAmount;

        HttpClient client = HttpClient.newHttpClient();
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(url))
                .build();

        try {
                        HttpResponse<String>  response  =  client.send(request,
HttpResponse.BodyHandlers.ofString());
            System.out.println("Response: " + response.body());

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
PS C:\Users\Saud\node_proj\rest-service> java .\JavaClient.java
Response: {"USD":"1.167"}
```

Client Side: .Net (C#)

1. In VS Code terminal to create a new console project run this command:
   dotnet new console -n API_test
2. Goto the folder by this command:
   cd API_test
3. Modify Program.cs (default file in the API_test) as given below: Program.cs

```
using System;
using System.Net.Http;
using System.Threading.Tasks;
class Program
{   static async Task Main(string[] args)
    {
        double inrAmount = 100;
        string apiUrl = $"http://localhost:3000/convert/{inrAmount}";

        using (HttpClient client = new HttpClient())
        {
            try
            {
                HttpResponseMessage response = await
client.GetAsync(apiUrl);
                response.EnsureSuccessStatusCode();
                string responseBody = await
response.Content.ReadAsStringAsync();
                Console.WriteLine($"API Response: {responseBody}");
            }
            catch (HttpRequestException ex)
            {
                Console.WriteLine($"Error calling API: {ex.Message}");
            }
        }
    }
}
```

4. Run the code: (Run this command in the API_test folder)

Output:

```
PS C:\Users\Saud\node_proj\rest-service\API_test> dotnet run
API Response: {"USD":"1.167"}
```

Result and Discussion: We have successfully implemented a simple web service that converts INR to USD.

Learning Outcomes:
1. Understood Representational State Transfer .
2. Understood REST Application Programming Interface.

Course Outcomes:
1. We have learned about Web Service, REST, and API.
2. We have learned about the http request and response model.

Conclusion:
We have successfully implemented a simple web service and understood REST API.

Viva Question:
1. What is web service?
2. What is HTTP?
3. What is REST?
4. What is API?

For Faculty Use

Correction Parameters	Formative Assessment [ ]	Timely completion practical [ ]	Attendance Learning Attitude[ ]

Theory-2
SOAP

1. SOAP (Simple Object Access Protocol):

- SOAP is a protocol for exchanging structured information in the implementation of web services.
- It relies on XML for its message format.
- SOAP can operate over various transport protocols, such as HTTP, SMTP, and TCP.
- It's known for its strict standards and emphasis on security and reliability.
- Historically it was very common in enterprise level applications.

2. XML (Extensible Markup Language):

- XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
- It's widely used for representing and transporting data between different systems.
- In the context of web services, XML is used to structure the messages exchanged between applications.
- XML provides a flexible way to define data structures, making it suitable for various applications.
- XML is the message format that SOAP uses.

Key Relationship:

- SOAP uses XML as its message format. This means that SOAP messages are structured using XML tags, which define the data being exchanged.

In essence, XML provides the structure, and SOAP provides the rules for how that structured data is exchanged.

SOAP vs REST:

SOAP:

- Is a strict protocol.
- Uses XML.
- More complex, often for enterprise-level security.

REST:

- Is an architectural style.
- Can use various data formats (like JSON).
- Simpler, more flexible, and widely used for web APIs.

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

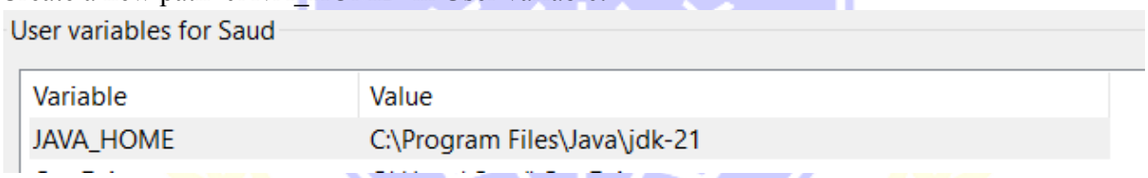
Downloads Required:

- 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)

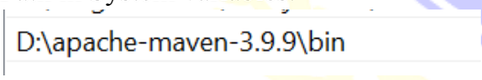
After Downloading JDK and Maven, Set the Path in Environment Variable in User Variables > Path > Edit > New and Paste the Path for JDK and Maven and check the versions for both in command prompt.

Demonstration:

Create a new path `JAVA_HOME` in User variable:



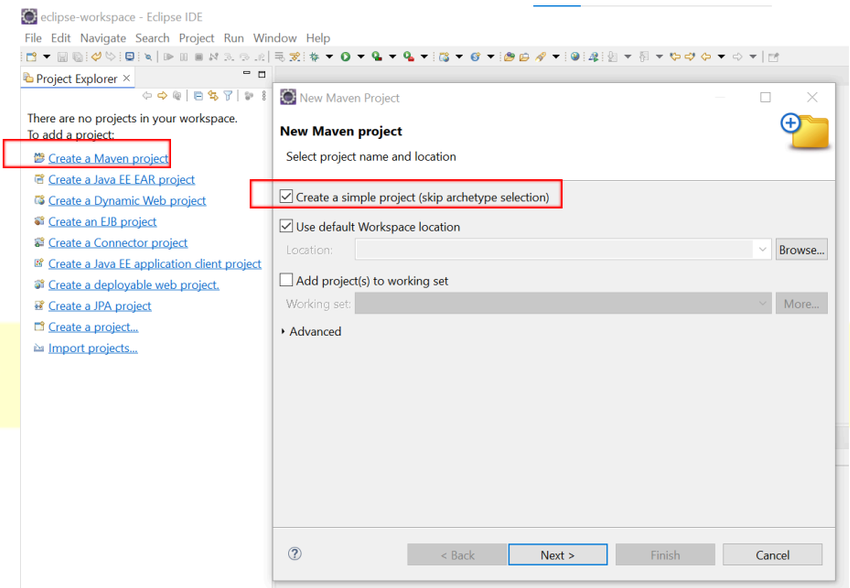
Path in System variables:



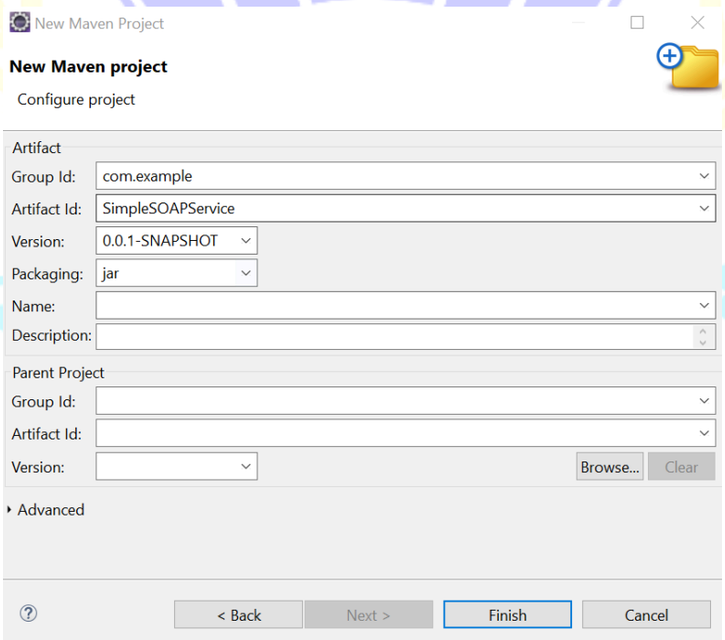
Version checked in command prompt

```
C:\Users\Saud>mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: C:\Program Files\apache-maven-3.9.9-bin\apache-maven-3.9.9
Java version: 21.0.6, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-21
Default locale: en_IN, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

C:\Users\Saud>java --version
java 21.0.6 2025-01-21 LTS
Java(TM) SE Runtime Environment (build 21.0.6+8-LTS-188)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.6+8-LTS-188, mixed mode, sharing)
```

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:

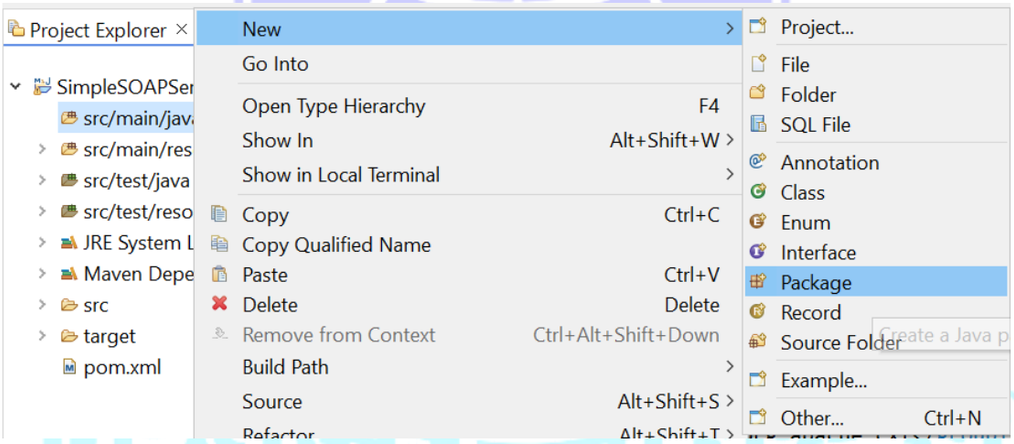
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>SimpleSOAPService</artifactId>
  <version>0.0.1-SNAPSHOT</version>
```

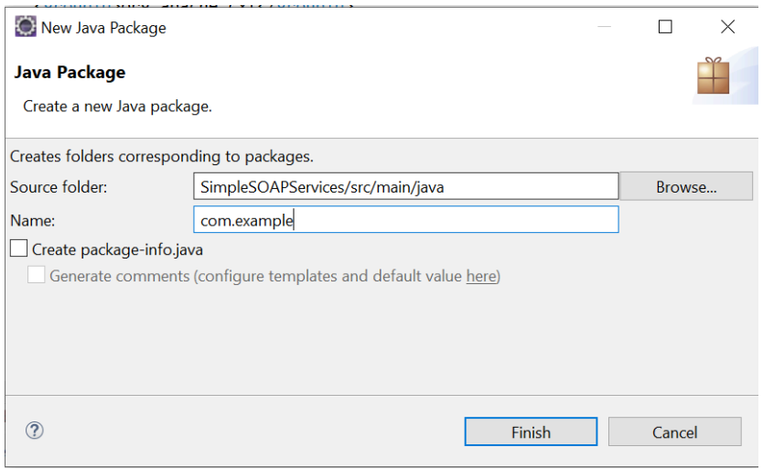
```
<dependencies>
    <!-- Apache CXF for JAX-WS -->
    <dependency>
      <groupId>org.apache.cxf</groupId>
      <artifactId>cxf-rt-frontend-jaxws</artifactId>
      <version>3.4.5</version>
    </dependency>

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

After adding dependencies Save it by Ctrl+S

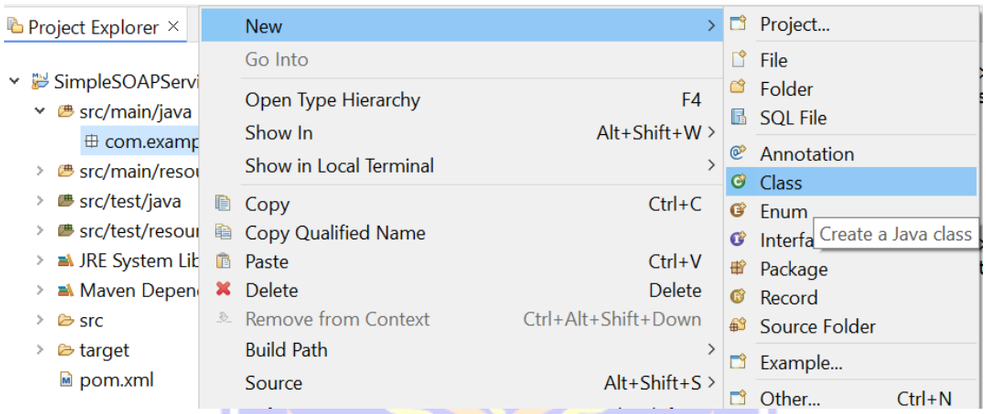
Step 4: Create the Service Interface
Now, create a simple SOAP service interface to define the operations.

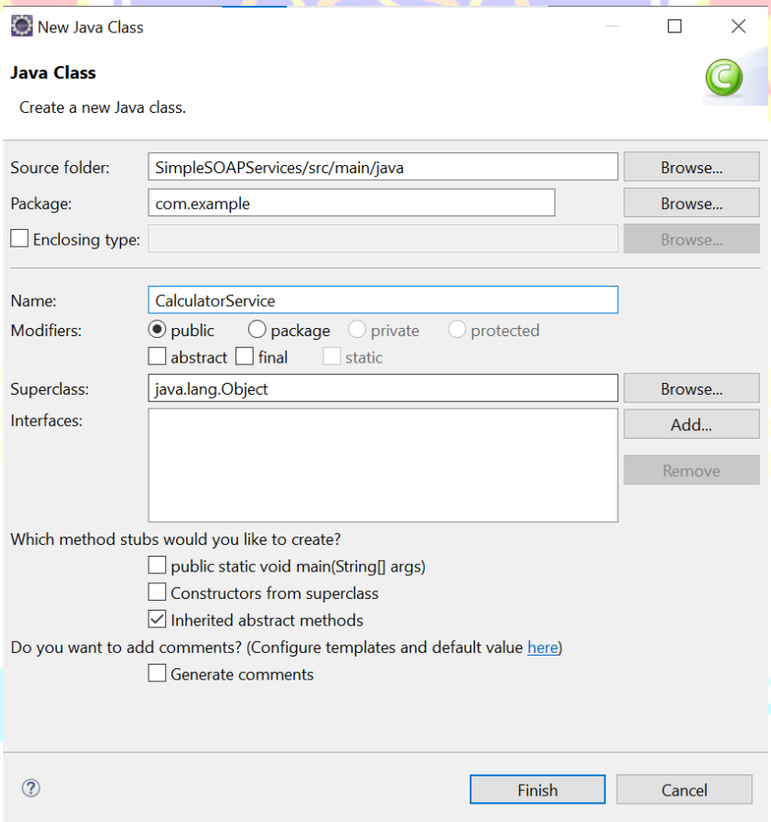
1. **Right-click** on src/main/java → **New** → **Package** and name it com.example





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.

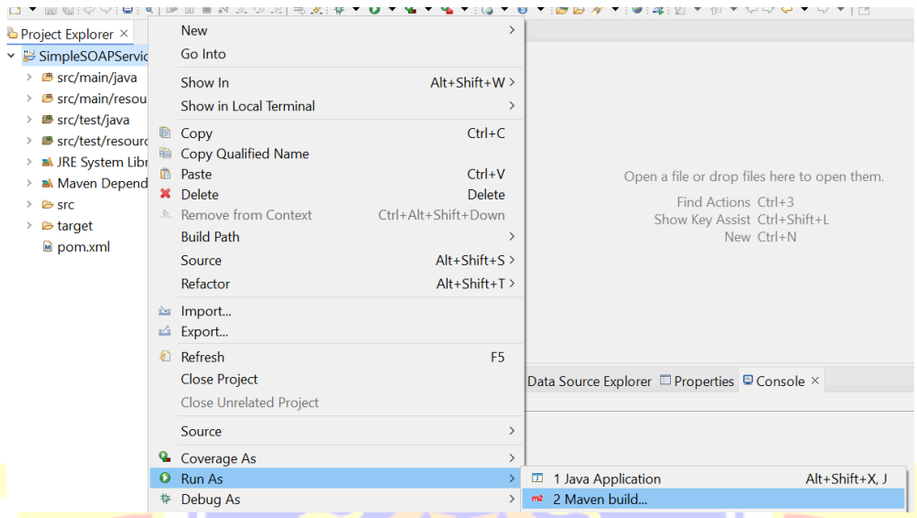
1. **Right-click** on com.example → **New** → **Class** and name it SOAPServer.java.
2. Add the following code to SOAPServer.java:

File Name: SOAPServer.java
Code:

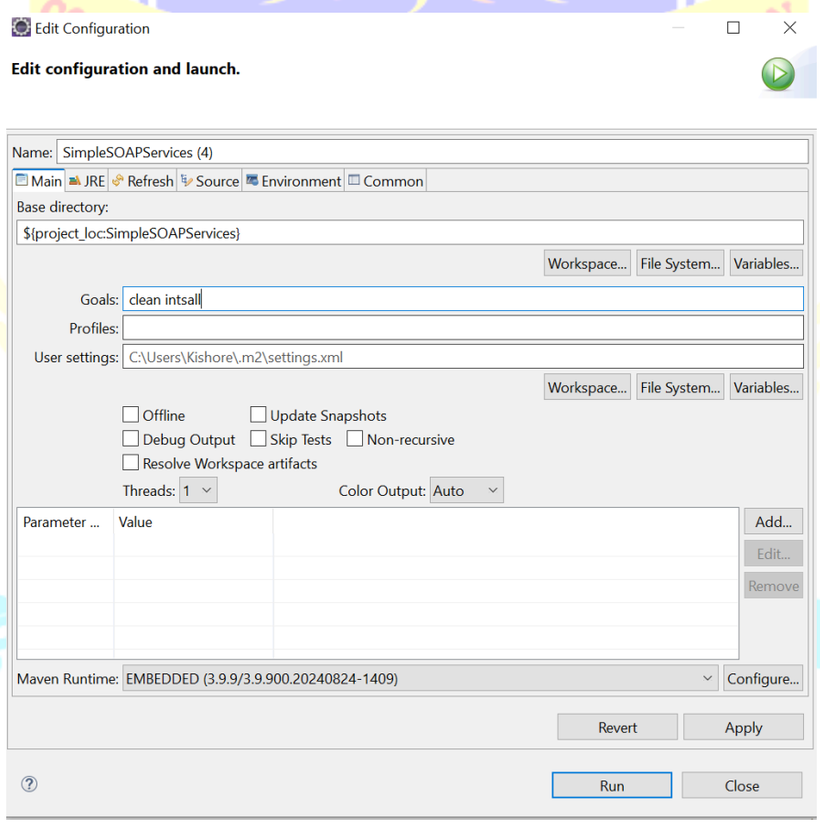
```
package com.example;

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);
    }
}
```

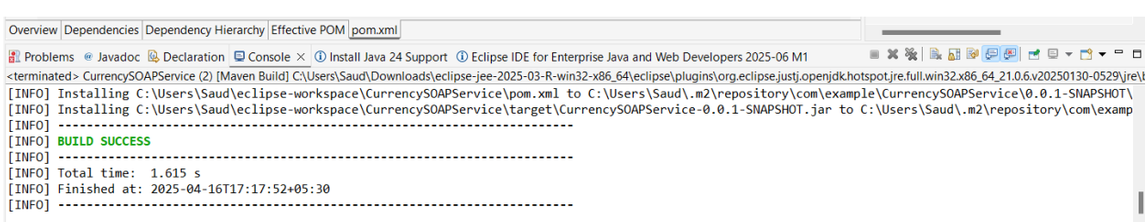
Save It

Output:
Right click on the project folder then go to run as then click on maven build
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

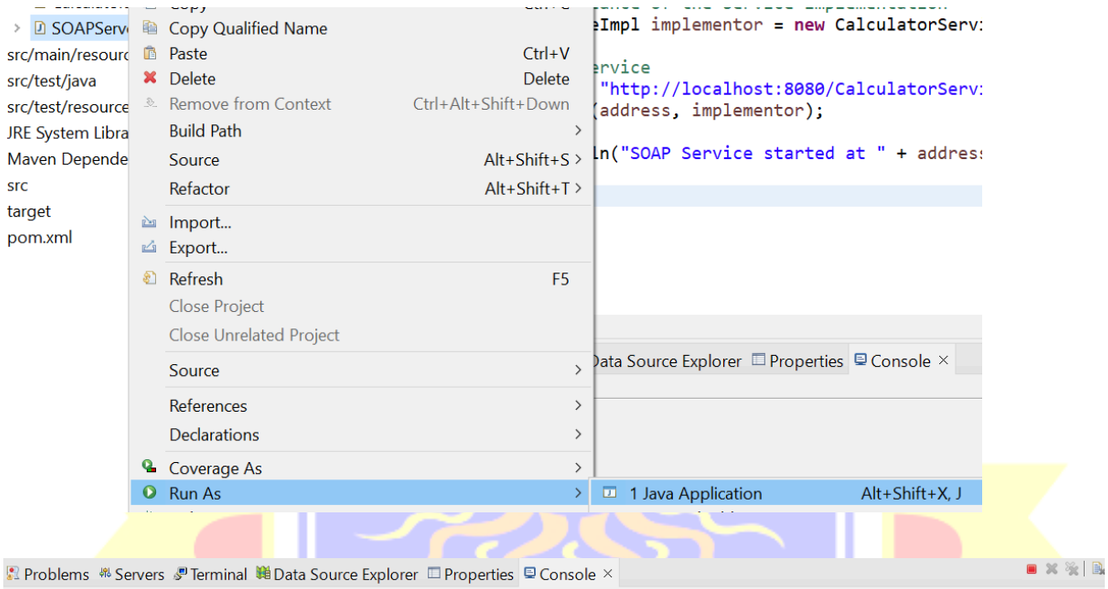


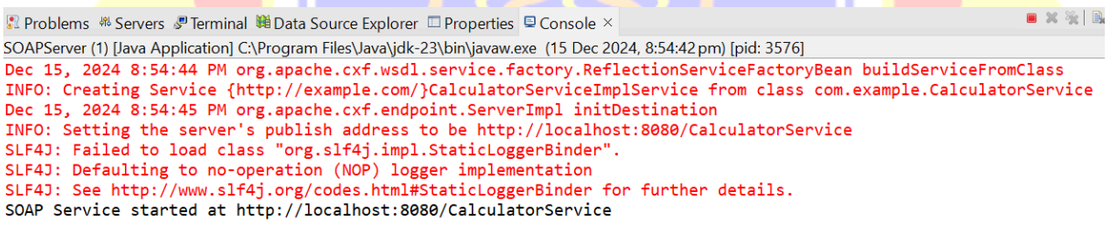


Run the Application

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

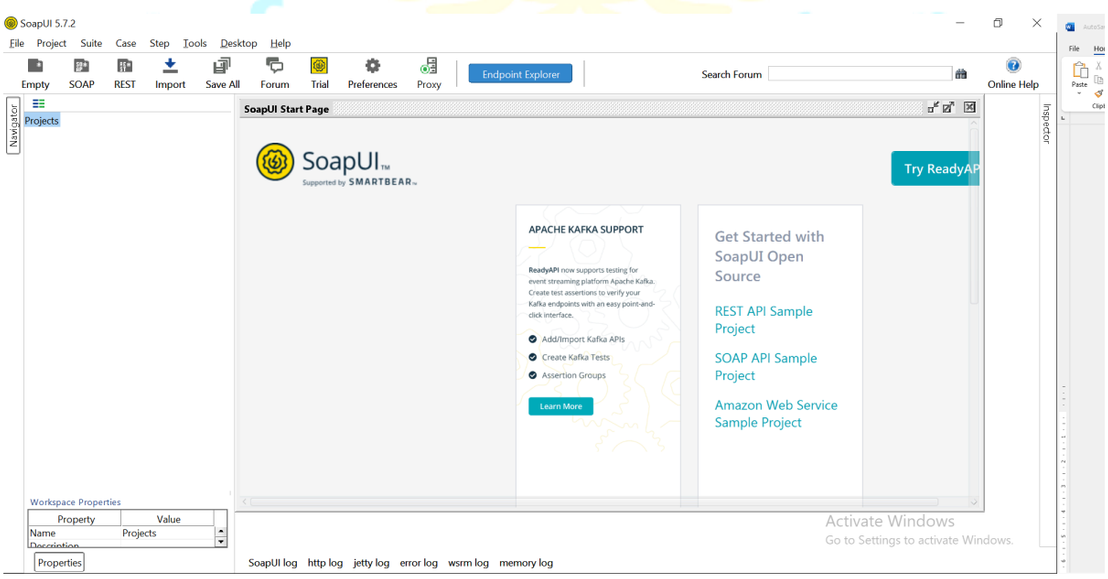


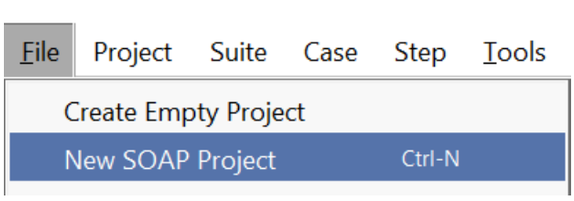


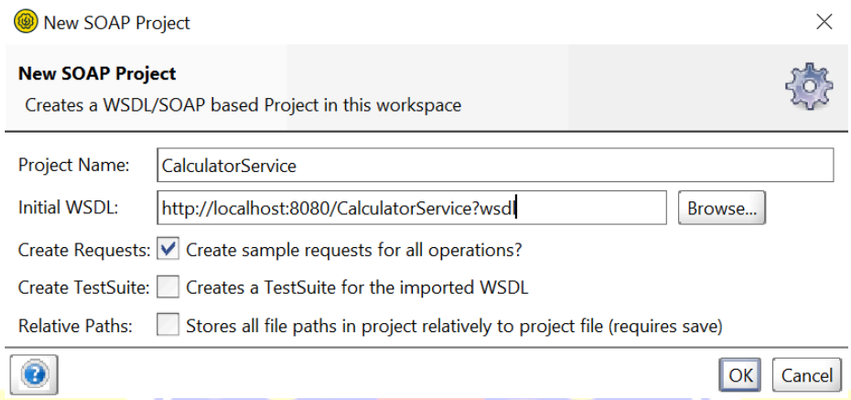
Step 7. Test the SOAP Service Using SOAPUI

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:

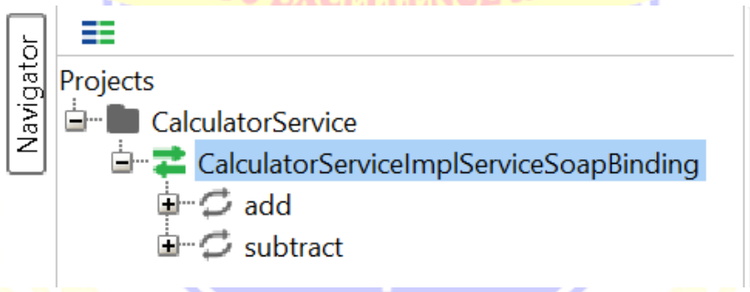


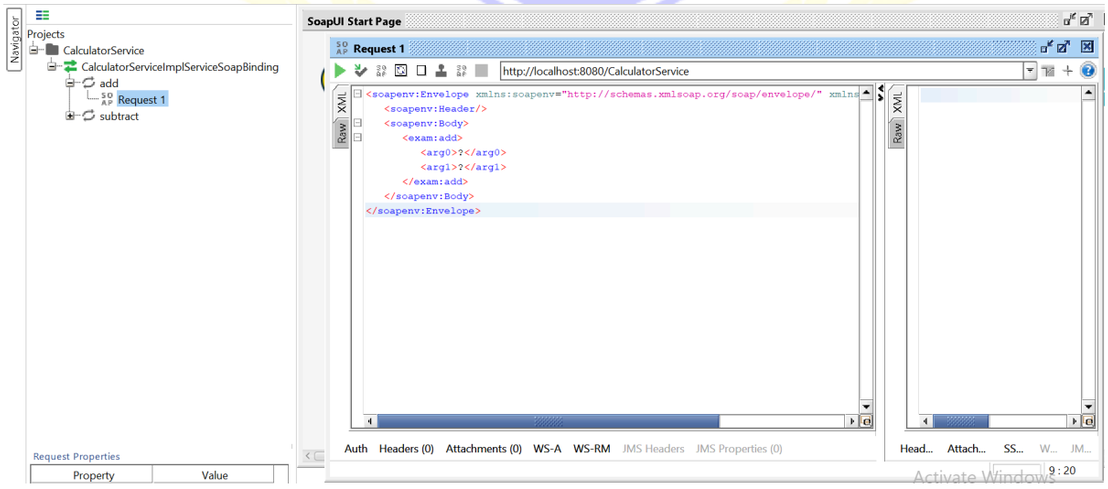




Click OK

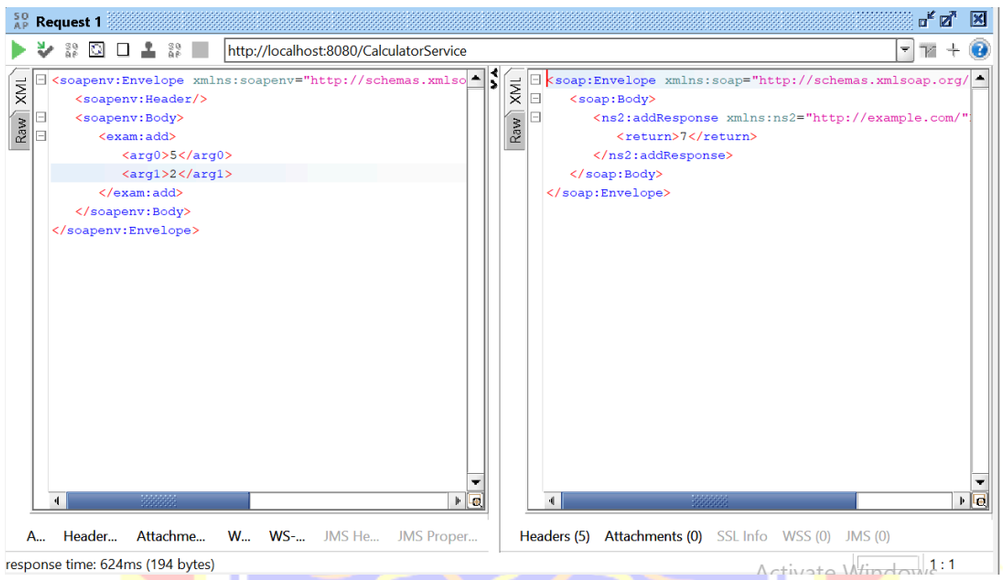
SOAPUI will import the WSDL and display the operations (add and subtract). You can now test the SOAP service by right-clicking on the operations and sending requests.





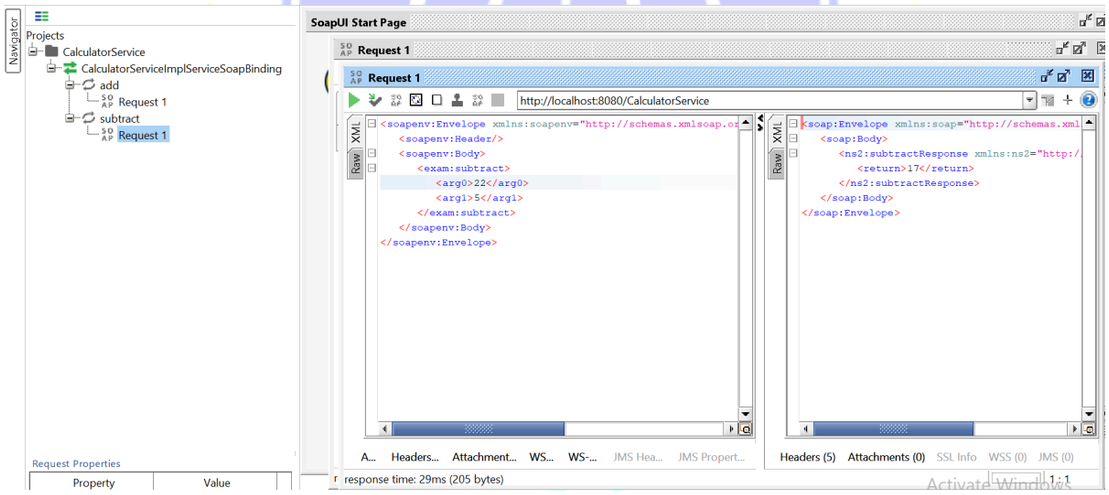
add > Request 1

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.



Result and Discussion: We have successfully implemented simple SOAP service

Learning Outcomes:
1. Successfully created a simple soap service.
2. Understood the SOAP, XML, and API.

Course Outcomes:
1. We have learned how to create a SOAP service.
2. We have learned how to consume a SOAP service.

Conclusion:
We have successfully implemented simple SOAP service

Viva Question:
1. What is SOAP?
2. What is XML?
3. What is API?
4. What is HTTP?

For Faculty Use

Correction Parameters	Formative Assessment [ ]	Timely completion practical [ ]	Attendance Learning Attitude[ ]

Theory-3
REST

REST (Representational State Transfer):
REST is an architectural style for web communication that utilizes URLs to identify resources and standard HTTP methods (GET, POST, PUT, DELETE) to manipulate them. Each request is stateless, meaning it contains all necessary information for processing.
REST is based on request response model

Web Service:
A web service is a software system designed to support interoperable machine-to-machine interaction over a network.
Essentially, it's a way for different applications to communicate with each other over the internet.
Web services can use various protocols for communication, such as SOAP (Simple Object Access Protocol) or REST.

API (Application Programming Interface):
An API is a set of rules and specifications that allow software applications to communicate and interact with each other.
It defines how different software components should interact.
APIs enable developers to access and use the functionality of other applications or services.
Web services utilize APIs. So any web service provides an API.

REST API (Representational State Transfer Application Programming Interface):
A REST API is a specific type of web API that adheres to the architectural principles of REST.
REST is an architectural style that emphasizes:
Statelessness: Each request from a client to a server must contain all the information needed to understand and process the request.
Client-server architecture: Client and server are seperate.
Use of standard HTTP methods: Such as GET, POST, PUT, and DELETE, to perform operations on resources.
Resource identification through URLs: Each resource is uniquely identified by a URL.
Use of representations: Data is transferred in various formats, such as JSON or XML.
REST APIs are widely used because they are lightweight, scalable, and easy to use.

In summary:
A web service is a way for applications to communicate over a network.
An API defines how applications interact.
REST is a stateless web communication style using URLs and HTTP methods for resource interaction.
A REST API is a specific type of web API that follows the REST architectural style.

Practical-3: Create a Simple REST Service.
Software Tools Required:

- **Code Editor**: Visual Studio Code (VS Code)
- **Framework**: Express.js (web framework)
- **Runtime Environment**: Node.js
- **Programming Languages**: JavaScript (for Node.js and Express.js), Java

Downloads Required:

- Node.js: Node.js — Download Node.js®
- JDK: Java Downloads | Oracle India (x64 MSI Installer)

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

Demonstration:

```
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

```
C:\Users\Saud>node -v
v22.14.0
```

Here all versions are shown so path is set by checking versions for Nodejs
Then run this command in powershell:

Set-ExecutionPolicy -Scope CurrentUser RemoteSigned

In Node.js:
To Implement the Service:

d. Create a folder `rest-service` and cd to rest-service
e. Run this in `rest-service` folder: npm i express
f. Create file: `server.js` and write the code in it

Source code: To create a simple REST API that takes an amount in Indian Rupees and returns the equivalent amount in US Dollars

Server side: server.js

```
const express = require('express')
const app = express()

app.get('/convert/:amount', (req , res) => {
    const inr = +req.params.amount;
    let usd = inr * 0.0116650131;
    usd = usd.toPrecision(4);
    res.json({ 'USD' : usd })
})

app.listen(3000, ()=>{
    console.log('server is running at port 3000')
})
```

Run the code: node server.js

Output:

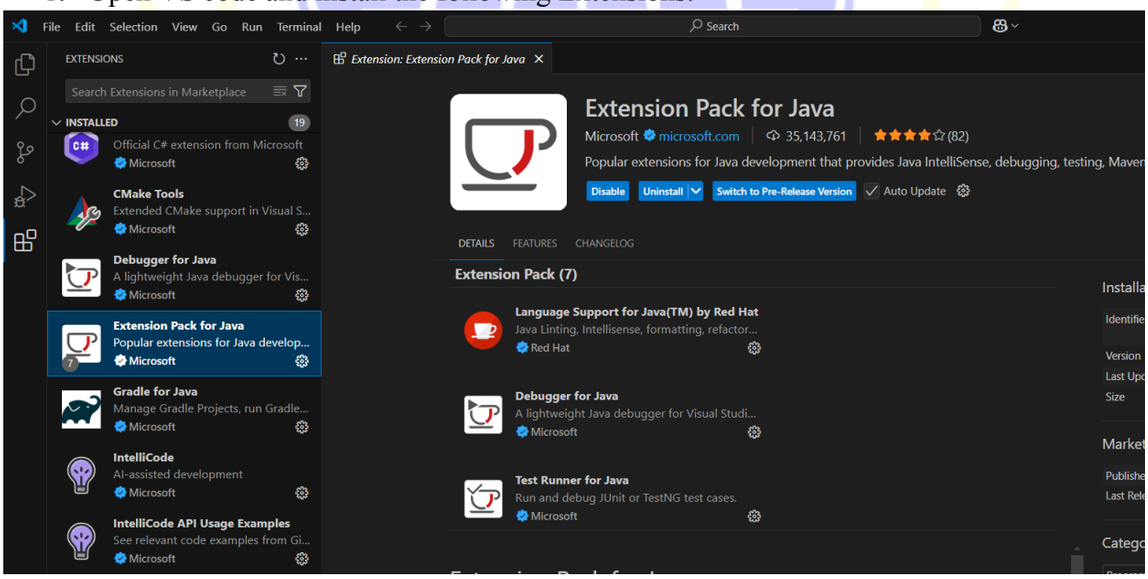
```
PS C:\Users\Saud\node_proj\rest-service> node server
server is running at port:  3000
```

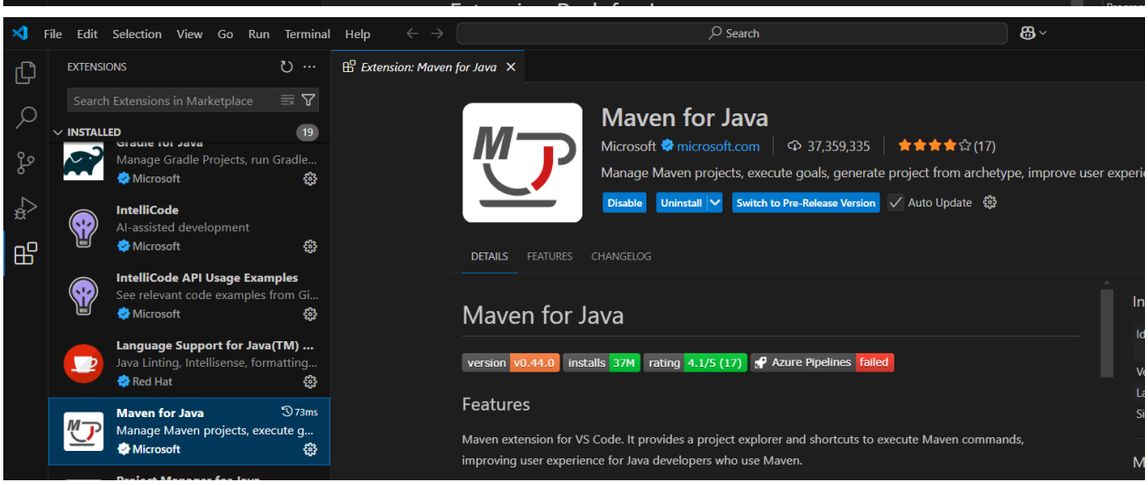
Test the RESTful API:

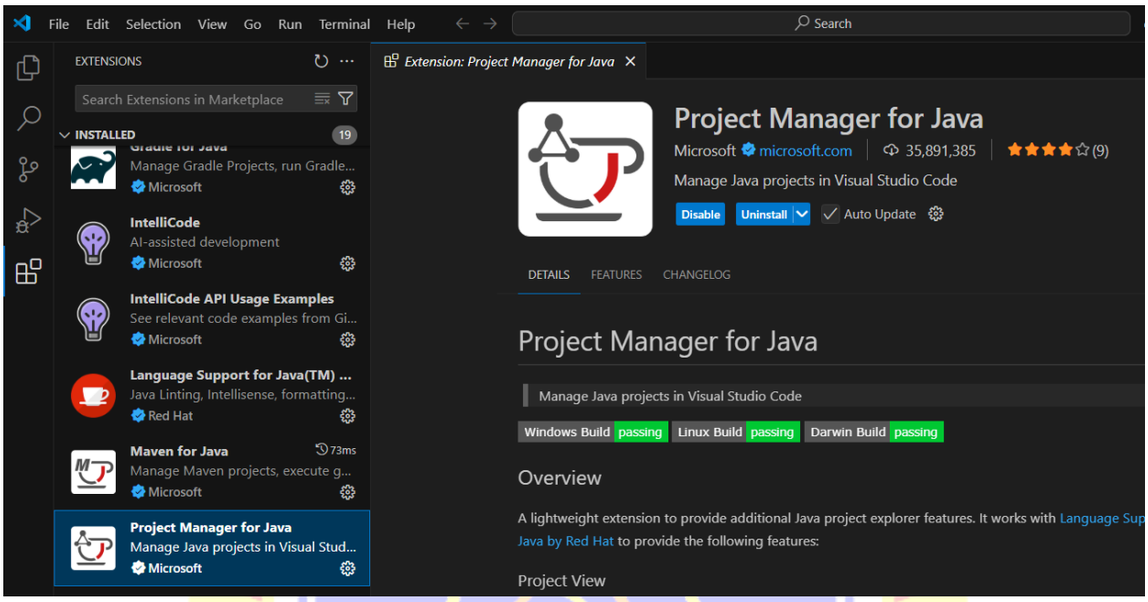
```
PS C:\Users\Saud\node_proj\rest-service> curl http://localhost:3000/convert/100


StatusCode        : 200
StatusDescription : OK
Content           : {"USD":"1.167"}
RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Keep-Alive: timeout=5
                    Content-Length: 15
                    Content-Type: application/json; charset=utf-8
                    Date: Wed, 16 Apr 2025 23:02:55 GMT
                    ETag: W/"f-gXfcn1ZB6zllY33qG2Dg9...
Forms             : {}
Headers           : {[Connection, keep-alive], [Keep-Alive, timeout=5], [Content-Length, 15], [Content-Type, application/json;
                    charset=utf-8]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 15
```

In Java:

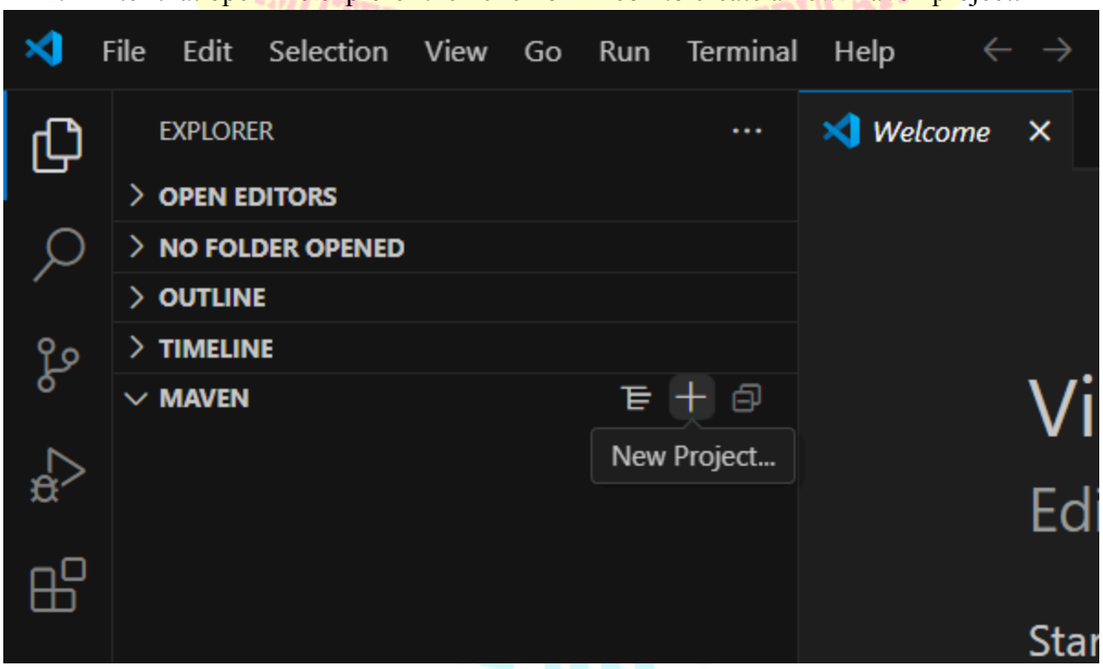
1. Open VS code and install the following Extensions:



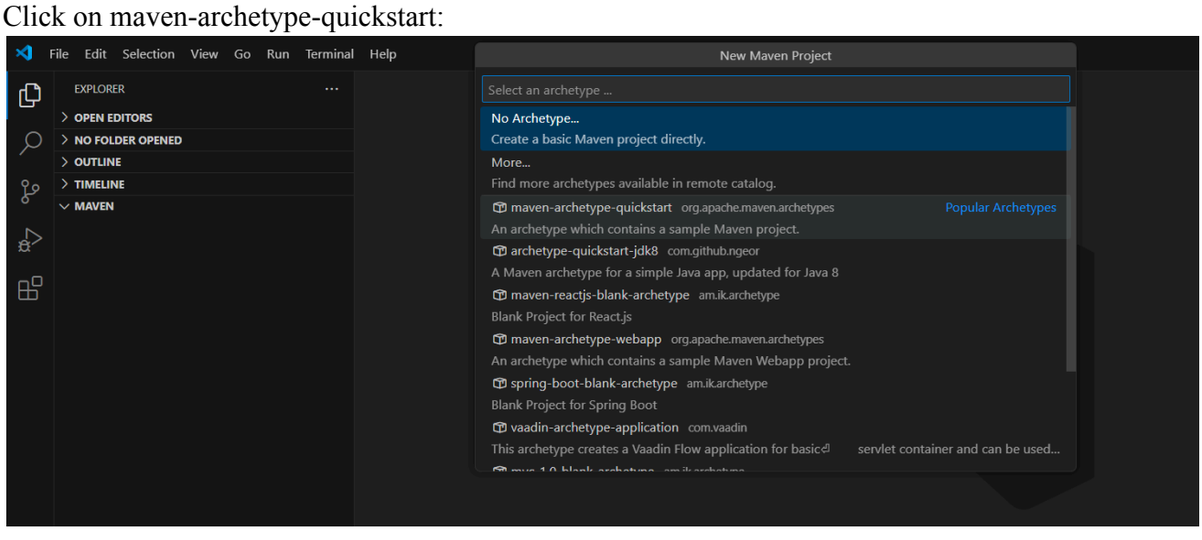




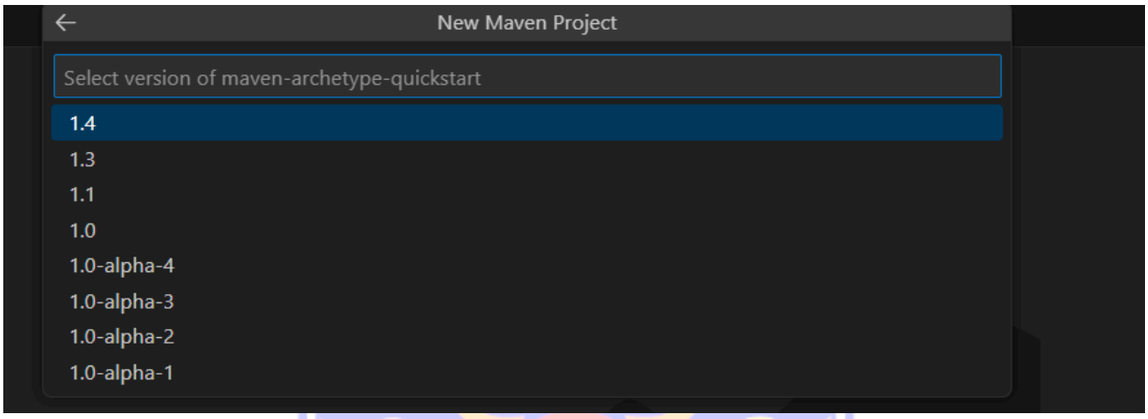
2. After that open file explorer then click on + icon to create a new maven project:



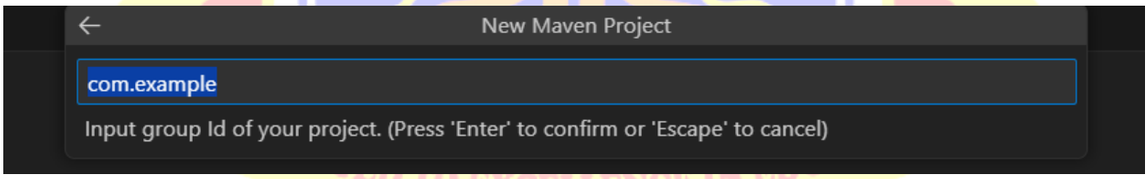
Click on maven-archetype-quickstart:



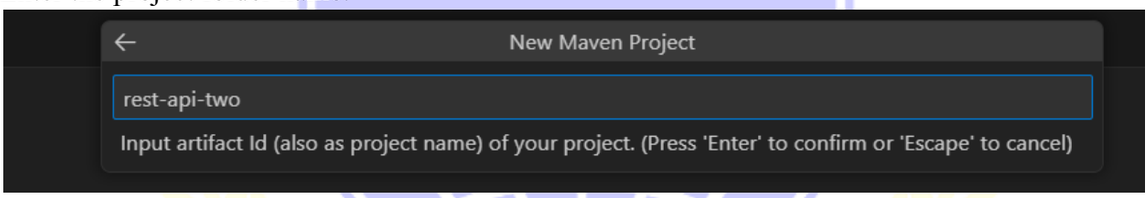
Click on 1.4:



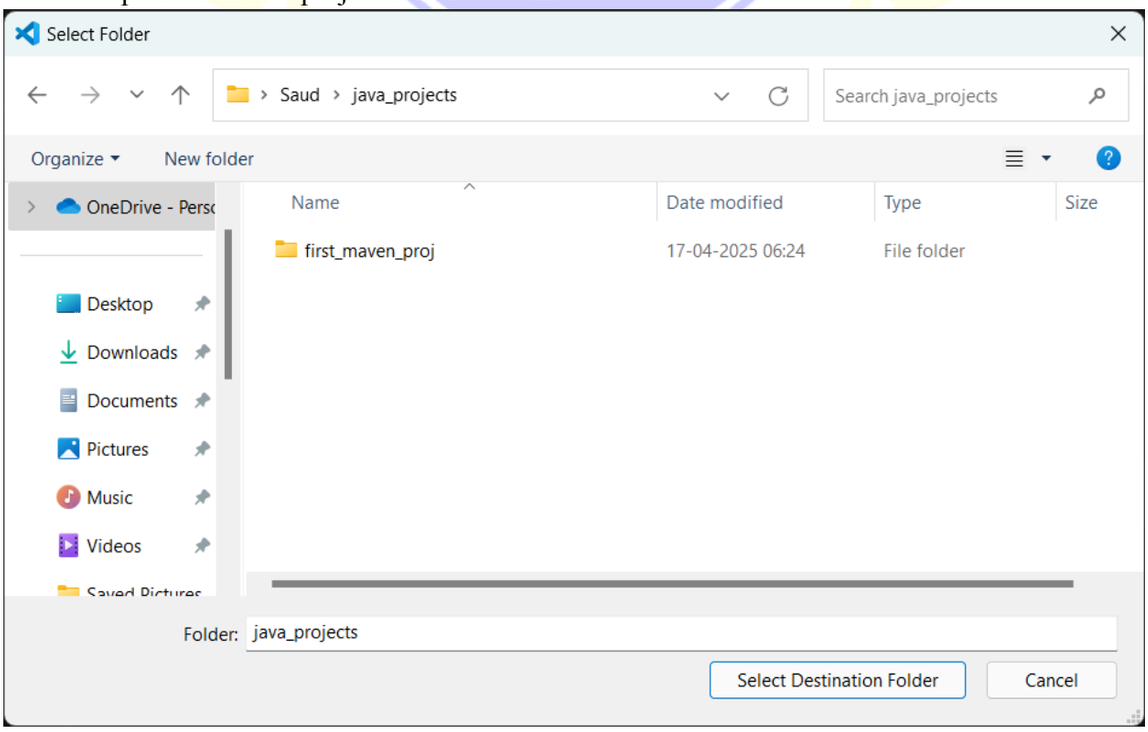
Continue:

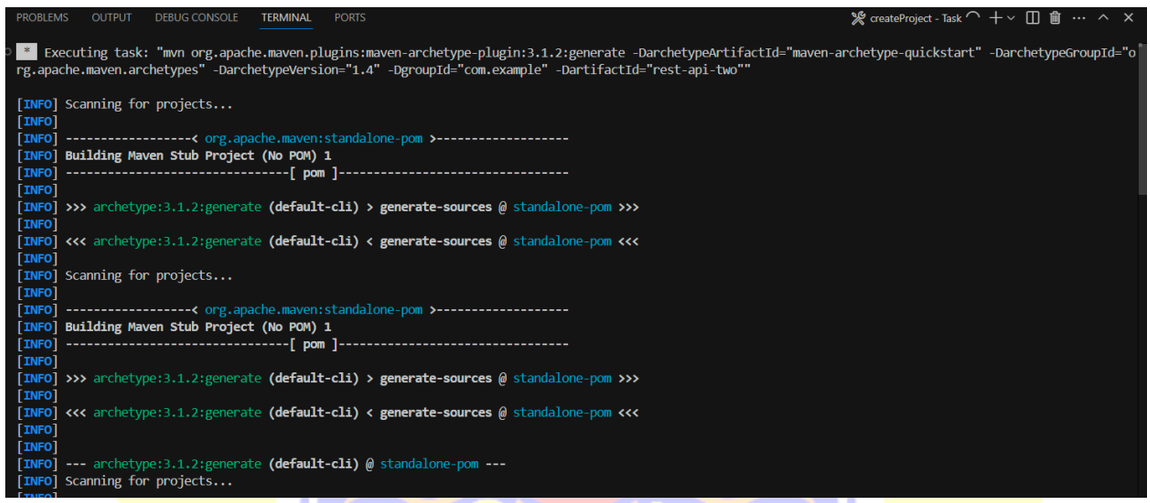


Enter the project folder name:

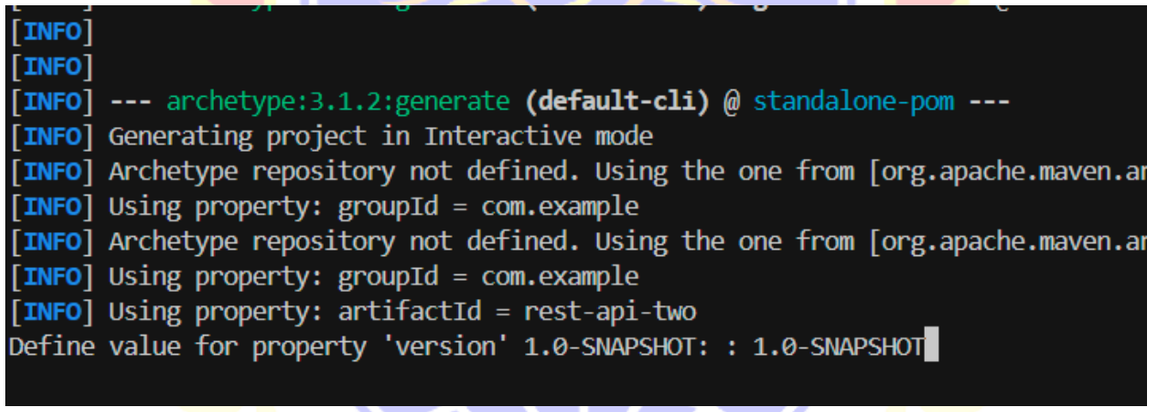


Select the path where the project folder to be made:

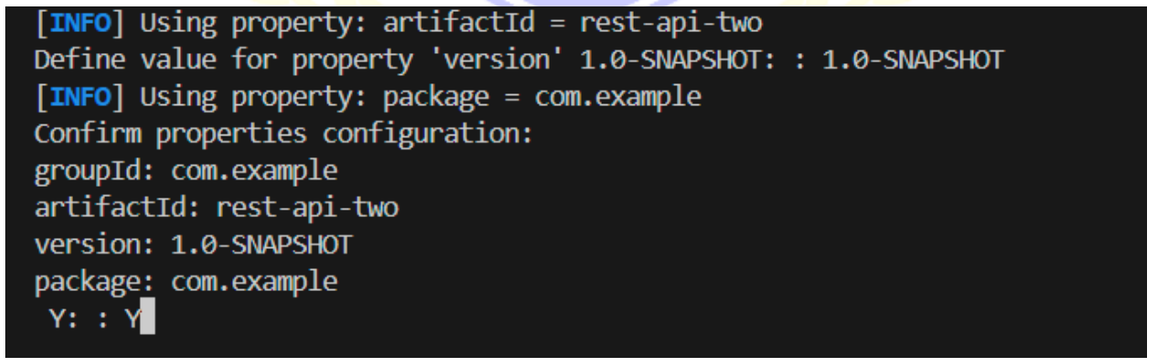


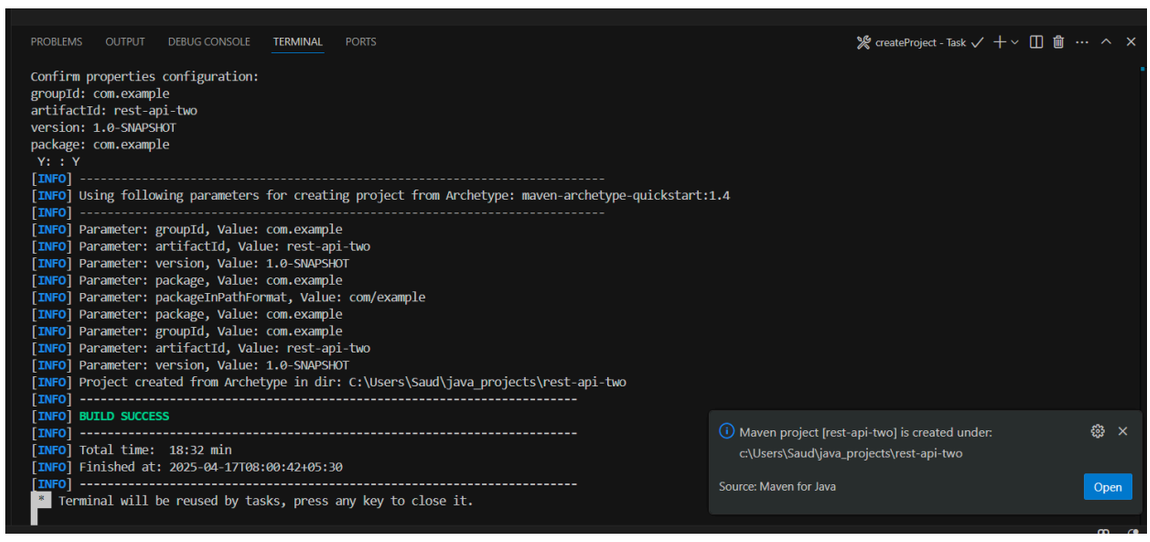


Enter: 1.0-SNAPSHOT

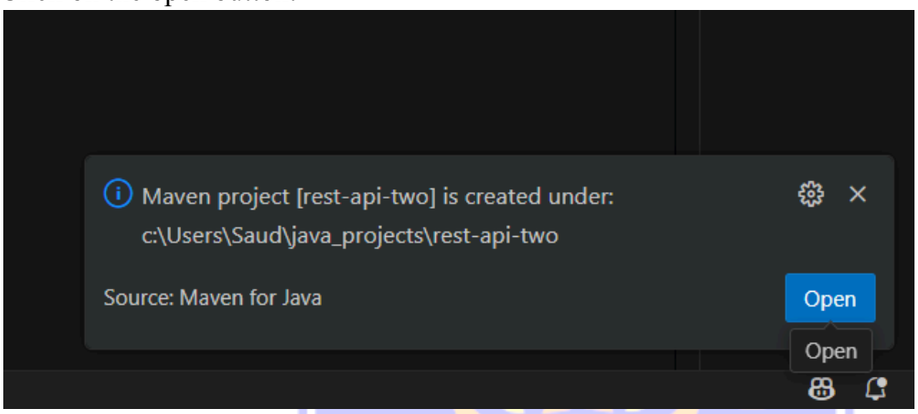


Enter: Y

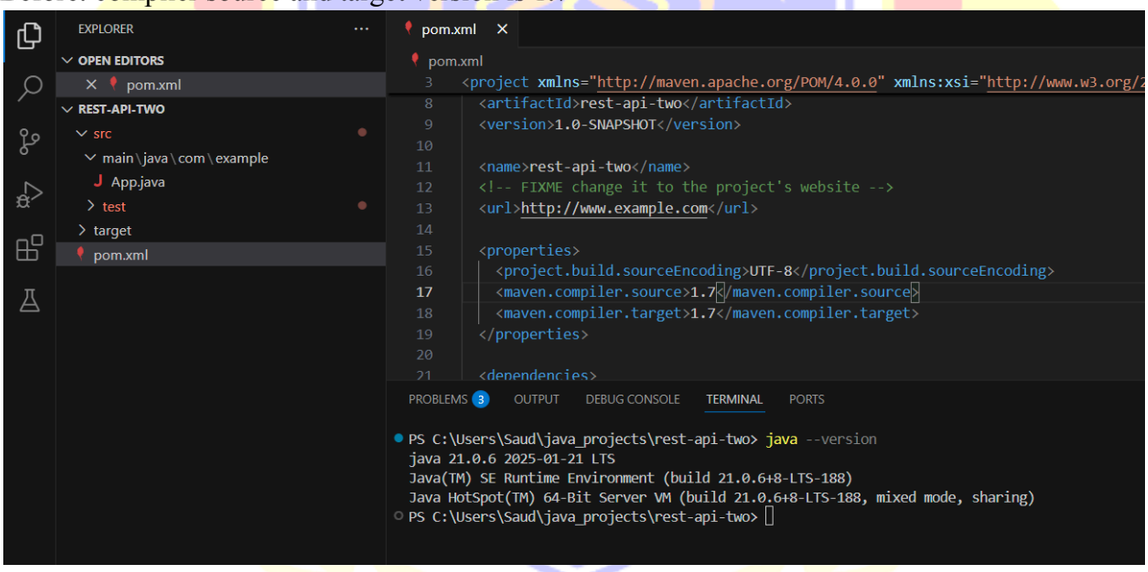




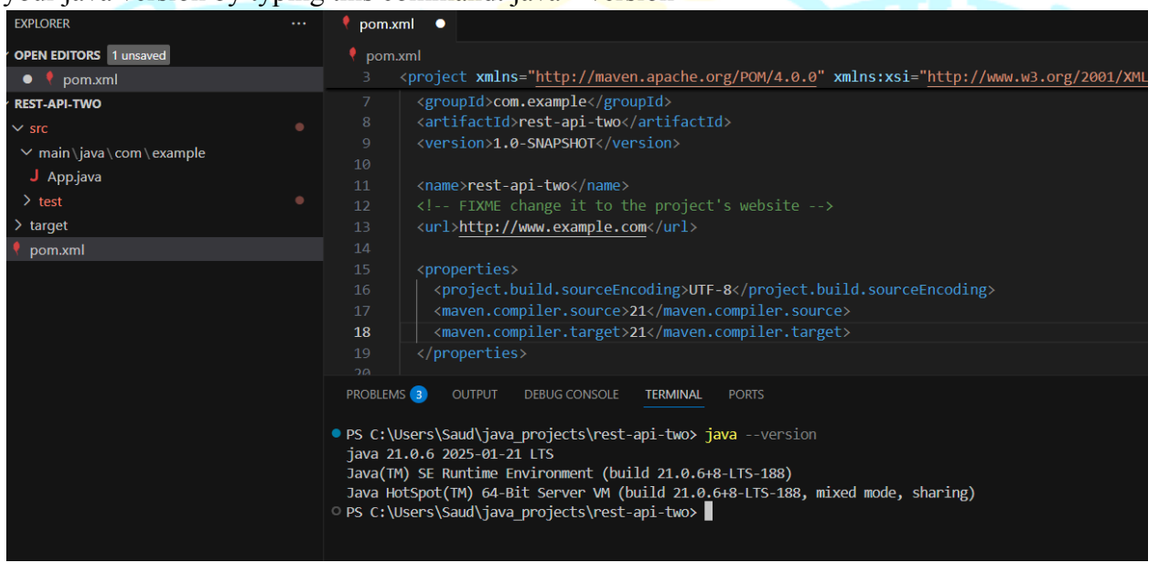
Click on the open button:



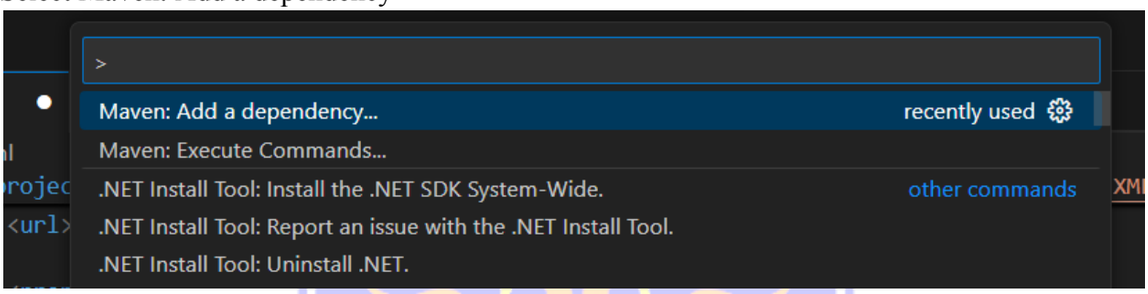
Edit pom.xml:
Before: compiler source and target version is 1.7



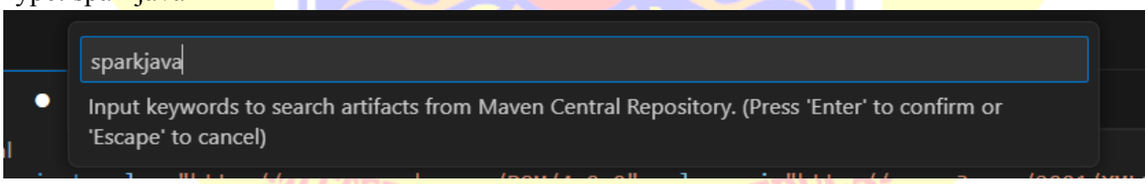
After: compiler source version is 21 because the java version installed on the system is 21, check your java version by typing this command: java --version



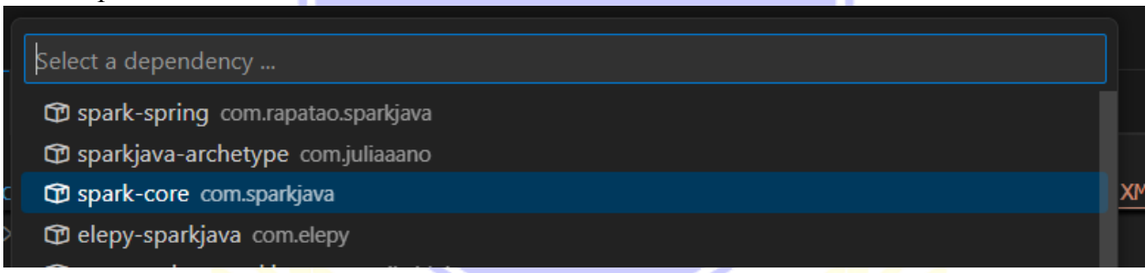
Type: Ctrl + Shift + P
It will open the command palette
Type: Maven: Add
Select Maven: Add a dependency



Type: sparkjava



Select: spark-core



It will add this in pom.xml file:

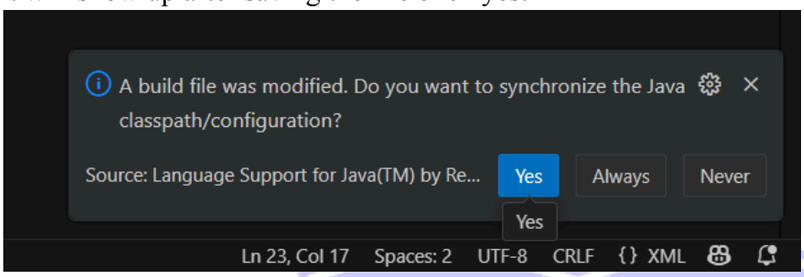
```
<dependencies>

  <dependency>
    <groupId>com.sparkjava</groupId>
    <artifactId>spark-core</artifactId>
    <version>2.9.4</version>
    <type>bundle</type>
  </dependency>
```

Remove type tag:

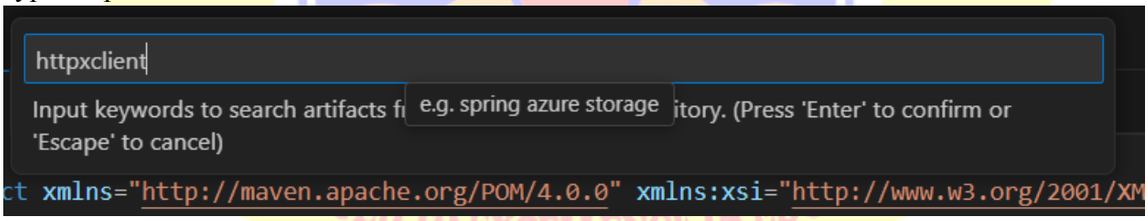
```
  <dependency>
    <groupId>com.sparkjava</groupId>
    <artifactId>spark-core</artifactId>
    <version>2.9.4</version>
  </dependency>
```

Save the file

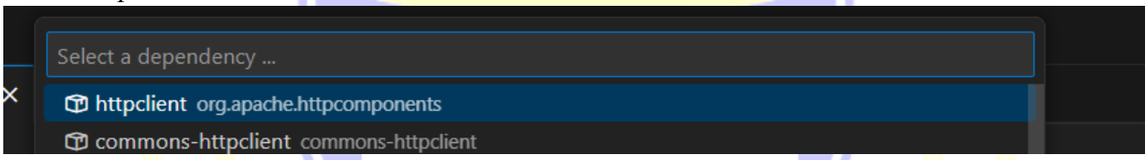
It will show up after saving the file click yes:



Again press: Ctrl + Shift + P
Select maven: add dependency
Type: httpclient



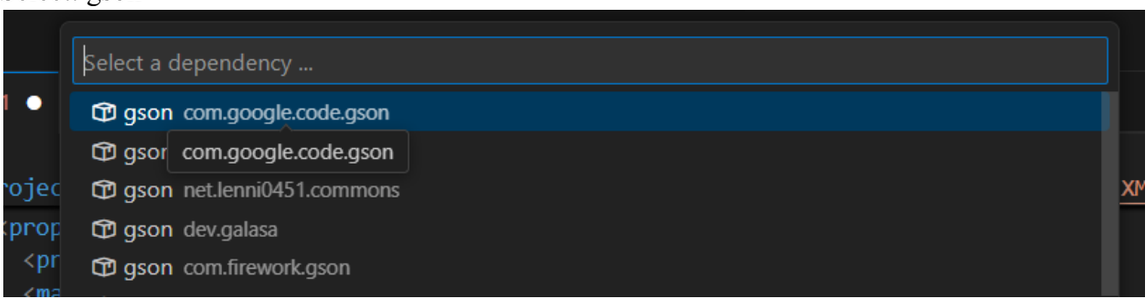
Select: httpclient



It will add this:

```
<dependencies>

  <dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.14</version>
  </dependency>
```

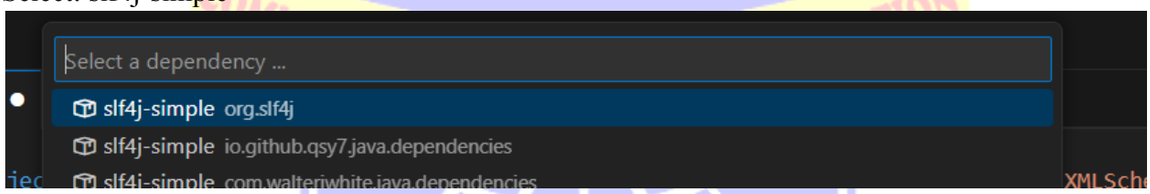
Again do same to open command palette: Ctrl+Shift + P
Type: gson
Select: gson



It will add this:

```
<dependencies>

  <dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.13.0</version>
  </dependency>
```

Again do same to open command palette: Ctrl+Shift + P
Type: slf4j-simple
Select: slf4j-simple



It will add this:

```
<dependencies>

  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>2.1.0-alpha1</version>
  </dependency>
```

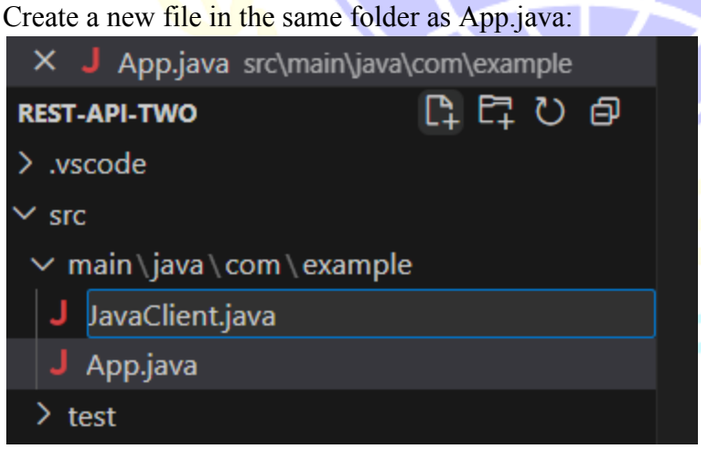
Modify App.js:

```
package com.example;

import static spark.Spark.*;
import java.util.Map;
import com.google.gson.Gson;

public class App {
    private static final Gson gson = new Gson();

    public static void main(String[] args) {
        port(8080);
        System.out.println("Starting currency conversion server...");

get("/convert/:inr", (req, res) -> {
```

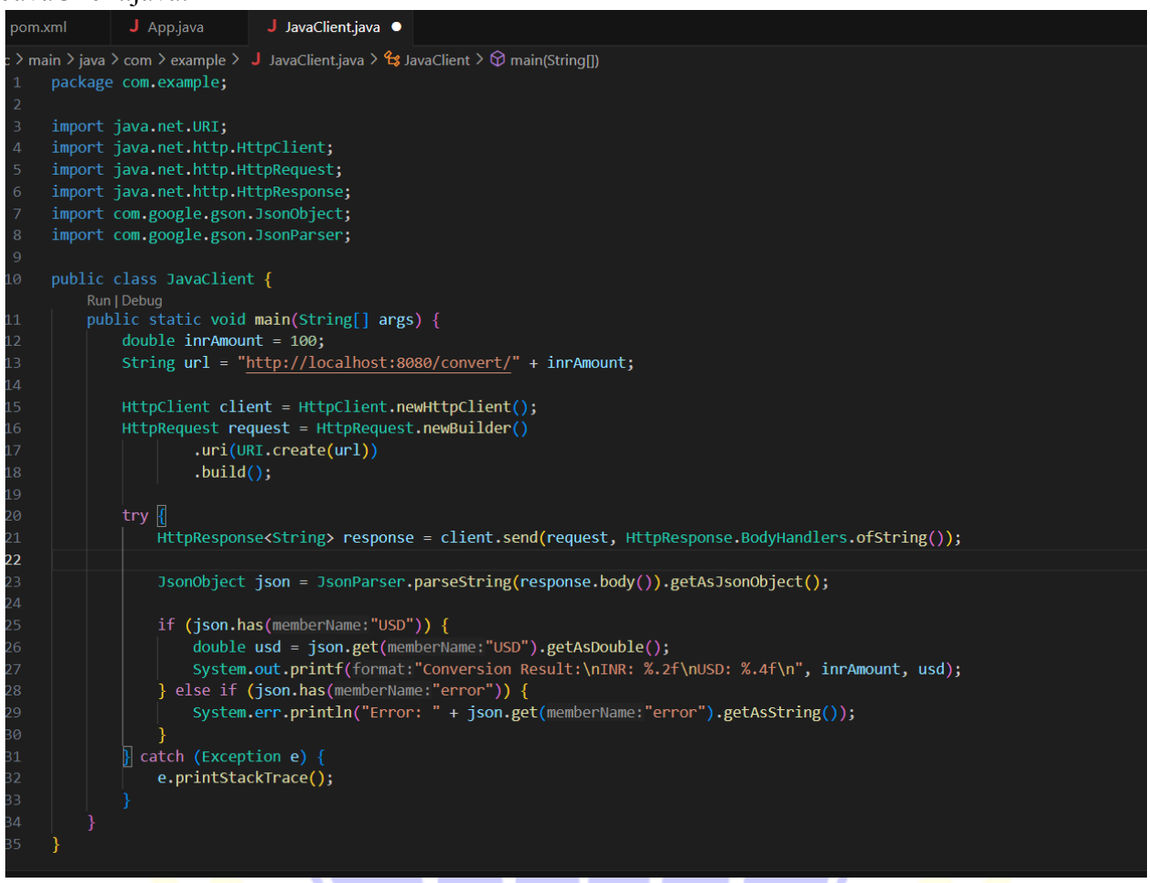
```
    try {
        double inr = Double.parseDouble(req.params("inr"));
        double usd = inr * 0.0116;
        res.type("application/json");
        return gson.toJson(Map.of("USD", usd));
    } catch (NumberFormatException e) {
        res.status(400);
        return gson.toJson(Map.of("error", "Invalid INR amount"));
    }
});


        Runtime.getRuntime().addShutdownHook(new Thread(() -> {
            System.out.println("Stopping server...");
            stop();
        }));

        System.out.println("Server ready: http://localhost:8080");
    }

}
```

Save it

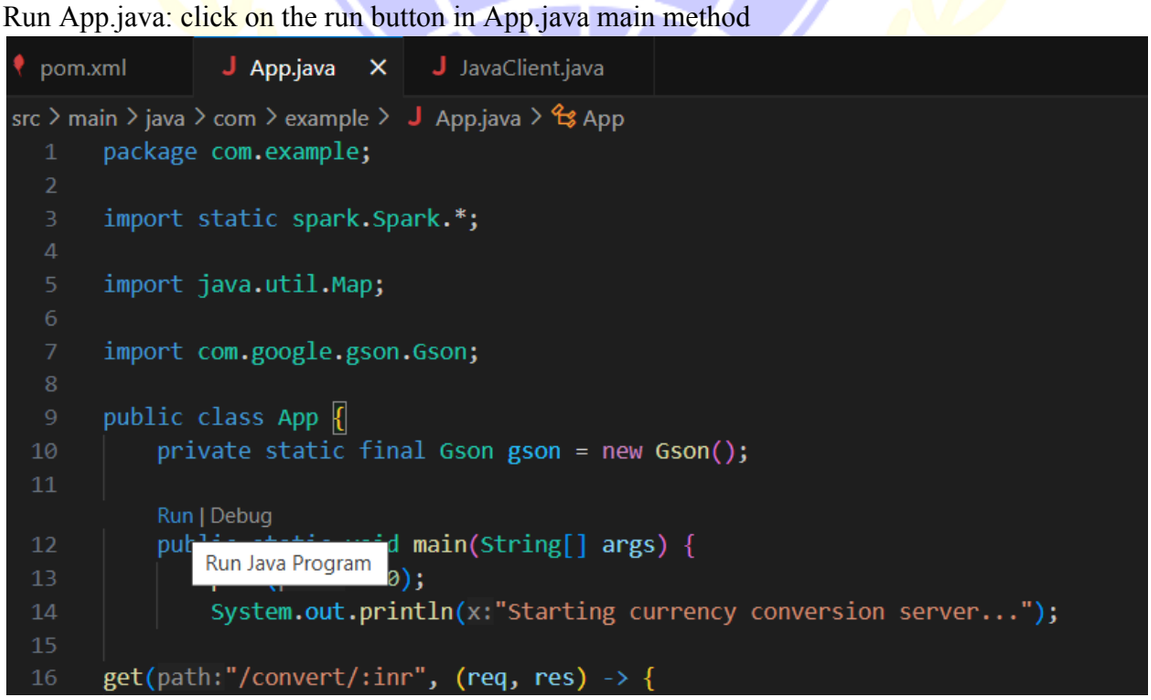
Create a new file in the same folder as App.java:



JavaClient.java:



Run App.java: click on the run button in App.java main method



```
ROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

S C:\Users\Saud\java_projects\rest-api-two>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '@C:\Users\Saud\AppData\Local\Temp\cp_r132eb84wfkwe
k7oqrwfy2n.argfile' 'com.example.App'
LF4J(I): Connected with provider of type [org.slf4j.simple.SimpleServiceProvider]
tarting currency conversion server...
erver ready: http://localhost:8080
Thread-0] INFO org.eclipse.jetty.util.log - Logging initialized @134ms to org.eclipse.jetty.util.log.Slf4jLog
Thread-0] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - == Spark has ignited ...
Thread-0] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - >> Listening on 0.0.0.0:8080
Thread-0] INFO org.eclipse.jetty.server.Server - jetty-9.4.48.v20220622; built: 2022-06-21T20:42:25.880Z; git: 6b67c5719d1f4371b33655ff2d047d2
e171e49a; jvm 21.0.6+8-LTS-188
Thread-0] INFO org.eclipse.jetty.server.session - DefaultSessionIdManager workerName=node0
Thread-0] INFO org.eclipse.jetty.server.session - No SessionScavenger set, using defaults
Thread-0] INFO org.eclipse.jetty.server.session - node0 Scavenging every 600000ms
Thread-0] INFO org.eclipse.jetty.server.AbstractConnector - Started ServerConnector@62361ccb{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
Thread-0] INFO org.eclipse.jetty.server.Server - Started @288ms
```

The server is started

Test the REST API:

```
PS C:\Users\Saud\java_projects\rest-api-two> curl http://localhost:8080/convert/100


StatusCode        : 200
StatusDescription : OK
Content           : {"USD":1.16}
RawContent        : HTTP/1.1 200 OK
                    Transfer-Encoding: chunked
                    Content-Type: application/json
                    Date: Thu, 17 Apr 2025 03:31:52 GMT
                    Server: Jetty(9.4.48.v20220622)

                    {"USD":1.16}
Forms             : {}
Headers           : {[Transfer-Encoding, chunked], [Content-Type, application/json], [Date, Thu, 17 Apr 2025 03:31:52 GMT], [Server,
                    Jetty(9.4.48.v20220622)]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 12


PS C:\Users\Saud\java_projects\rest-api-two>
```

Run the JavaClient:

pom.xml | App.java | JavaClient.java

src > main > java > com > example > JavaClient.java > JavaClient > main(String[])

```
1   package com.example;
2
3   import java.net.URI;
4   import java.net.http.HttpClient;
5   import java.net.http.HttpRequest;
6   import java.net.http.HttpResponse;
7   import com.google.gson.JsonObject;
8   import com.google.gson.JsonParser;
9
10  public class JavaClient {
        Run | Debug
11      pu  Run Java Program  d main(String[] args) {
12          double inrAmount = 100;
13          String url = "http://localhost:8080/convert/" + inrAmount;
14
```

Output:

```
PS C:\Users\Saud\java_projects\rest-api-two>  & 'C:\Program Fil
qk7oqrwfy2n.argfile' 'com.example.JavaClient'
Conversion Result:
INR: 100.00
USD: 1.1600
PS C:\Users\Saud\java_projects\rest-api-two>
```

Result and Discussion:
We have successfully performed REST API

Learning Outcomes:
1. Successfully implemented REST API
2. Understood REST API

Course Outcomes:
1. We have learned about REST API
2. We have learned about HTTP Methods

Conclusion:
We have successfully implemented RESTFul API

Viva Question:
1. What is REST?
2. What is RESTFul API?
3.What is GET HTTP Method used for?
4. What is POST HTTP Method used for?

For Faculty Use

Correction Parameters	Formative Assessment [ ]	Timely completion practical [ ]	Attendance Learning Attitude[ ]

Theory-4
REST

REST (Representational State Transfer):
REST is an architectural style for web communication that utilizes URLs to identify resources and standard HTTP methods (GET, POST, PUT, DELETE) to manipulate them. Each request is stateless, meaning it contains all necessary information for processing.
REST is based on request response model

Web Service:
A web service is a software system designed to support interoperable machine-to-machine interaction over a network.
Essentially, it's a way for different applications to communicate with each other over the internet.
Web services can use various protocols for communication, such as SOAP (Simple Object Access Protocol) or REST.

API (Application Programming Interface):
An API is a set of rules and specifications that allow software applications to communicate and interact with each other.
It defines how different software components should interact.
APIs enable developers to access and use the functionality of other applications or services.
Web services utilize APIs. So any web service provides an API.

REST API (Representational State Transfer Application Programming Interface):
A REST API is a specific type of web API that adheres to the architectural principles of REST.
REST is an architectural style that emphasizes:
Statelessness: Each request from a client to a server must contain all the information needed to understand and process the request.
Client-server architecture: Client and server are seperate.
Use of standard HTTP methods: Such as GET, POST, PUT, and DELETE, to perform operations on resources.
Resource identification through URLs: Each resource is uniquely identified by a URL.
Use of representations: Data is transferred in various formats, such as JSON or XML.
REST APIs are widely used because they are lightweight, scalable, and easy to use.
A REST API, also known as a RESTful API, is a simple, uniform interface that is used to make data, content, algorithms, media, and other digital resources available through web URLs. REST APIs are the most common APIs used across the web today.

❖ **To Get Google Custom Search API Key and Search Engine ID:**
➢ Go to the Google Cloud Console: console.cloud.google.com
➢ Create a project or select an existing one.
➢ Enable the Custom Search API.
➢ Create API credentials (API key).
➢ Go to https://cse.google.com/cse/ and create a custom search engine.
➢ Grab the custom search engine ID (cx).

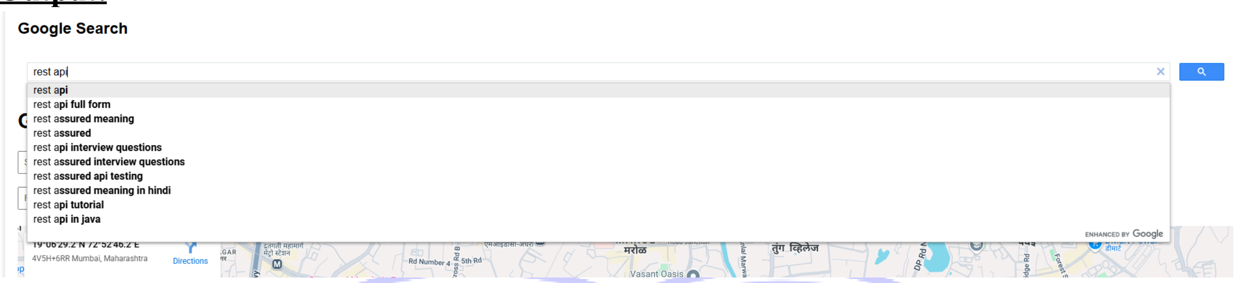
Practical-4: Develop application to consume Google's search / Google's Map RESTful Web service.

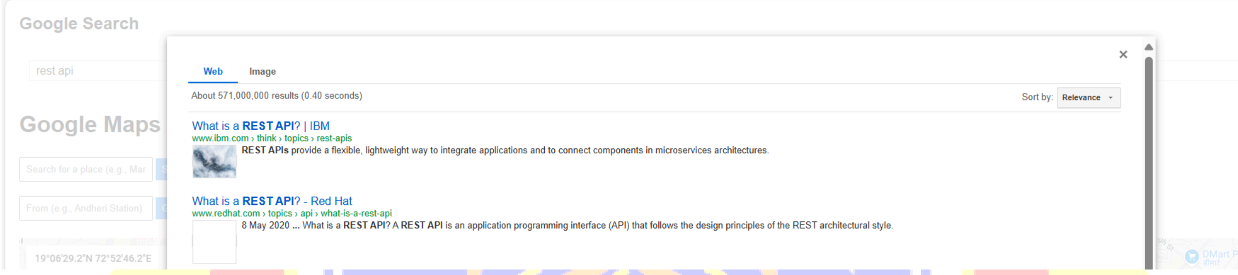
Source code:

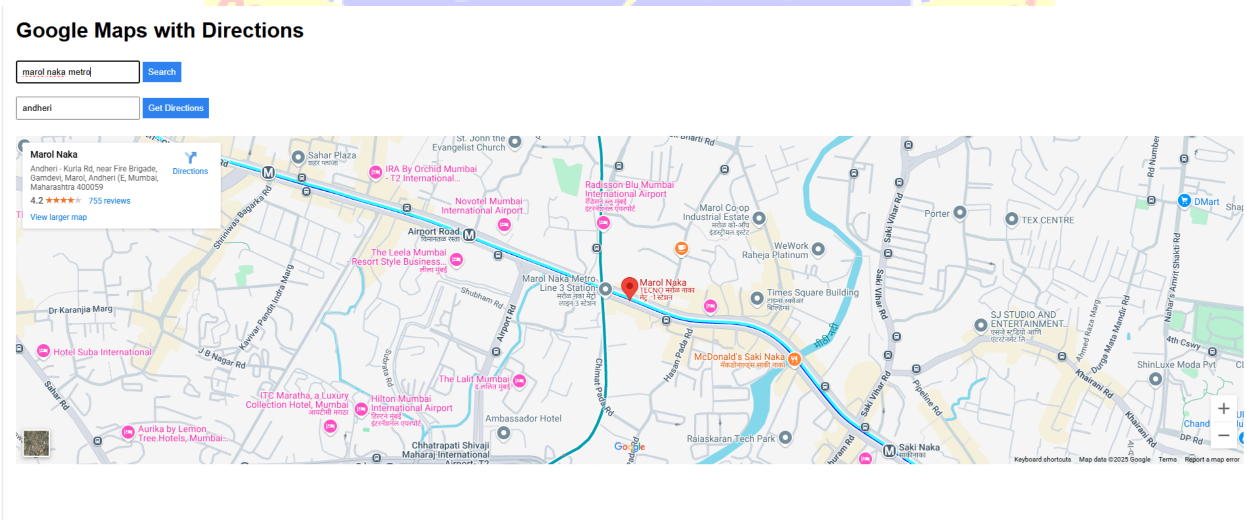
```
<!DOCTYPE html>
<html>
<head>
  <title>Google Maps with Direction</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 20px; }
    #map-container { margin: 20px 0; }
    input, button { padding: 8px; margin: 5px 0; }
    button { background: #4285F4; color: white; border: none; cursor:
pointer; }
    button:hover { background: #3367D6; }
    .directions-panel { margin-top: 10px; }
  </style>
</head>
<body>
  <h2>Google Search</h2>
  <script async
src="https://cse.google.com/cse.js?cx=b24005a08cd97459f"></script>
  <div class="gcse-search"></div>
  <h1>Google Maps with Directions</h1>

  <!-- Search Box -->
  <input type="text" id="search-box" placeholder="Search for a place (e.g.,
Marol Naka)" />
  <button onclick="searchPlace()">Search</button>

  <!-- Directions Panel -->
  <div class="directions-panel">
    <input type="text" id="origin" placeholder="From (e.g., Andheri
Station)" />
    <button onclick="showDirections()">Get Directions</button>
  </div>

  <!-- Embedded Google Maps (No API Key) -->
  <div id="map-container">
    <iframe
      id="embedded-map"
      width="100%"
      height="500"
      frameborder="0"
      scrolling="no"
      marginheight="0"
      marginwidth="0"

src="https://maps.google.com/maps?q=19.1081,72.8795&z=15&output=embed">
    </iframe>
```

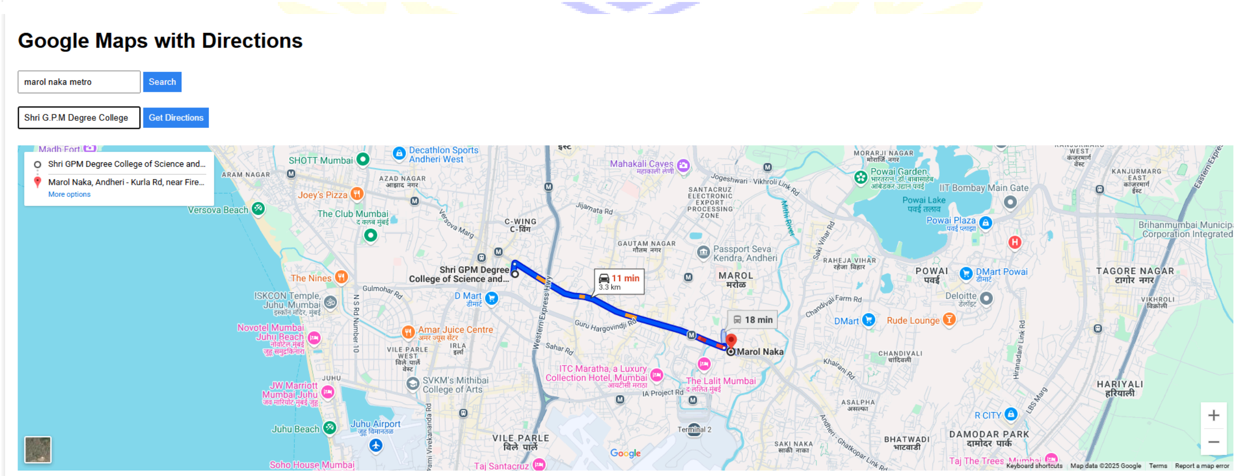
```
  </div>

  <script>
    // Search for a place and update the map
    function searchPlace() {
      const place = document.getElementById("search-box").value.trim();
      if (!place) return;

      const iframe = document.getElementById("embedded-map");
      iframe.src =
`https://maps.google.com/maps?q=${encodeURIComponent(place)}&output=embed`;
    }

    // Show directions between two points
    function showDirections() {
      const origin = document.getElementById("origin").value.trim();
      const destination =
document.getElementById("search-box").value.trim();

      if (!origin || !destination) {
        alert("Please enter both 'From' and 'To' locations!");
        return;
      }

      const iframe = document.getElementById("embedded-map");
      iframe.src =
`https://maps.google.com/maps?q=&saddr=${encodeURIComponent(origin)}&daddr=$
{encodeURIComponent(destination)}&output=embed`;
    }

    // Press "Enter" to trigger search
    document.getElementById("search-box").addEventListener("keypress", (e)
=> {
      if (e.key === "Enter") searchPlace();
    });
    document.getElementById("origin").addEventListener("keypress", (e) => {
      if (e.key === "Enter") showDirections();
    });
  </script>
</body>
</html>
```

Output:


Google Search
rest api
rest api
rest api full form
rest assured meaning
rest assured
rest api interview questions
rest assured interview questions
rest assured api testing
rest assured meaning in hindi
rest api tutorial
rest api in java
ENHANCED BY Google


Google Search
Google Maps
Web
Image
About 571,000,000 results (0.40 seconds)
Sort by: Relevance
What is a REST API? | IBM
www.ibm.com › think › topics › rest-apis
REST APIs provide a flexible, lightweight way to integrate applications and to connect components in microservices architectures.
What is a REST API? - Red Hat
www.redhat.com › topics › api › what-is-a-rest-api
8 May 2020 ... What is a REST API? A REST API is an application programming interface (API) that follows the design principles of the REST architectural style.


Google Maps with Directions
marol naka metro
Search
andheri
Get Directions
Marol Naka
Andheri - Kurla Rd, near Fire Brigade, Garndevi, Marol, Andheri (E, Mumbai, Maharashtra 400059
4.2 755 reviews
Directions
View larger map


Google Maps with Directions
marol naka metro
Search
Shri G.P.M Degree College
Get Directions
Shri GPM Degree College of Science and...
Marol Naka, Andheri - Kurla Rd, near Fire...
More options
11 min
3.3 km
18 min

Result and Discussion: We have successfully implemented an app that consumes google maps and search functionality.

Learning Outcomes:
1. Successfully implemented an app that consumes google maps and search functionality.
2. Understood how to integrate google search and maps in an application.

Course Outcomes:
1. We have learned how to integrate google search in an app
2. We have learned how to integrate google map in an app

Conclusion:
We have successfully implemented an app that consumes google maps and search functionality.

Viva Question:
1. What is API?
2. What is REST?
3. What is web service?
4. What is RESTful API?

For Faculty Use

Correction Parameters	Formative Assessment [ ]	Timely completion practical [ ]	Attendance Learning Attitude[ ]

Theory-5
Virtualization & KVM

Virtualization:

- In simple terms, virtualization is the process of creating a virtual version of something, like a computer, operating system, or network.
- It allows you to run multiple operating systems or applications on a single physical machine.
- This increases efficiency, reduces hardware costs, and provides greater flexibility.

VM (Virtual Machine):

- A VM is essentially a software-based emulation of a physical computer.
- It allows you to run an operating system and applications within a simulated hardware environment.
- Think of it as a computer within a computer.
- VMs are used for various purposes, including:
    - Running different operating systems on the same hardware.
    - Testing software in isolated environments.
    - Consolidating server resources.

Kernel:

- The kernel is the core of an operating system.
- It's the software that manages the computer's hardware resources, such as the CPU, memory, and storage devices.
- It acts as an intermediary between the hardware and the software applications.
- Key functions of the kernel include:
    - Process management: Controlling how applications run.
    - Memory management: Allocating and managing memory.
    - Device drivers: Enabling communication with hardware devices.
    - File system management: Organizing and accessing files.
- Essentially, the kernel is what makes the hardware usable by the software.

KVM (Kernel-based Virtual Machine):

- KVM is a specific virtualization technology that's built into the Linux kernel.
- It allows you to turn a Linux machine into a hypervisor. A hypervisor is software that creates and runs virtual machines.
- Key points about KVM:
    - It's open-source.
    - It leverages the Linux kernel's capabilities.
    - It enables you to run multiple virtual machines (VMs) on a Linux host.
    - Each VM runs as a separate Linux process.
    - It provides near native performance.

Essentially, KVM lets you run different operating systems (like Windows or other Linux distributions) on a Linux machine, as if they were separate physical computers.

Hypervisor:

A hypervisor is essentially the software that makes virtualization possible.

- **Core Function:**
    - A hypervisor allows you to run multiple virtual machines (VMs) on a single physical computer (host).
    - It acts as a layer of software between the physical hardware and the virtual machines.

- **Resource Management:**
    - The hypervisor manages the allocation of the host's resources (CPU, memory, storage) to each VM.
    - This ensures that each VM has the resources it needs to run, while also preventing VMs from interfering with each other.

- **Abstraction:**
  - ➢ It creates an abstraction layer, hiding the underlying hardware from the VMs.
  - ➢ This allows each VM to run its own operating system as if it were running on dedicated hardware.

- **Types of Hypervisors:**
  - ➢ **Type 1 (Bare-metal):** Runs directly on the host's hardware.
  - ➢ **Type 2 (Hosted):** Runs as an application within a host operating system.

In simpler terms, a hypervisor is like a traffic controller for your computer's resources, allowing multiple "virtual computers" to run smoothly on a single physical machine.

Ubuntu:

Ubuntu is a very popular Linux distribution

- **Linux Distribution:**
  - ➢ It's an operating system based on the Linux kernel. This means it's a complete software system that manages computer hardware and software resources.

- **Debian-based:**
  - ➢ Ubuntu is derived from the Debian Linux distribution, which is known for its stability.

- **Free and Open-Source:**
  - ➢ It's primarily composed of free and open-source software, meaning it's available for anyone to use, modify, and distribute

Practical 5: Installation and Configuration of virtualization using KVM.

Source code:

Software Tools Required

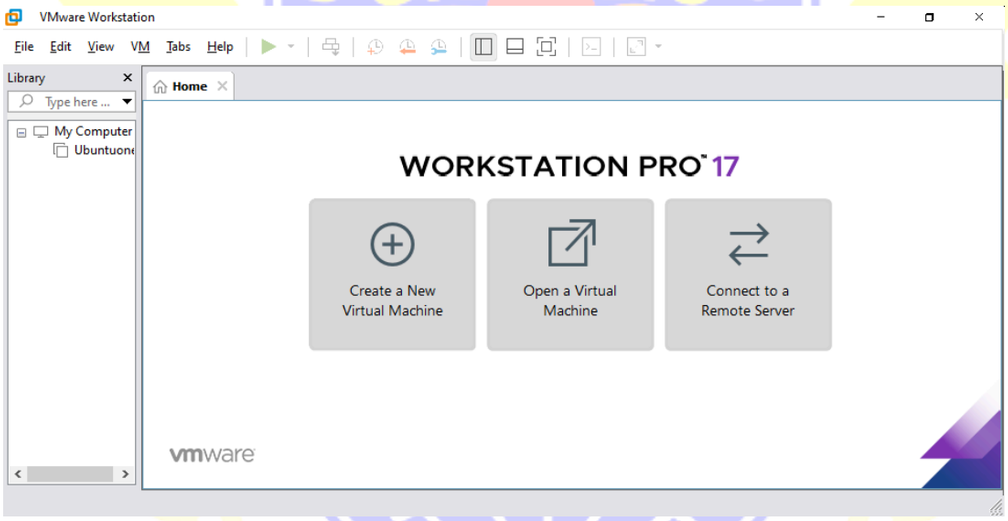
- **Virtual Machine Monitor:** VMware Workstation
- **Operating System:** Linux

Downloads Required:

- Virtual Machine Monitor**:** Download VMware Workstation Pro 17.6.0 Build 24238078 for Windows | Uptodown.com
- Operating System: Download Ubuntu Desktop | Ubuntu

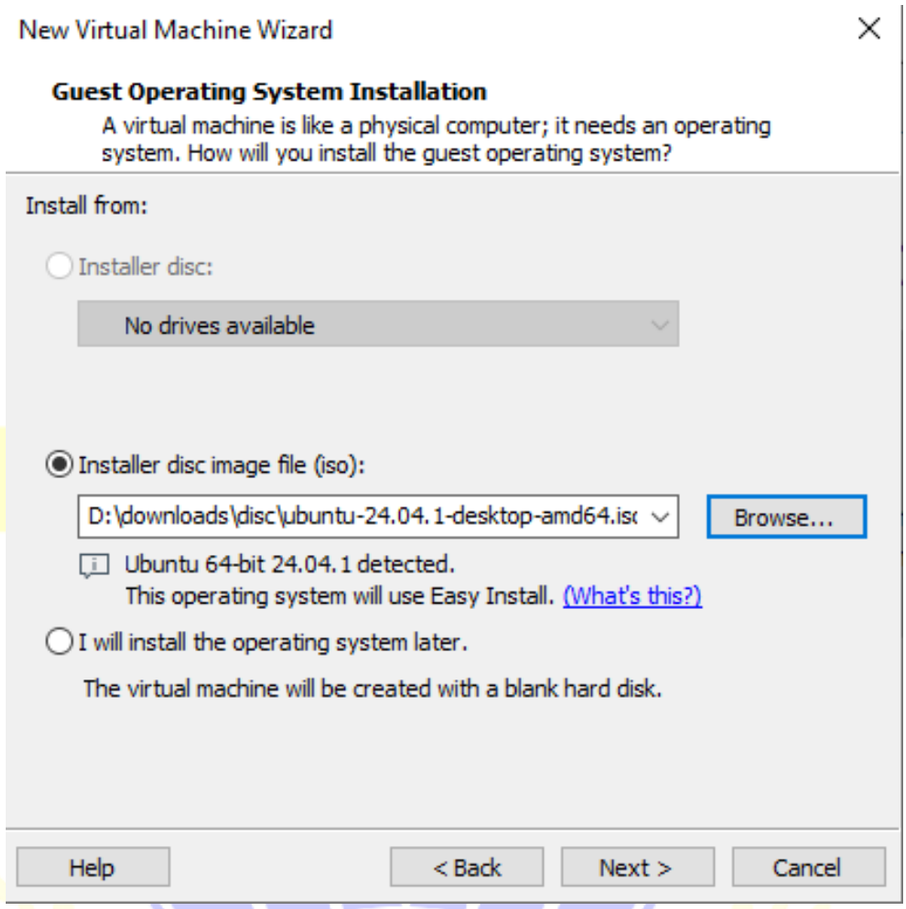
Open Vmware Workstation and start the steps

Demonstration:

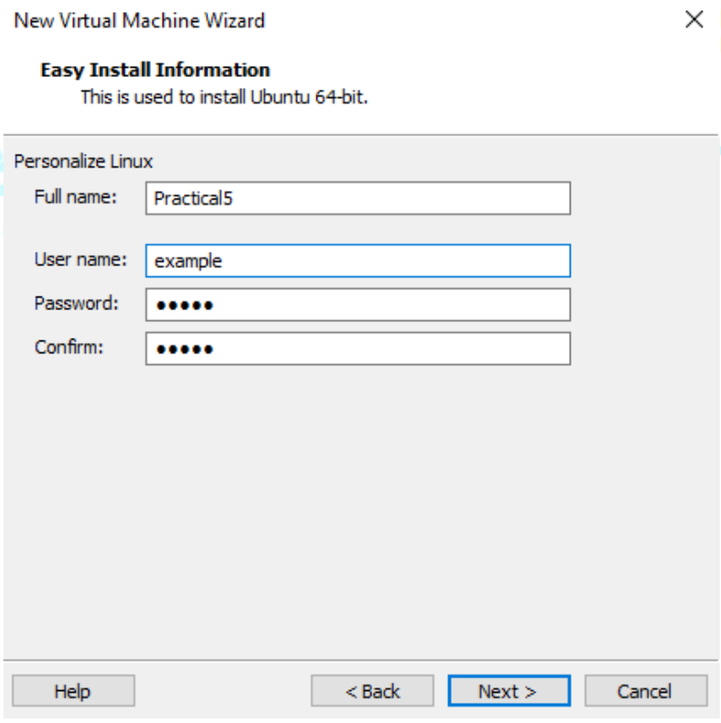


Create a New Virtual Machine:

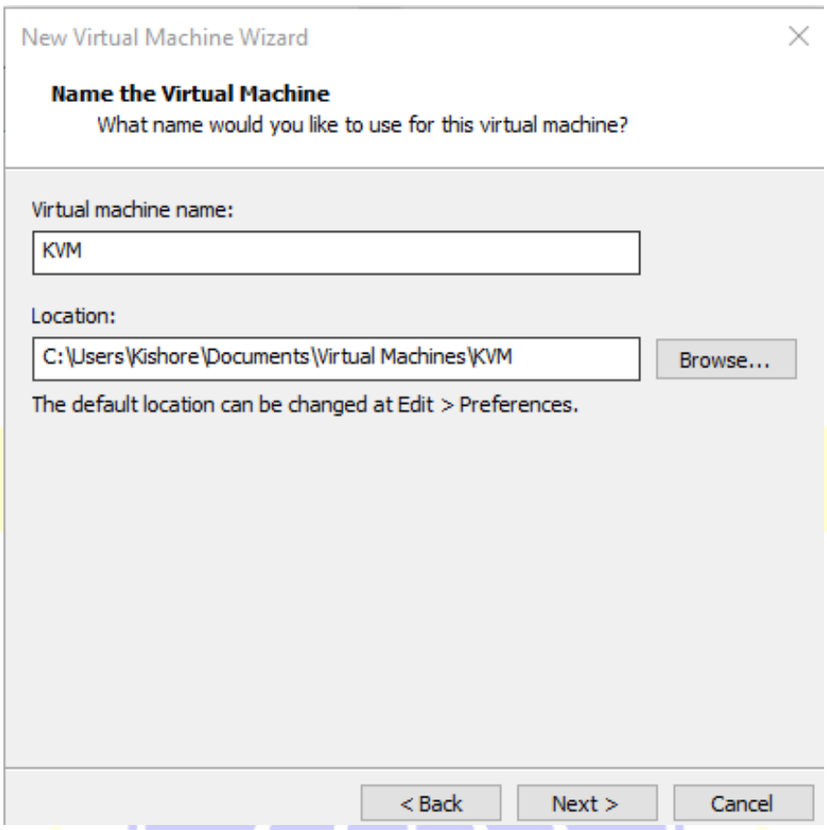
Browse and place the ISO File



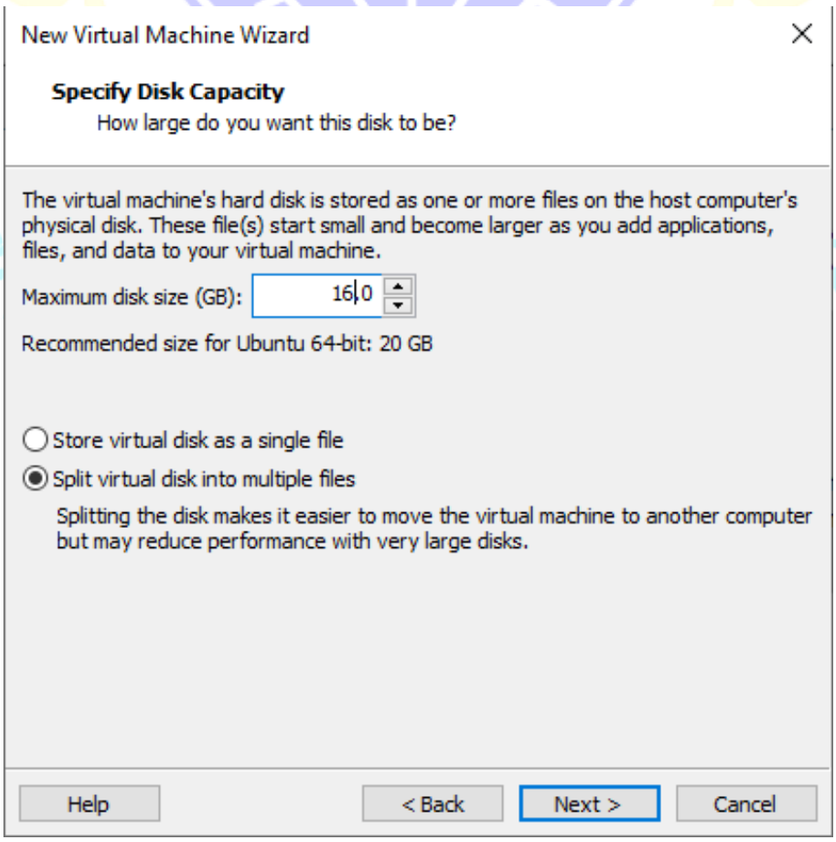
Provide full name as **Practical5** and user name as **example** and password and confirm password as **linux**



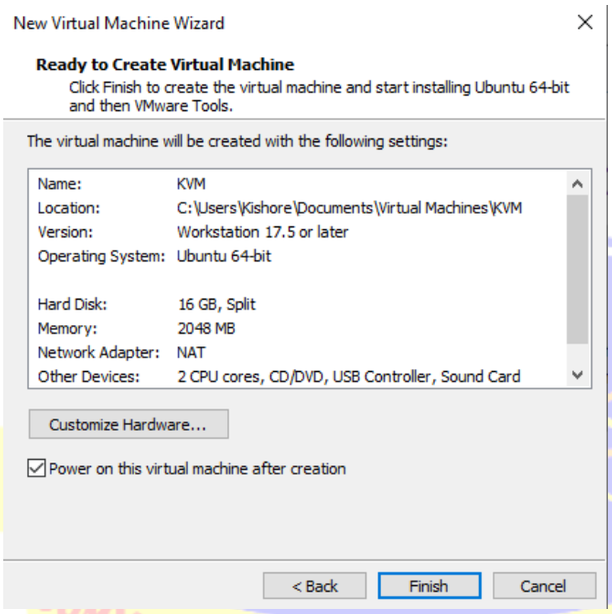
Give virtual machine name as **KVM**



Provide disk size: 16.0GB then click Next



Click on Customize Hardware

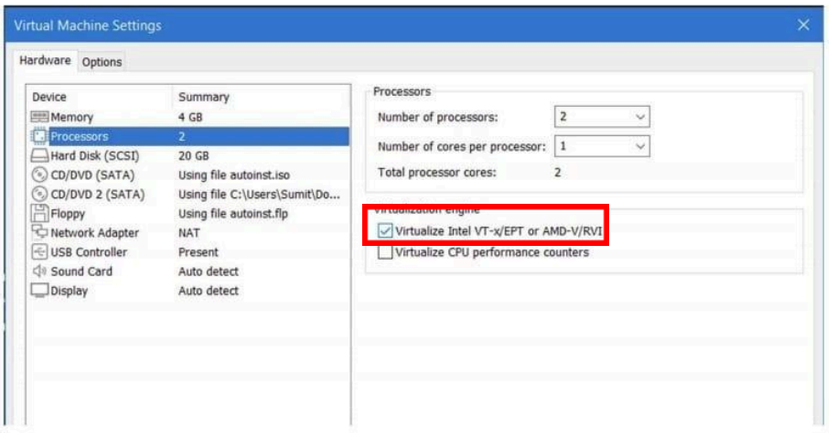


To enable KVM: Check the Virtualize Intel VT

For enabling KVM

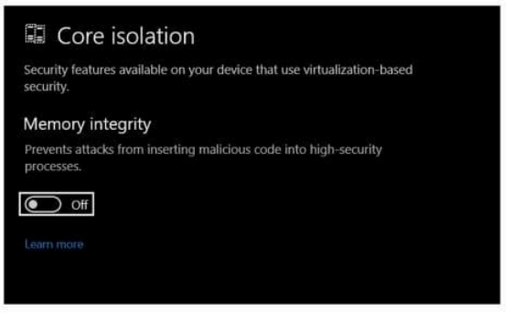
1.Edit virtual machine settings

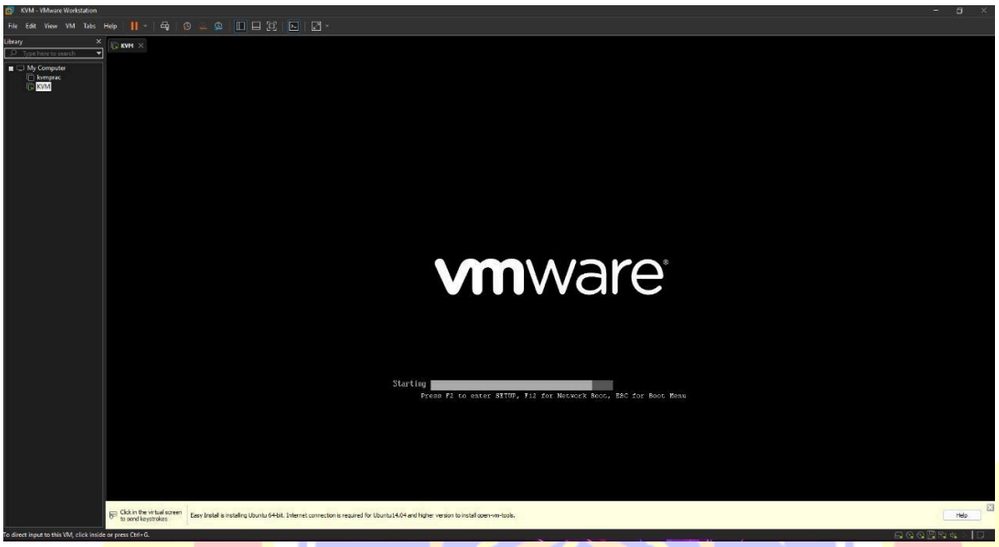
2.Select processor then tick the Virtualize Intel VT



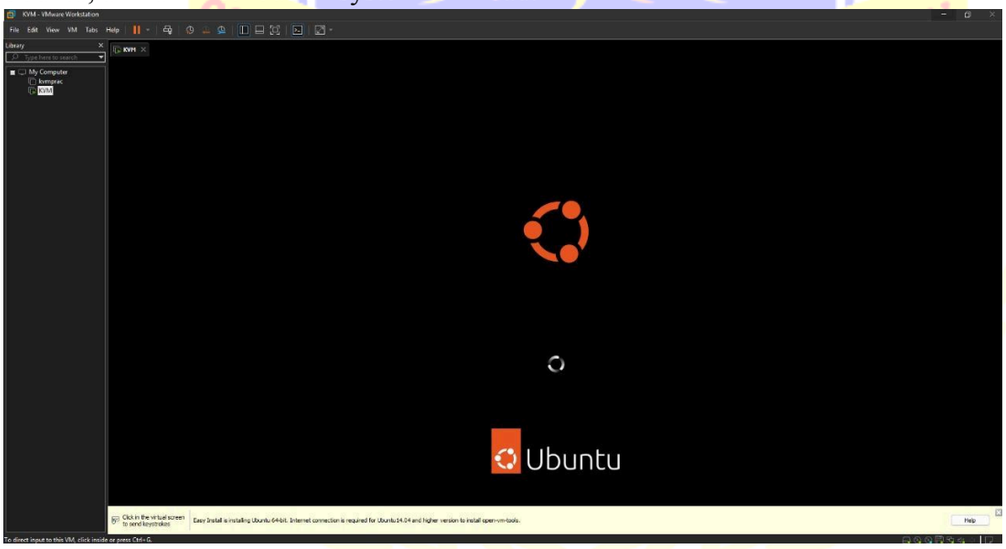
Select OK

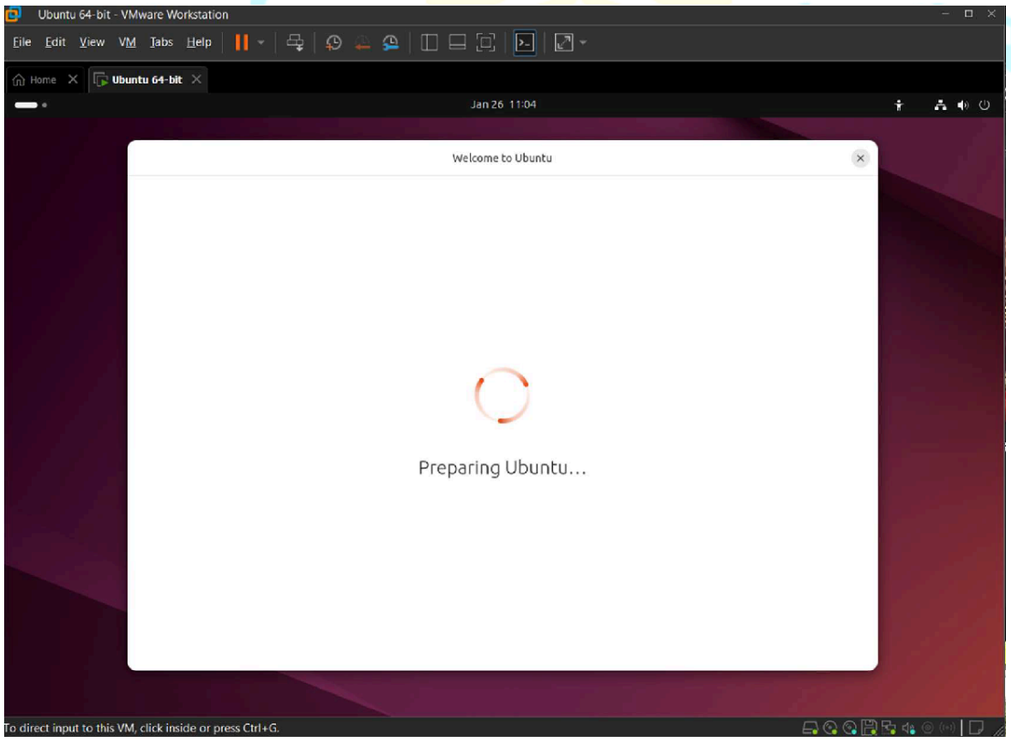
(In case You get an error like Intel VT not supported, Turn off the Memory Integrity from core isolation)





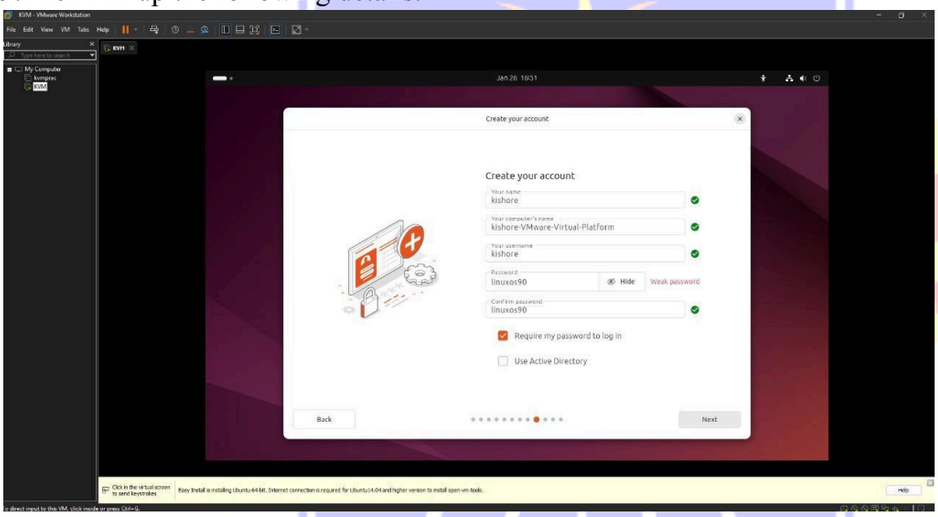
Ubuntu, a linux Distribution system will load.



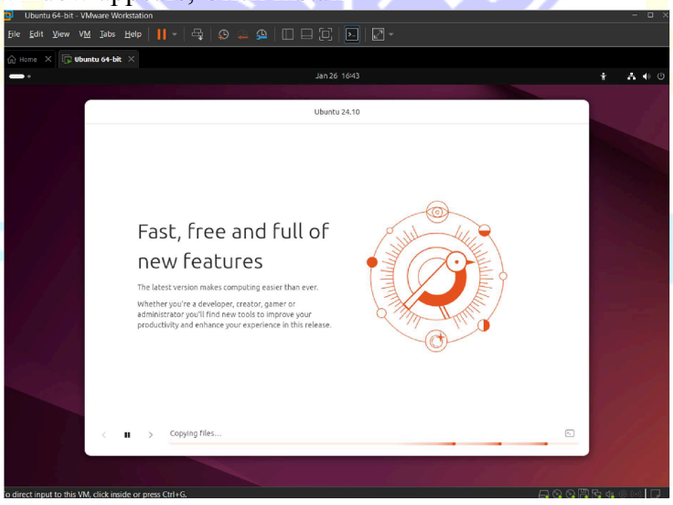


Ubuntu Setup:

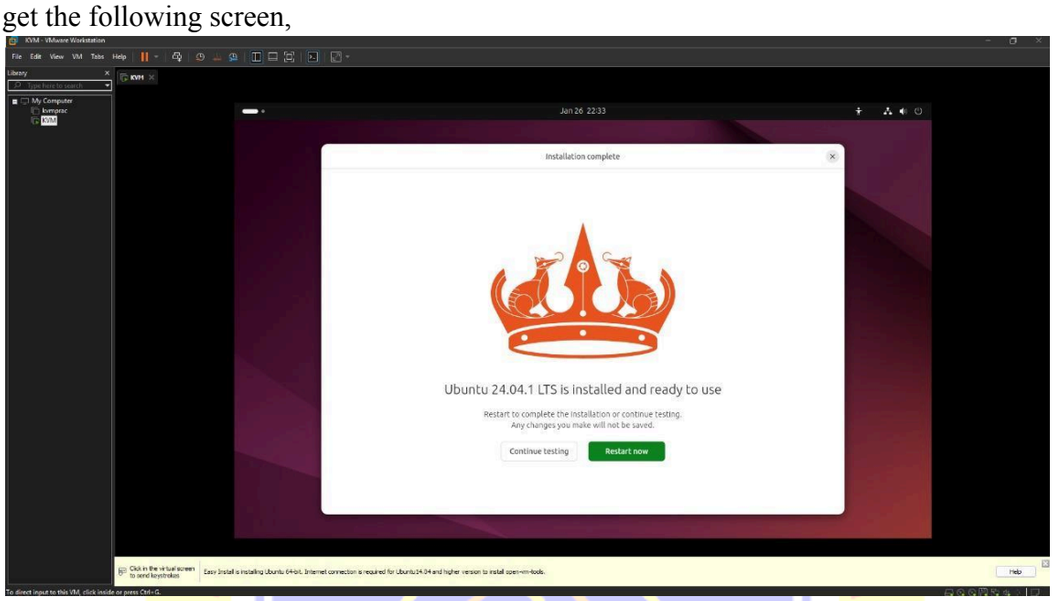
1.Choose Language then click next
2.Then in Accessibility Tab Click next
3.Keyboard Layout As English US then next
4.Connect to Internet then next
5.Tick Install ubuntu then next
6.Interactive then Default Installation
7.Install Proprietary Software (tick both options)
8.Then Tick Erase disk and install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



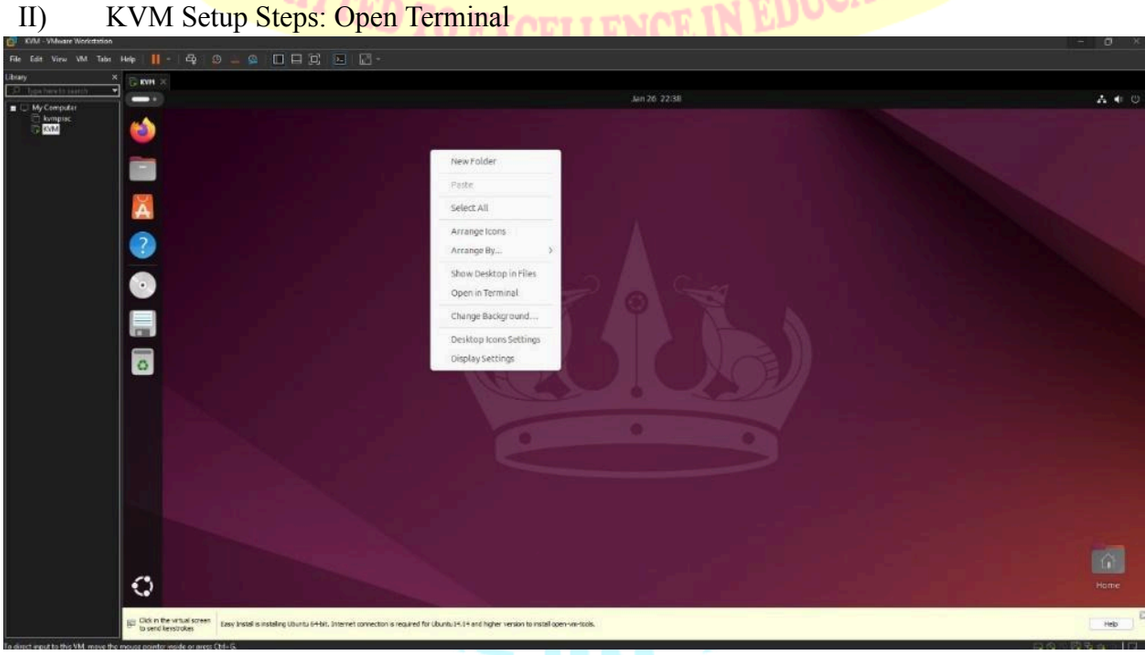
We get the following screen,



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 sydby 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

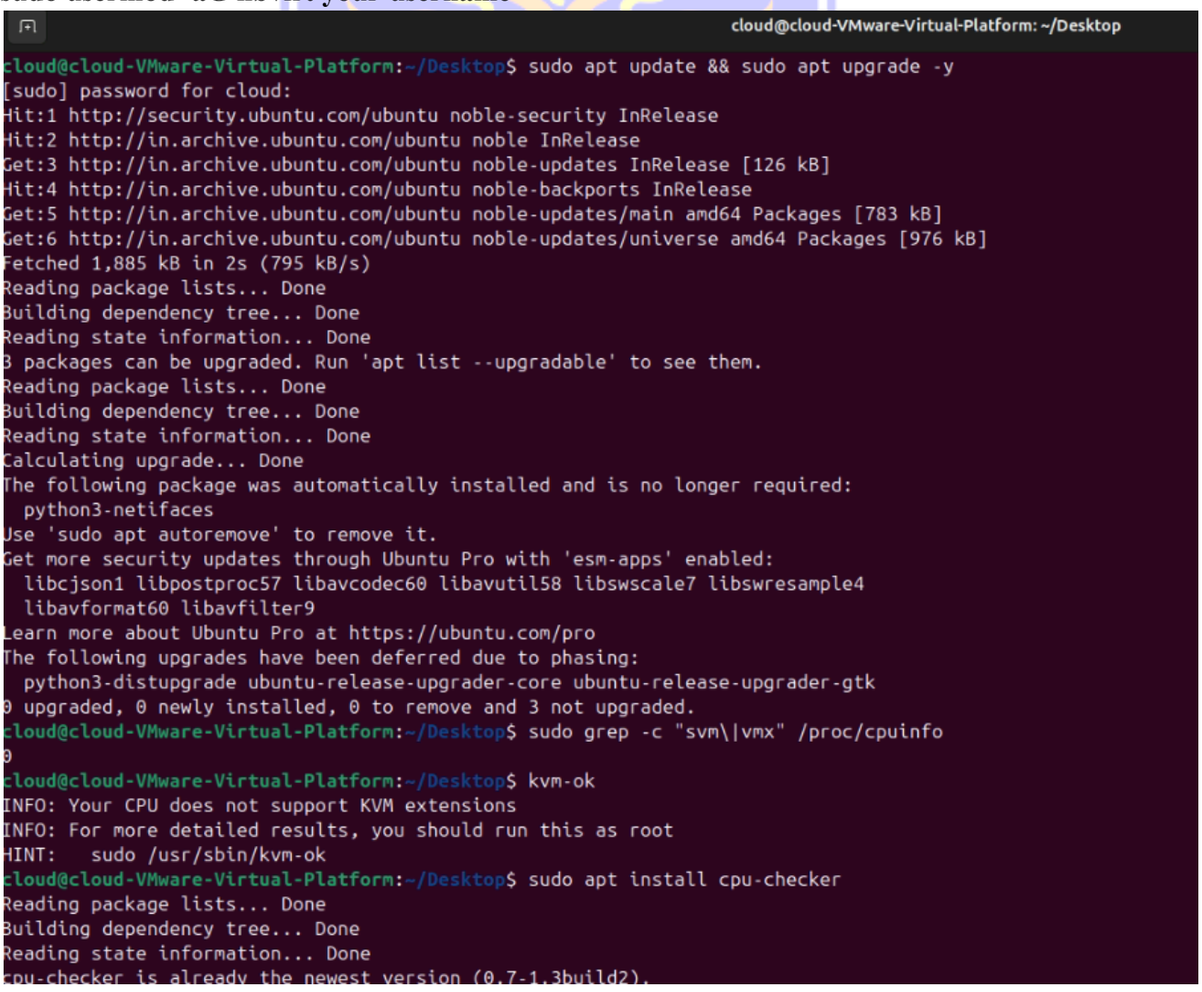
sudo systemctl start libvirtd

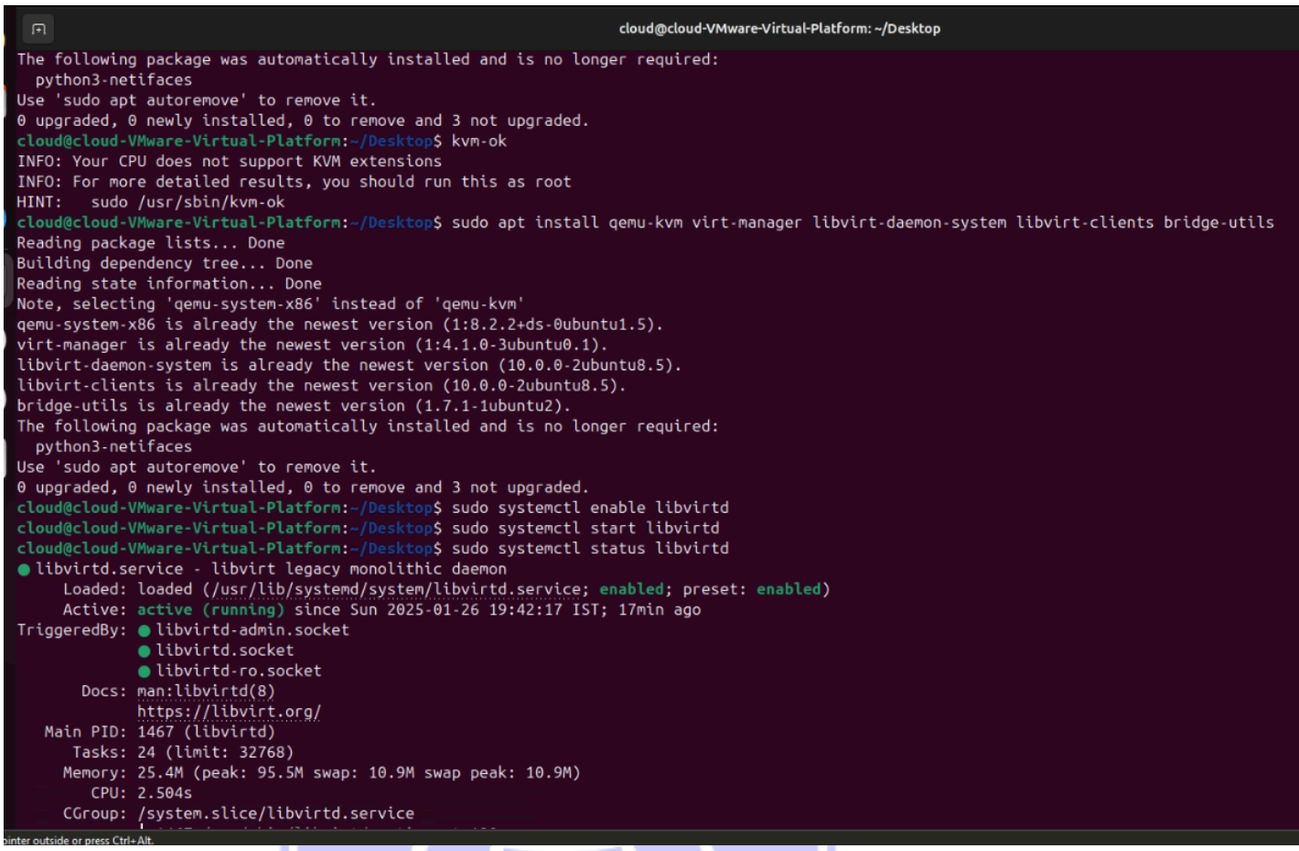
6. Check the Status of the Libvirt Daemon

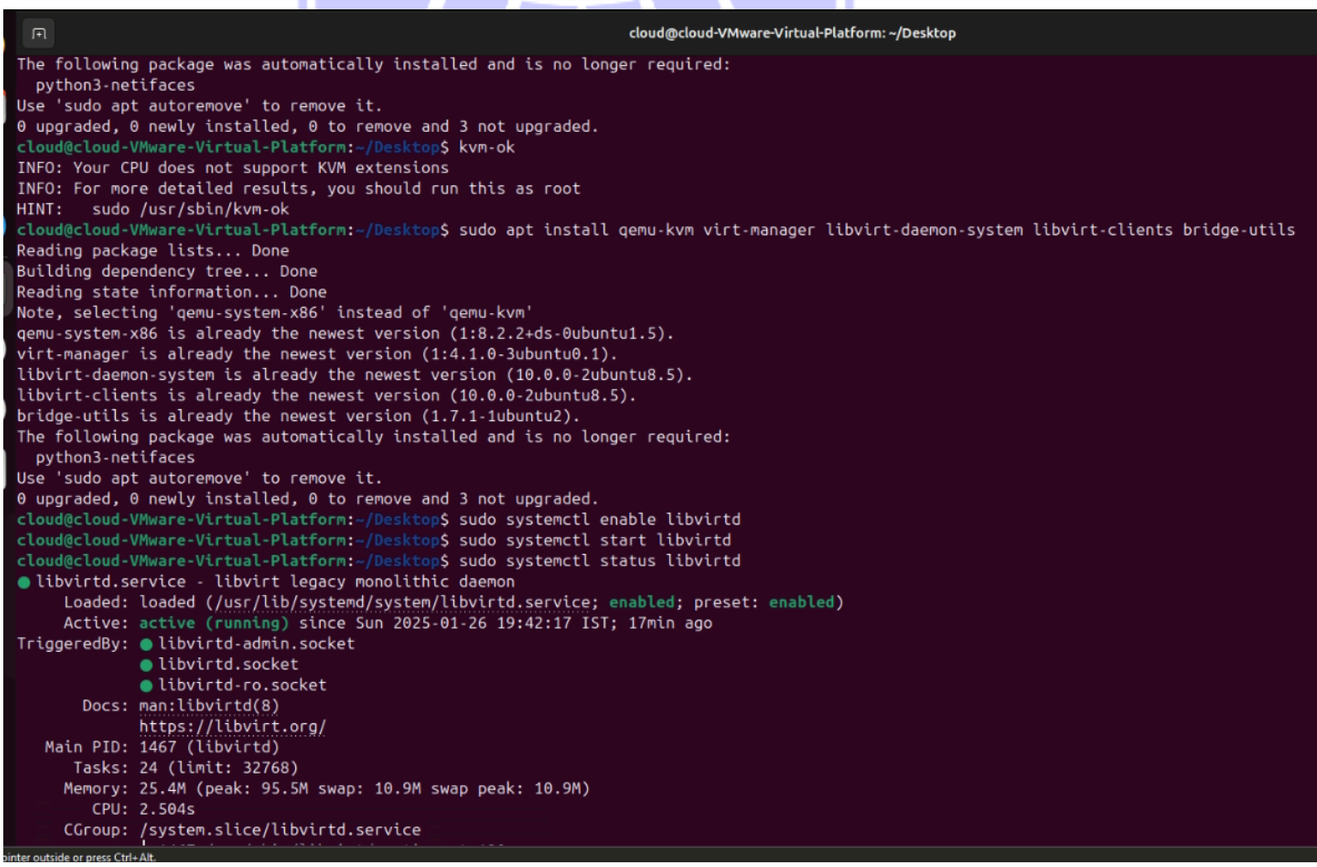
Verify that the daemon is running:
sudo systemctl status libvirtd

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

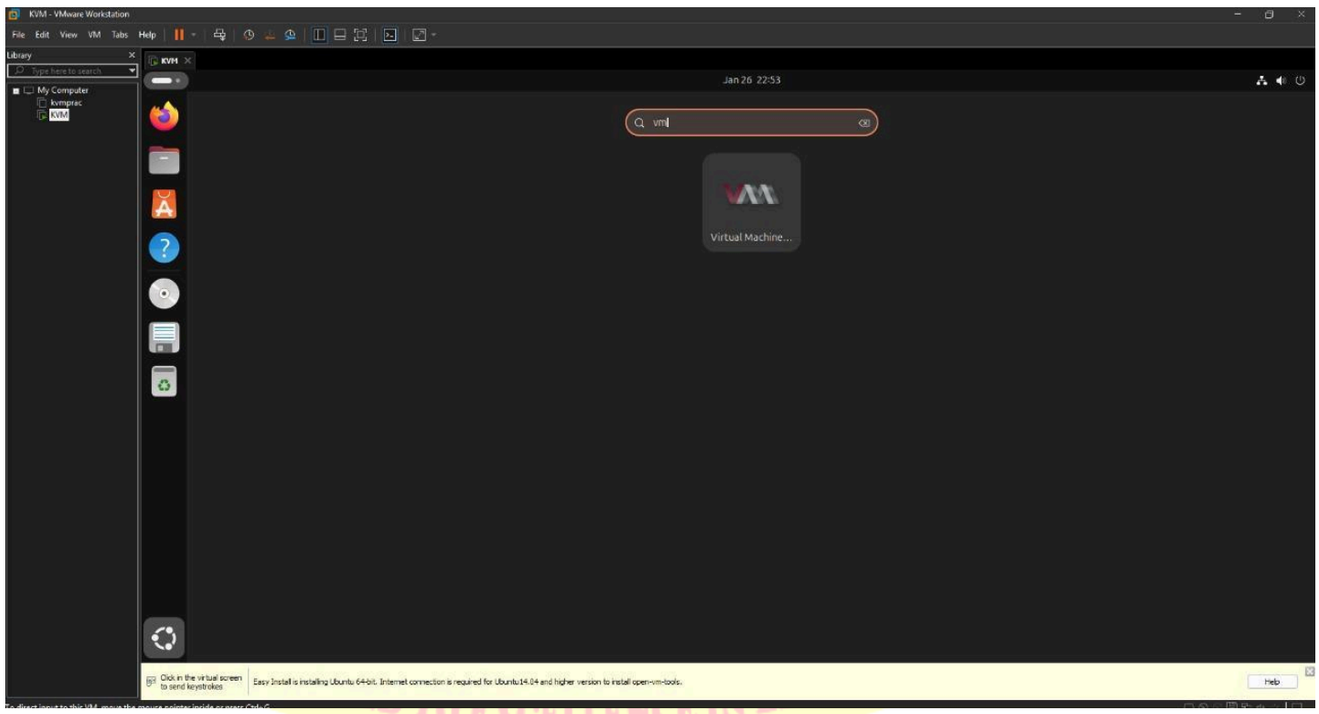


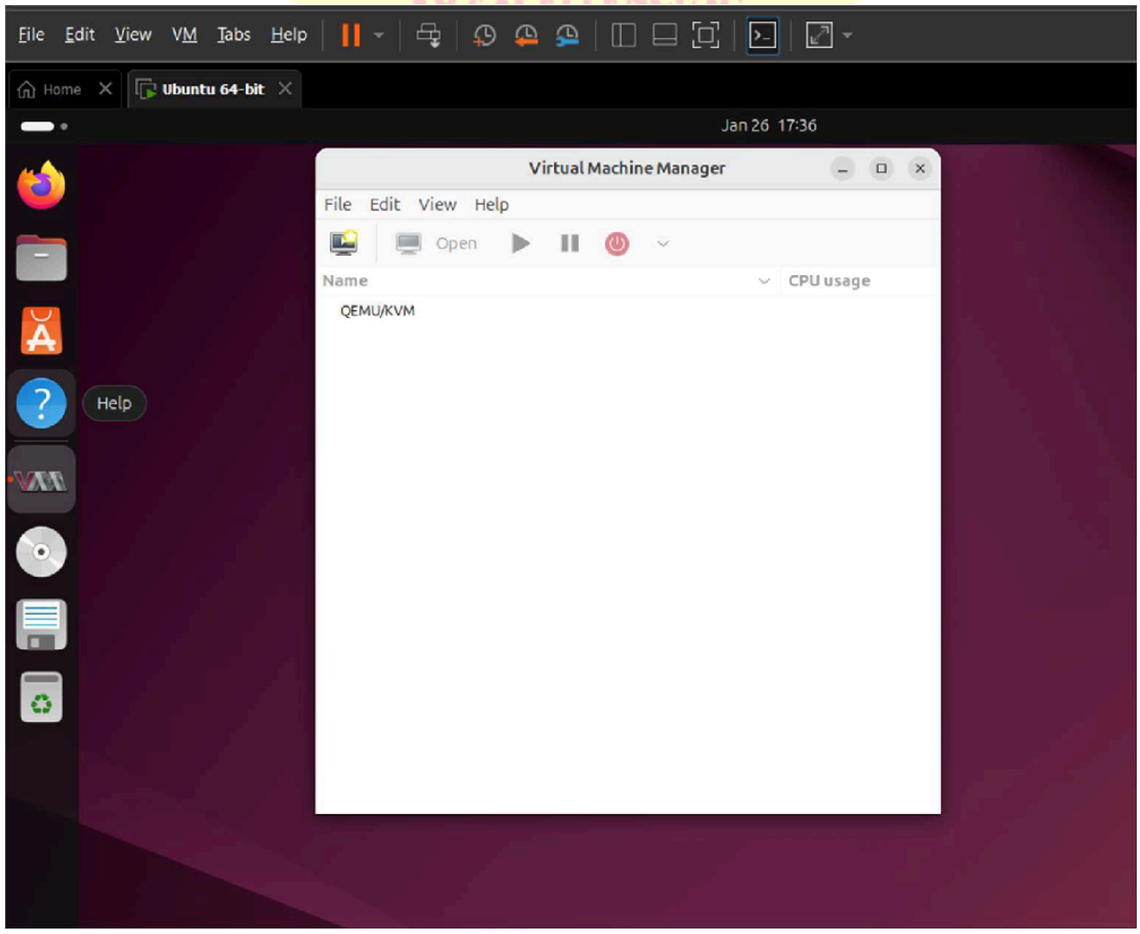


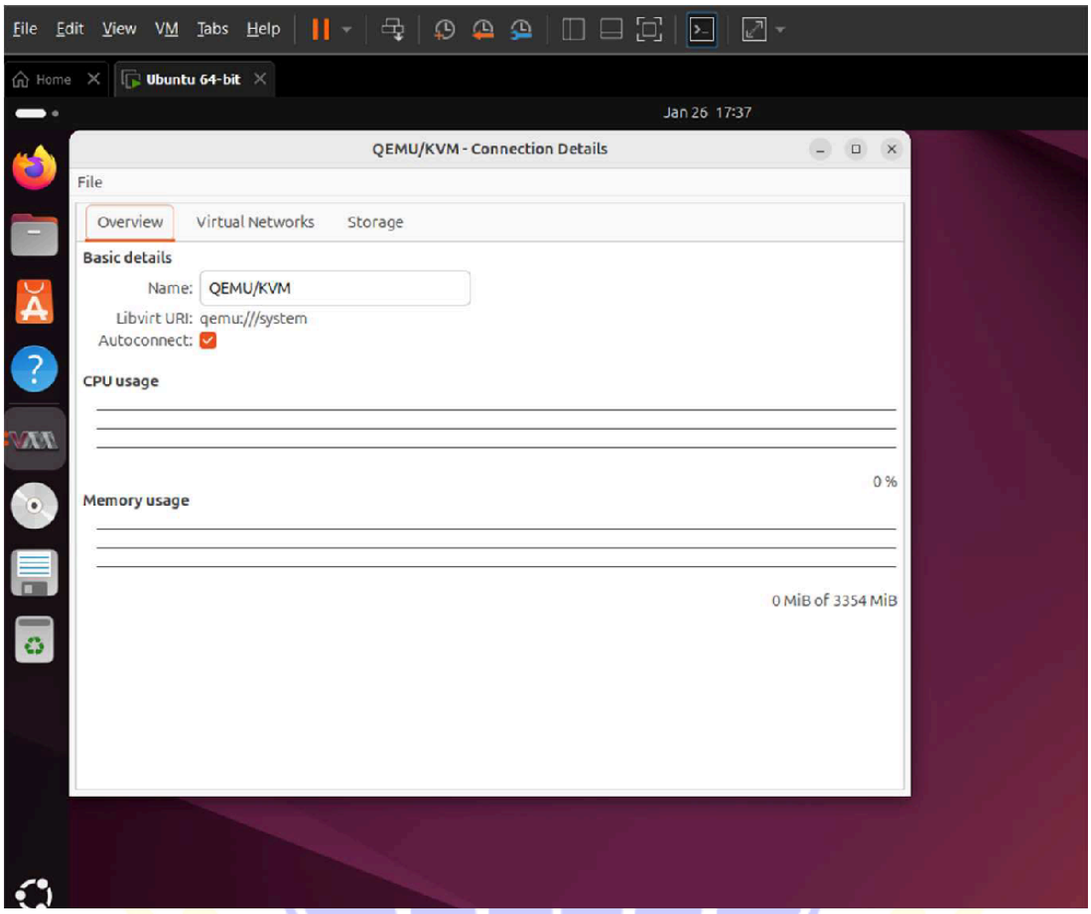


8. Log Out and Re-login
Log out from your system and log back in to apply group changes.

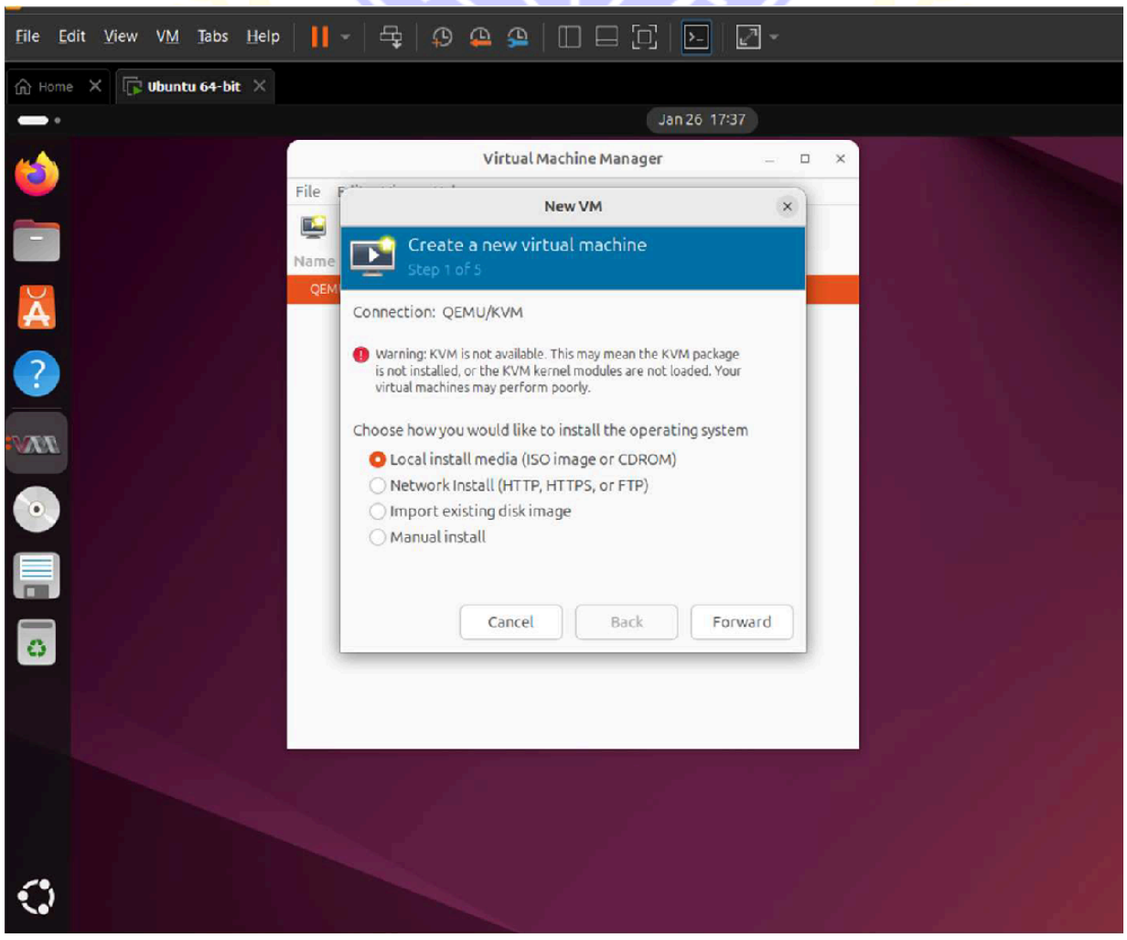
9. Run KVM Virtual Machine Manager
Search for "Virtual Machine Manager" in your system applications and launch it.
10. Prepare to Create Virtual Machines
In Virtual Machine Manager, click on File -> New Virtual Machine to begin setting up virtual environments.


KVM - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
kvmprac
KVM
KVM
Jan 26 22:53
vmi
Virtual Machine...
Click in the virtual screen to send keystrokes
Easy Install is installing Ubuntu 64-bit. Internet connection is required for Ubuntu14.04 and higher version to install open-vm-tools.
Help

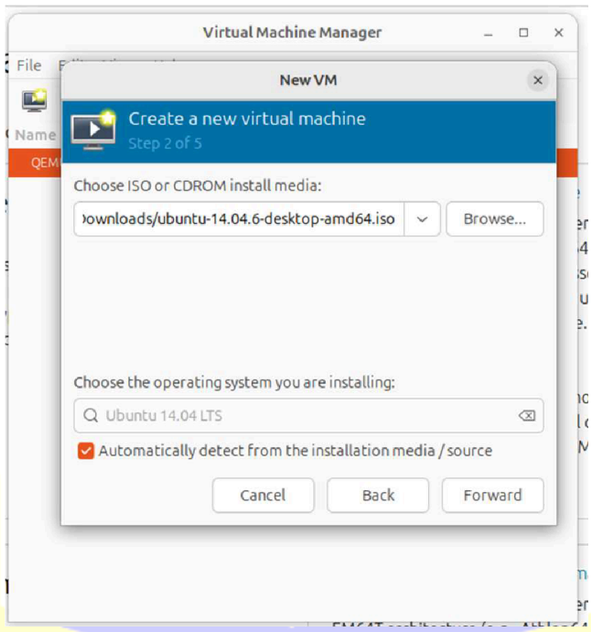

File Edit View VM Tabs Help
Home
Ubuntu 64-bit
Jan 26 17:36
Virtual Machine Manager
File Edit View Help
Open
Name
CPU usage
QEMU/KVM
Help

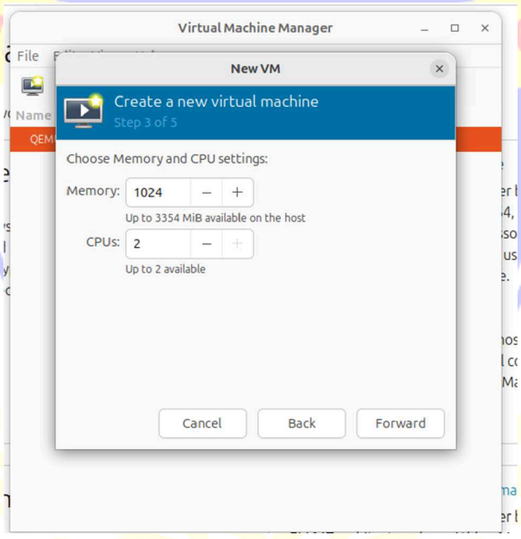


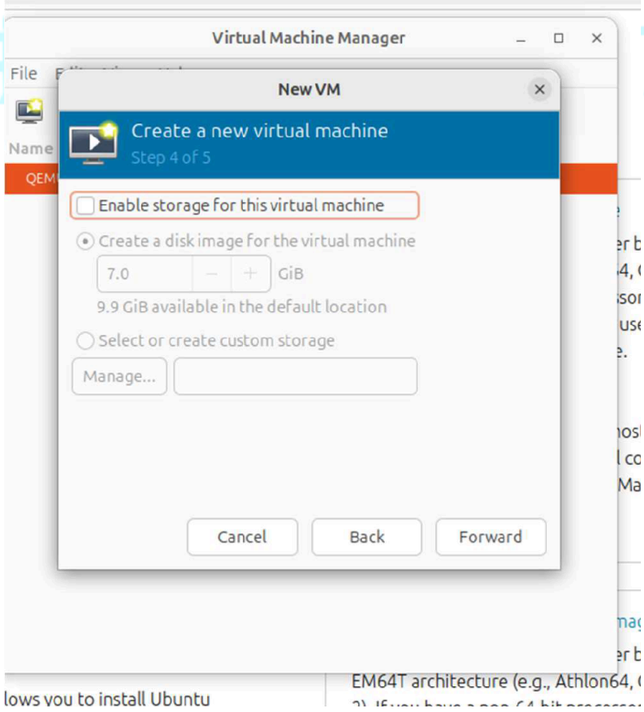
Now we can create new virtual machine from File>Create New Virtual Machine



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

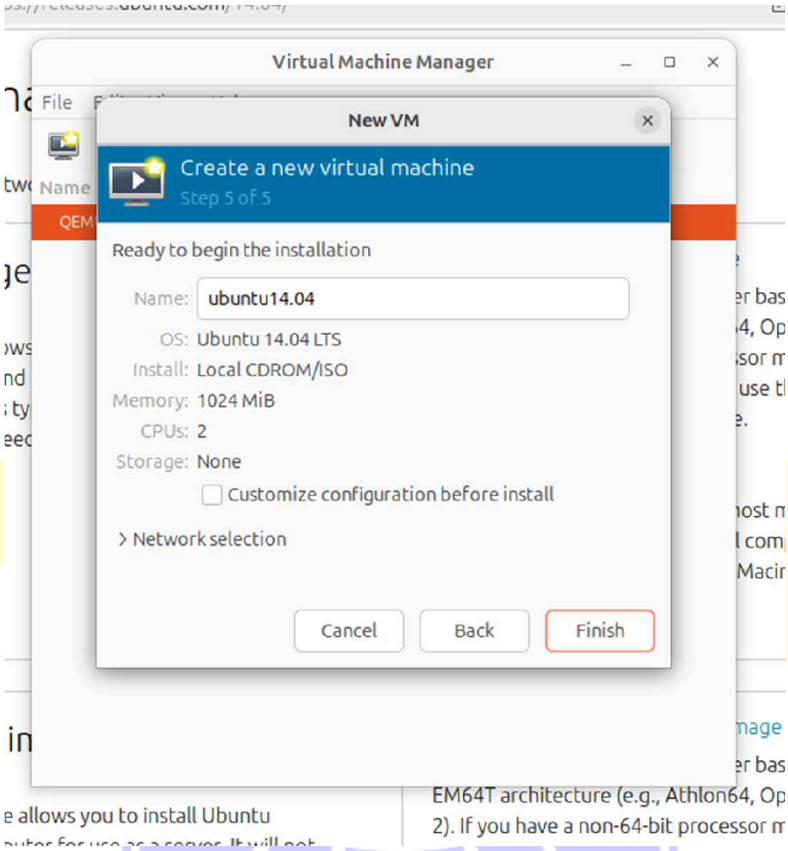


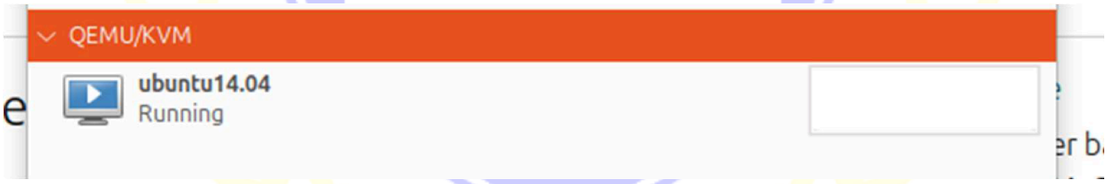


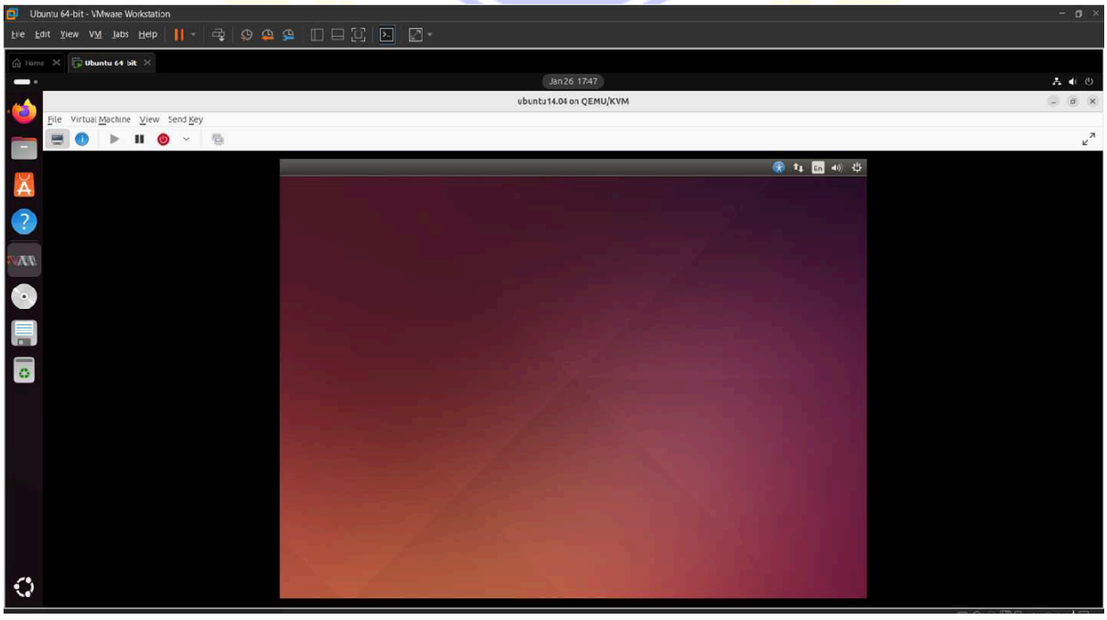


lows you to install Ubuntu

EM64T architecture (e.g., Athlon64,


Virtual Machine Manager
File
New VM
Create a new virtual machine
Step 5 of 5
Ready to begin the installation
Name: ubuntu14.04
OS: Ubuntu 14.04 LTS
Install: Local CDROM/ISO
Memory: 1024 MiB
CPUs: 2
Storage: None
Customize configuration before install
Network selection
Cancel
Back
Finish


QEMU/KVM
ubuntu14.04
Running


Ubuntu 64-bit - VMware Workstation
ubuntu14.04 on QEMU/KVM

Result and Discussion: We have successfully installed ubuntu and configured a virtual machine.

Learning Outcomes:
1. Successfully implemented virtualization.
2. Understood virtualization.

Course Outcomes:
1. We have learned about virtualization.
2. We have learned about KVM.

Conclusion:
We have successfully installed ubuntu and configured a virtual machine.

Viva Question:
1. What is virtualization?
2. What is virtual machine?
3. What is hypervisor?
4. What is KVM?

For Faculty Use

Correction Parameters	Formative Assessment [ ]	Timely completion practical [ ]	Attendance Learning Attitude[ ]

Theory-6
MTOM

The Problem: Sending Big Stuff in SOAP

Imagine you're sending a letter (SOAP message) to a friend. Inside, you want to include a photo (binary data).

- **Traditional SOAP (Without MTOM):**
  - You'd take the photo, convert it into a long string of text (base64 encoding), and paste that string into your letter.
  - This makes the letter huge and hard to read.
  - It also wastes space, because base64 encoding inflates the size of the data.
  - This process is slow, especially for big photos or videos.

MTOM: The Solution

MTOM is like sending the photo in a separate envelope attached to the letter.

- **How it Works:**
  - The letter (SOAP message) contains a reference (a pointer) to the photo.
  - The photo itself is sent as a separate, raw binary file.
  - The letter and the photo are sent together as a "multipart" message.
  - The receiving side puts the letter and the photo back together.

Intuition:

- **Efficiency:** MTOM avoids the overhead of encoding binary data as text. This saves space and speeds up transmission.
- **Clarity:** The SOAP message stays relatively small and readable, because it doesn't contain the bulky binary data.
- **Optimization:** It optimizes the transmission of large binary data within SOAP messages.

Think of it like this:

- **SOAP (Without MTOM):** Trying to stuff an entire pizza into a small mailbox.
- **MTOM:** Sending the pizza in its own box, with a note in the mailbox saying, "The pizza is in the box next to it."

In technical terms:

- MTOM is a W3C standard that defines a way to optimize the transmission of XML messages that contain binary data.
- It is very often used with SOAP web services.

Key Benefits:

- **Reduced Message Size:** Smaller messages mean faster transmission.
- **Improved Performance:** Less encoding/decoding overhead.
- **Better Handling of Large Files:** More efficient for images, videos, and other binary data.

When to Use MTOM:

- When you need to send large binary data (images, videos, files) in SOAP messages.
- When performance is important.

Practical-6: Develop application to download image/video from server or upload image/video to server using MTOM techniques.

Software Tools Required:

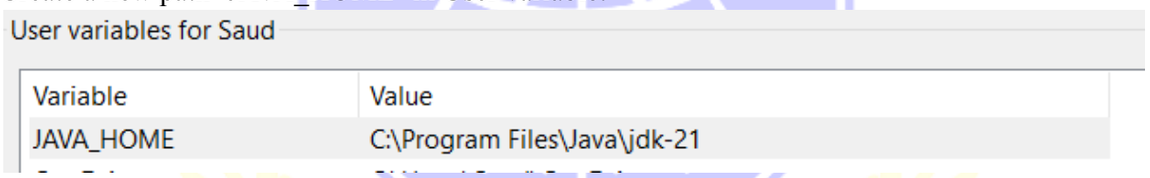
- **Code Editor:** Eclipse IDE
- **Build Automation Tool:** Apache Maven
- **SOAP Testing Tool:** SOAPUI
- **Framework:** Apache CXF
- **Programming Language:** Java

Downloads Required:

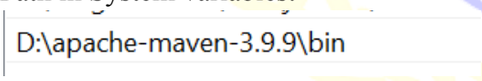
- Apache Maven: Download Apache Maven – Maven
- JDK: Java Downloads | Oracle India (x64 MSI Installer)
- Eclipse IDE: Eclipse downloads - Select a mirror | The Eclipse Foundation
- SOAP Testing Tool: Download REST & SOAP Automated API Testing Tool | Open Source | SoapUI (SoapUI Open Source)

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:
Create a new path `JAVA_HOME` in User variable:



Path in System variables:



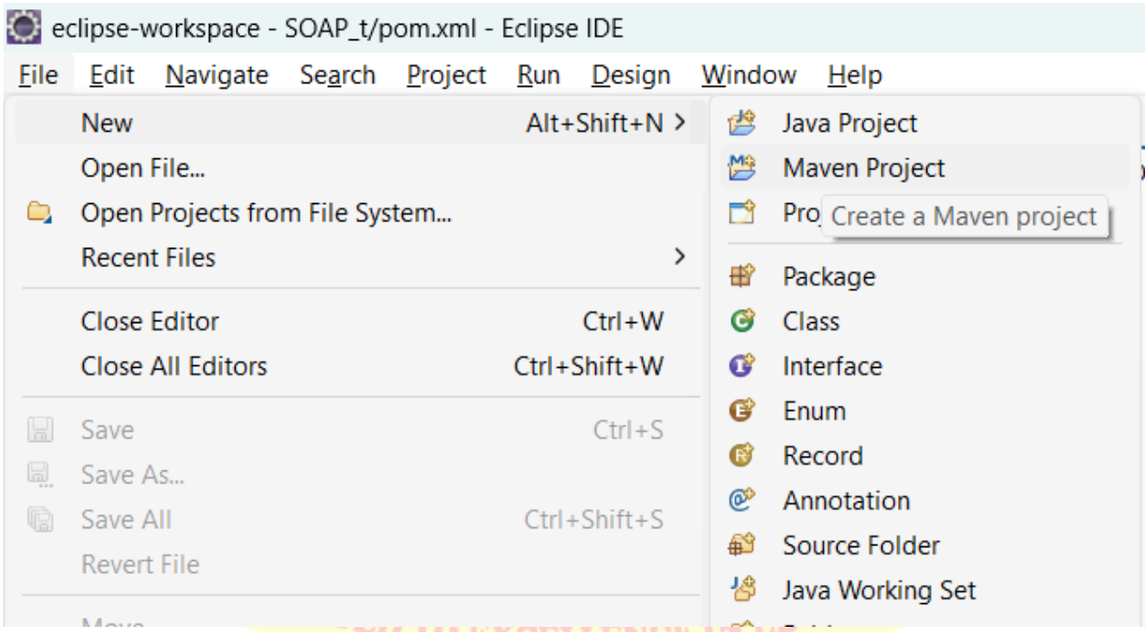
C:\Program Files\Common Files\Oracle\Java\javapath

Version checked in command prompt

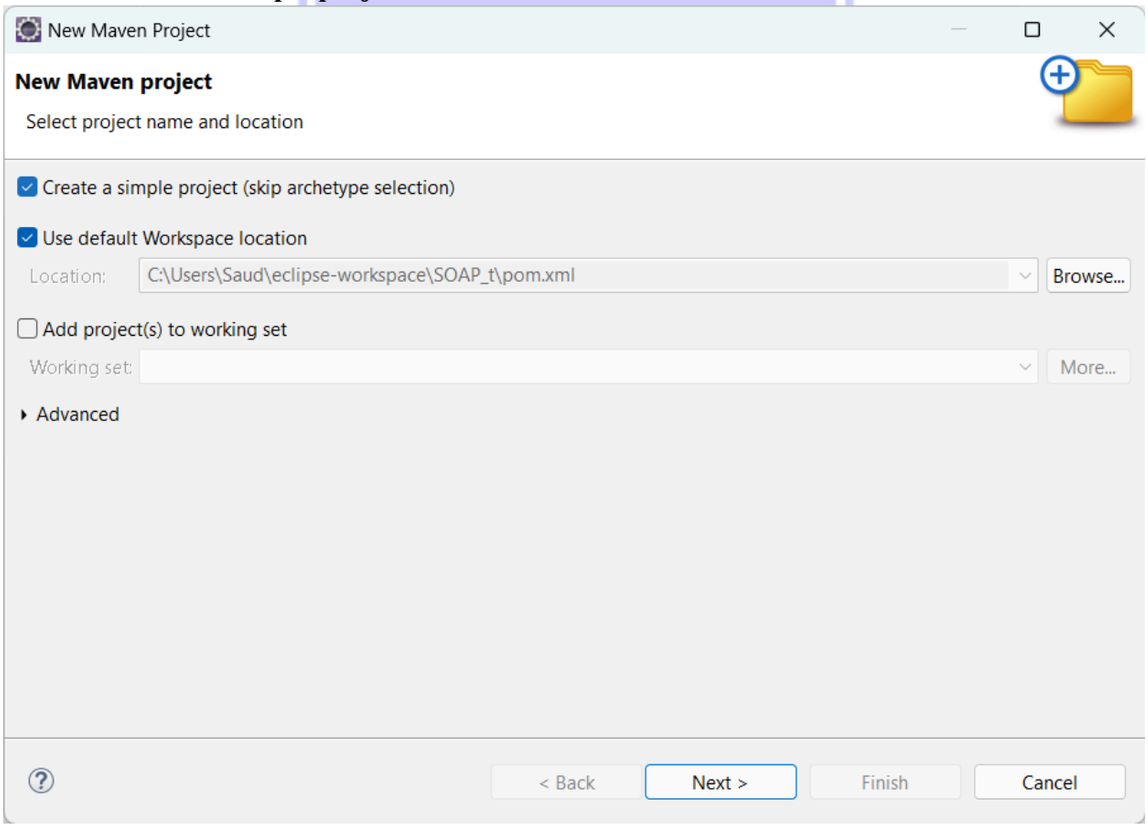
```
C:\Users\Saud>mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: C:\Program Files\apache-maven-3.9.9-bin\apache-maven-3.9.9
Java version: 21.0.6, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-21
Default locale: en_IN, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

C:\Users\Saud>java --version
java 21.0.6 2025-01-21 LTS
Java(TM) SE Runtime Environment (build 21.0.6+8-LTS-188)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.6+8-LTS-188, mixed mode, sharing)
```

Implementing the Service: Open Eclipse IDE and follow the steps

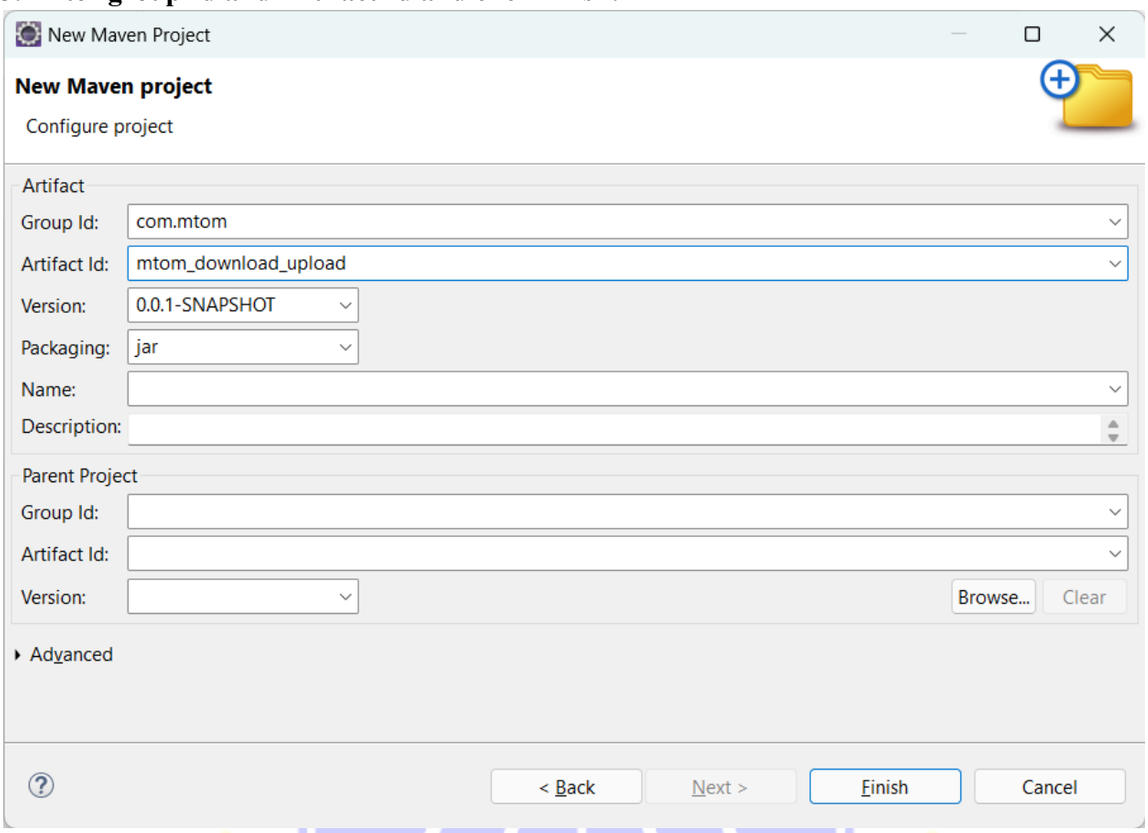
1. Goto File -> New -> Maven Project



2 . Select Create a simple project and click next



3. Enter group Id and Artifact Id and click finish:



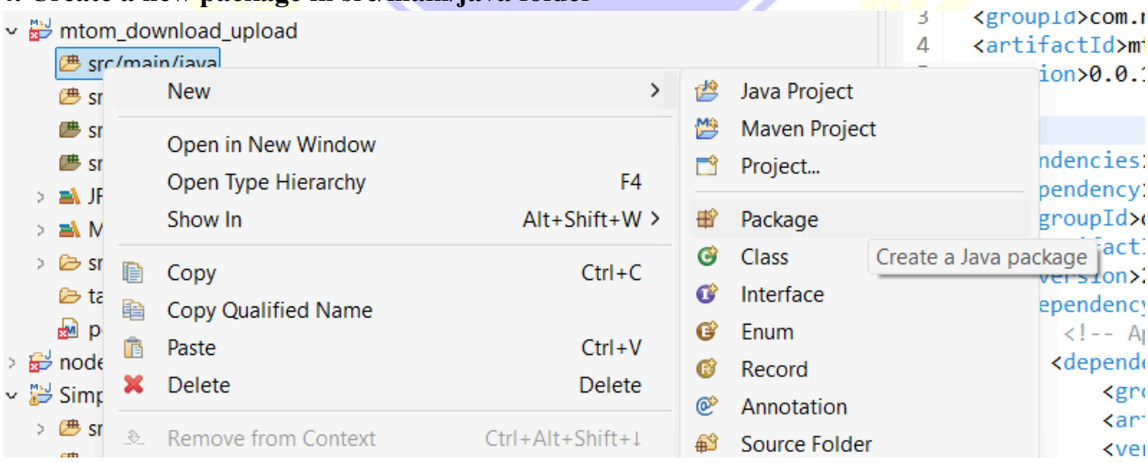
It will create`mtom_download_upload` folder

In the `mtom_download_upload` folder there will be a pom.xml file add dependencies in it as follow:

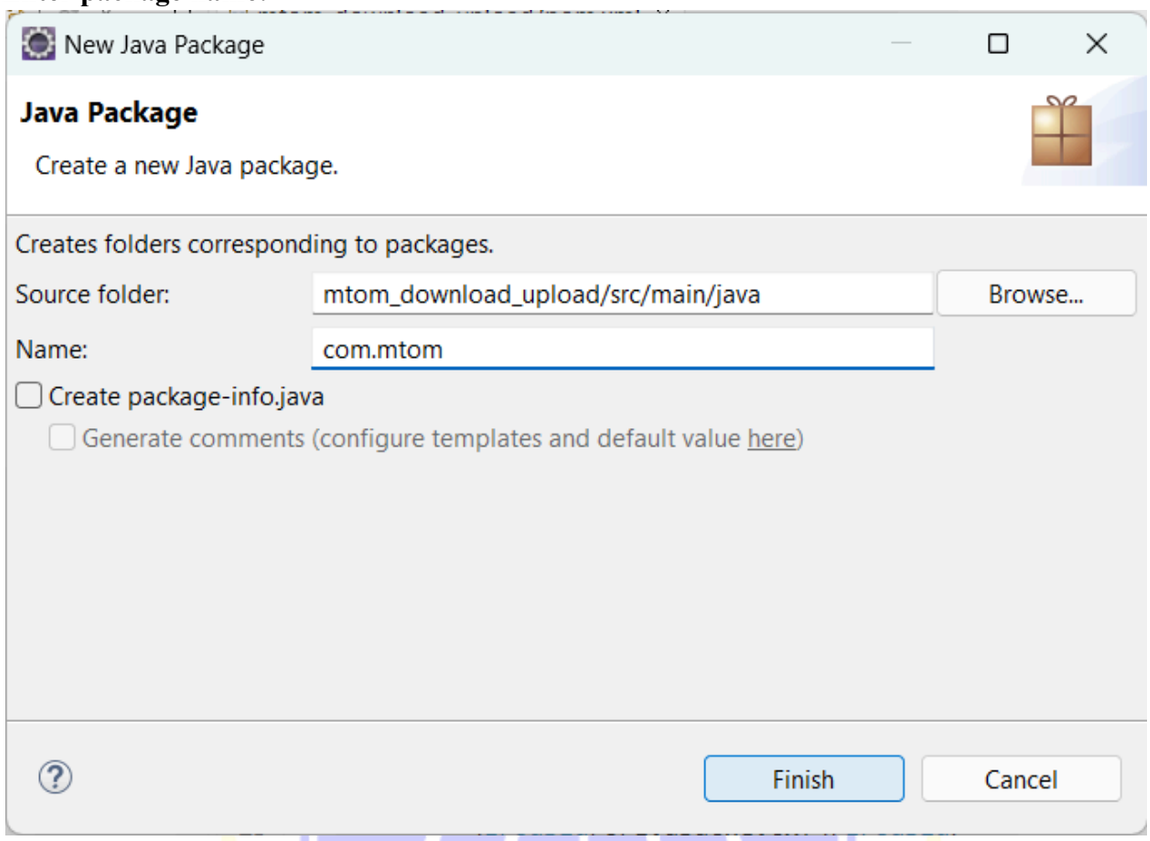
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.mtom</groupId>
 <artifactId>mtom_download_upload</artifactId>
 <version>0.0.1-SNAPSHOT</version>

 <dependencies>
  <dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-simple</artifactId>
   <version>2.1.0-alpha1</version>
  </dependency>
     <!-- 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-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>cxf-rt-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>
      <!-- Java 11+ HTTP Client -->
  <dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.13</version>
  </dependency>

<dependency>
  <groupId>jakarta.activation</groupId>
  <artifactId>jakarta.activation-api</artifactId>
  <version>1.2.2</version>
</dependency>
<dependency>
  <groupId>com.sun.activation</groupId>
  <artifactId>jakarta.activation</artifactId>
  <version>1.2.2</version>
</dependency>
 </dependencies>
</project>
```

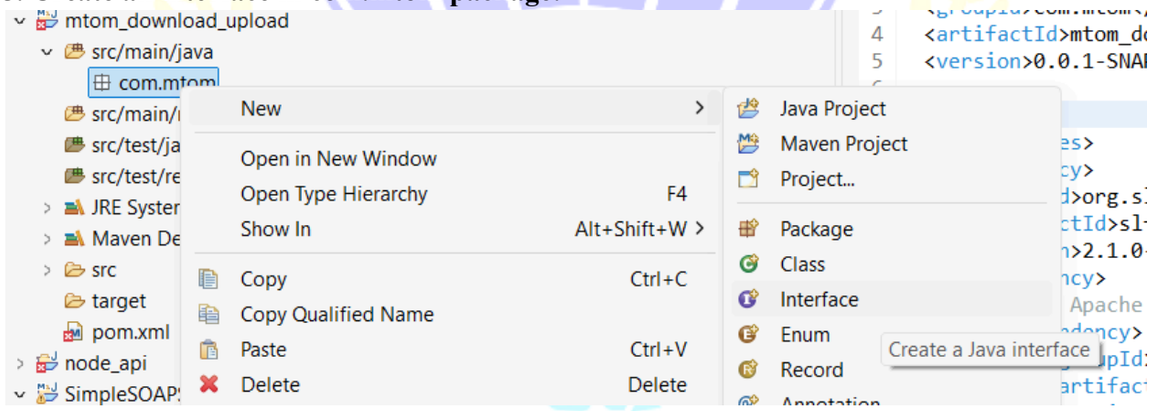
4. Create a new package in src/main/java folder



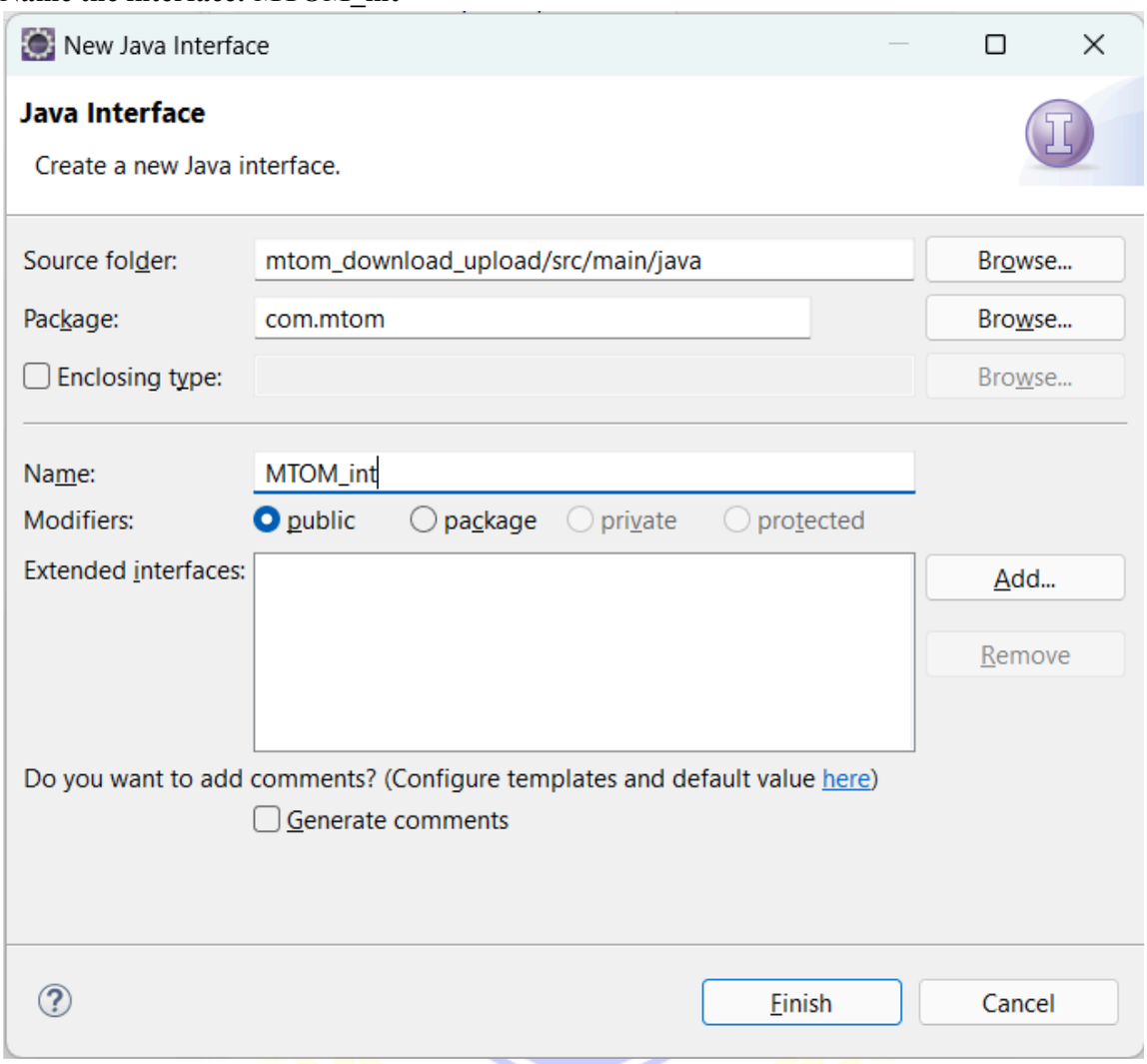
Enter package name:



5. Create an interface in com.mtom package:



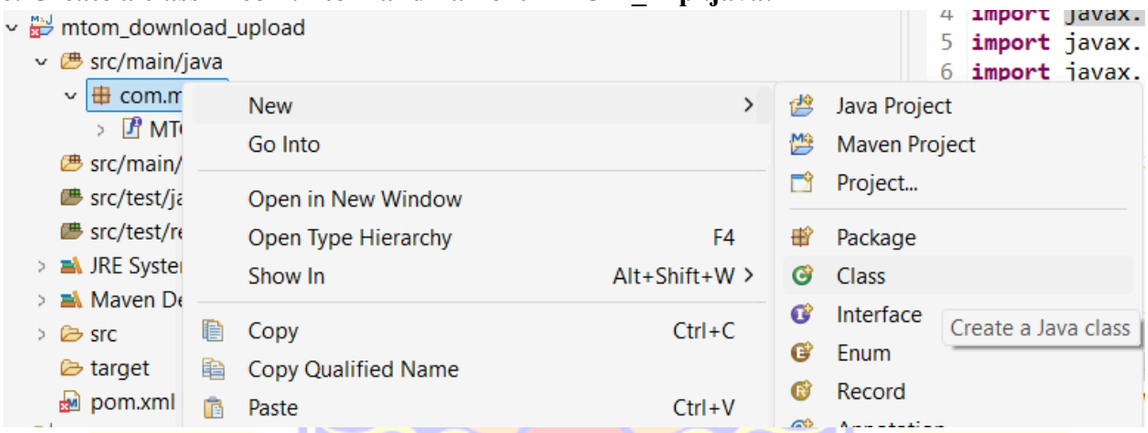
Name the interface: MTOM_int



MTOM_int: source code

```
package com.mtom;
import java.io.IOException;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;
import javax.jws.soap.SOAPBinding.Use;
import javax.xml.ws.soap.MTOM;

@MTOM
@WebService
@SOAPBinding(style = Style.DOCUMENT, use = Use.LITERAL)
public interface MTOM_int {
  @WebMethod
  byte[] downloadImage(String fileName) throws IOException;

  @WebMethod
  String uploadImage(byte[] imageData, String fileName) throws IOException;
}
```

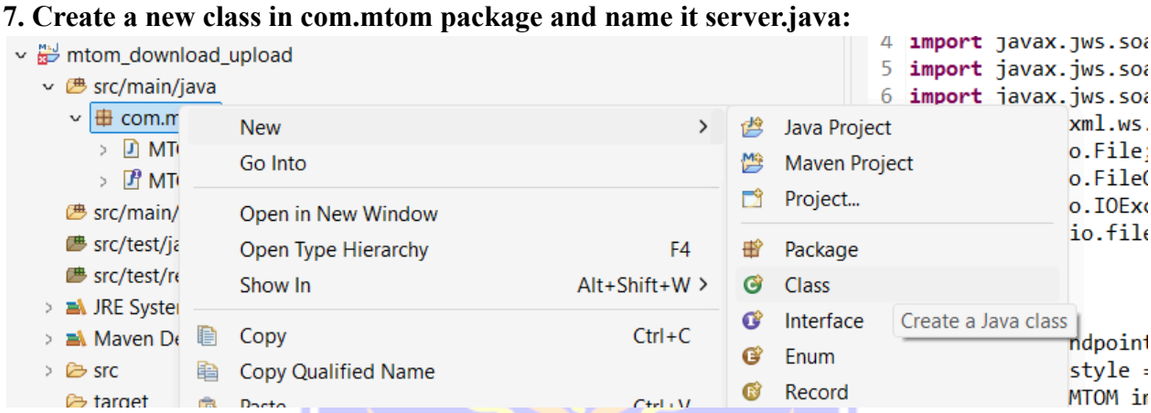
6. Create a class in com.mtom and name it MTOM_impl.java:

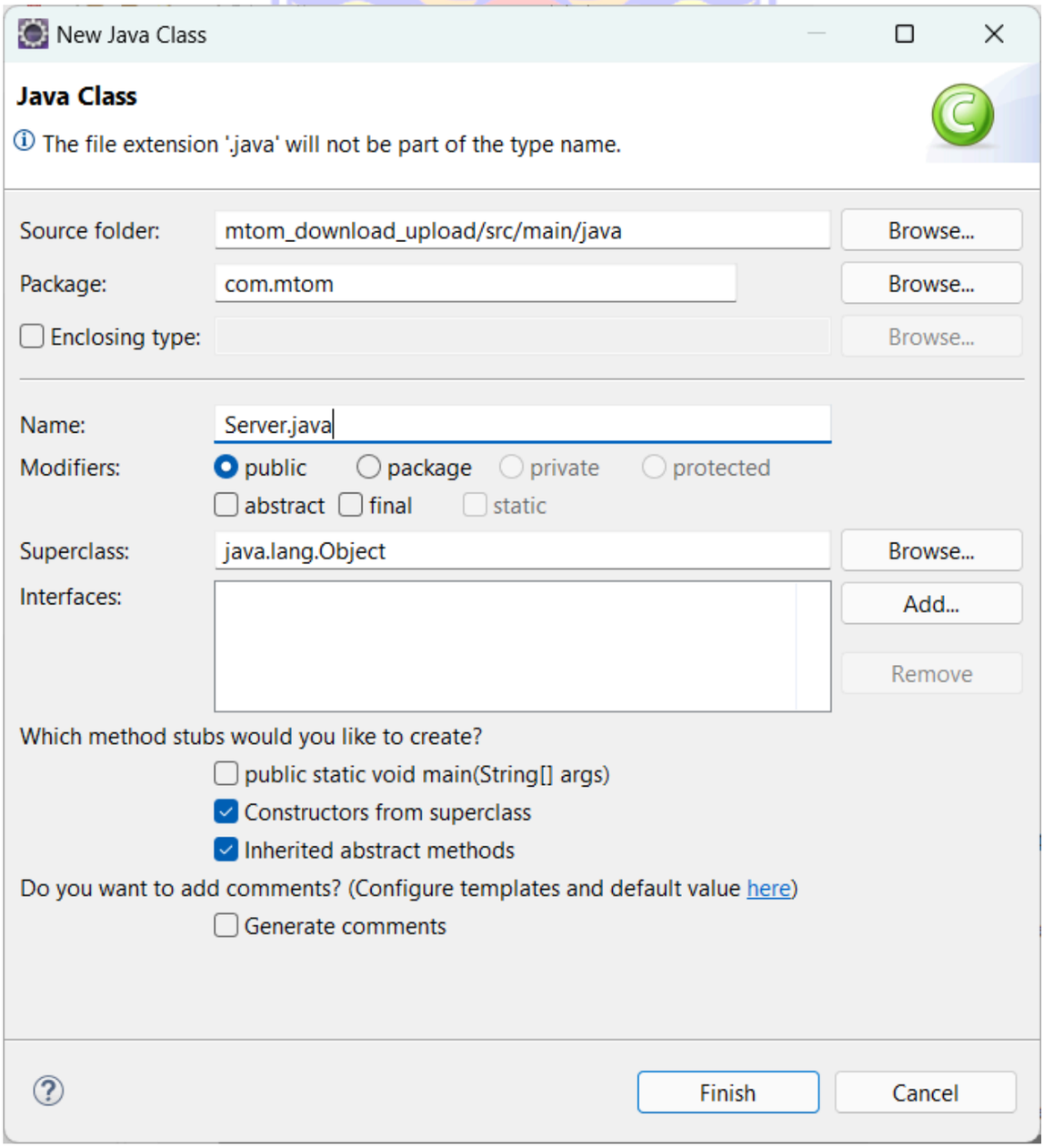


MTOM_intl.java:

```
package com.mtom;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;
import javax.jws.soap.SOAPBinding.Use;
import javax.xml.ws.soap.MTOM;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.file.Files;
@MTOM
@WebService(endpointInterface="com.mtom.MTOM_int")
@SOAPBinding(style = Style.DOCUMENT, use = Use.LITERAL)
public class MTOM_impl implements MTOM_int {

        private static final String MEDIA_DIR = "media/";

  static {
    new File(MEDIA_DIR).mkdirs();
  }
  @Override
  public byte[] downloadImage(String fileName) throws IOException {
    File file = new File(MEDIA_DIR + fileName);
    if (!file.exists()) {
      throw new IOException("File not found: " + fileName);
    }
    return Files.readAllBytes(file.toPath());
  }

  @Override
  public String uploadImage(byte[] imageData, String fileName) throws IOException {
    if (imageData == null || imageData.length == 0) {
      throw new IOException("No image data provided");
    }

    String filePath = MEDIA_DIR + fileName;
    try (FileOutputStream fos = new FileOutputStream(filePath)) {
      fos.write(imageData);
    }
    return "File uploaded successfully: " + fileName + " (" + imageData.length + " bytes)";
  }
}
```

7. Create a new class in com.mtom package and name it server.java:

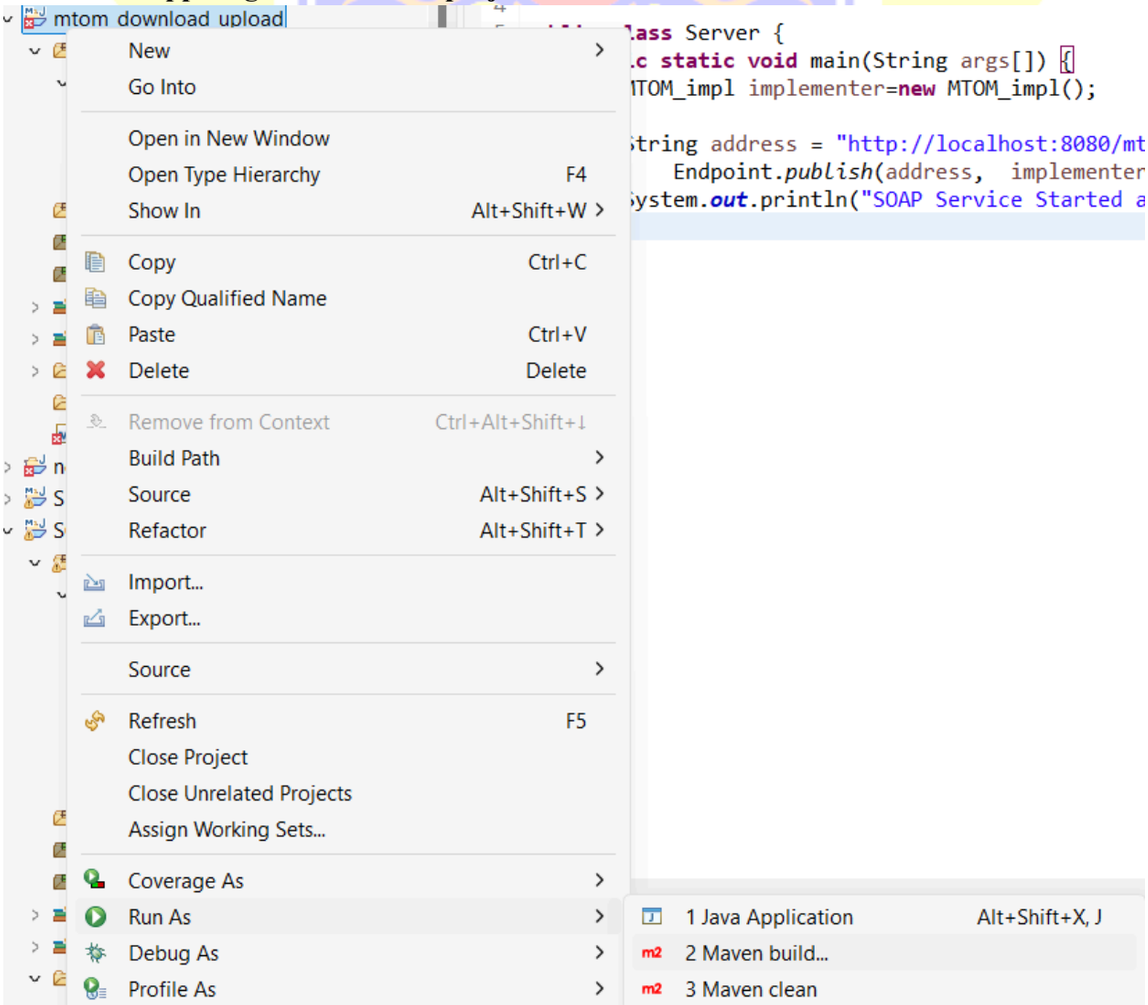




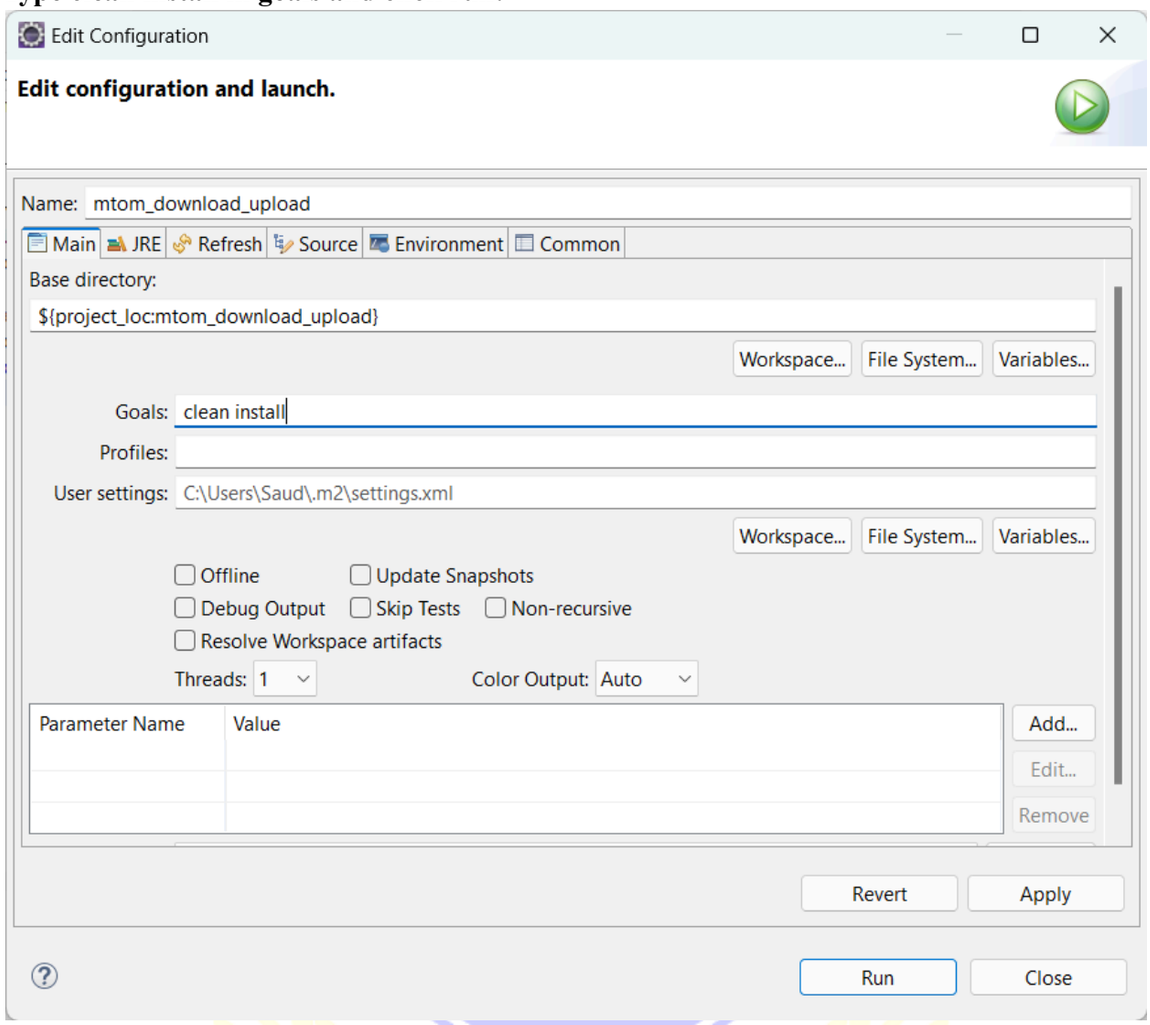
Server.java:

```
package com.mtom;
import javax.xml.ws.Endpoint;
public class Server {
	public static void main(String args[]) {
		MTOM_impl implementer=new MTOM_impl();

		String address = "http://localhost:8080/mtom";
			Endpoint.publish(address, implementer);
		System.out.println("SOAP Service Started at: " +address);
	}
}
```

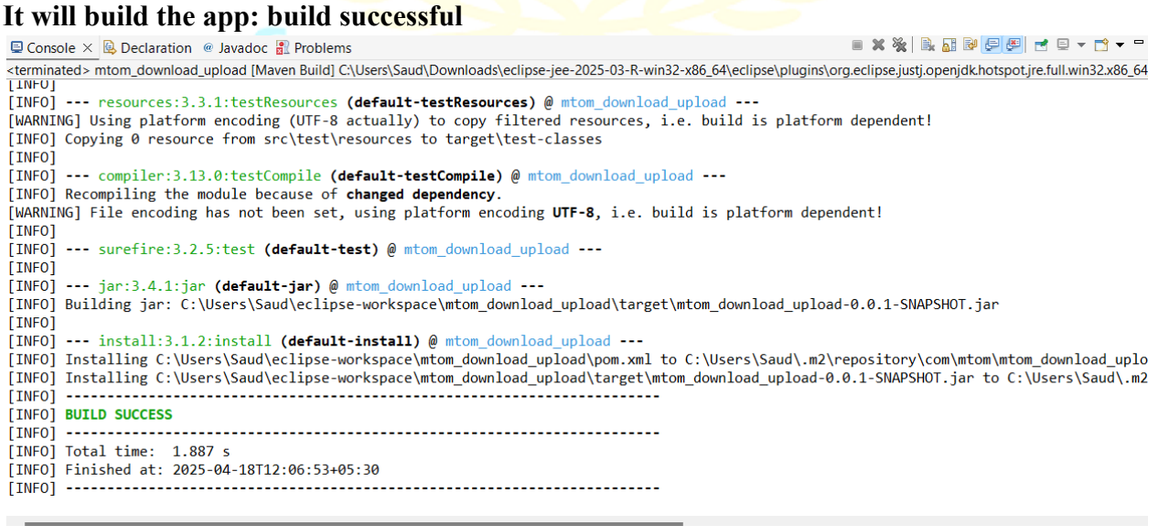
8. Build the app: Right-click on the project folder -> Run as -> Maven Build



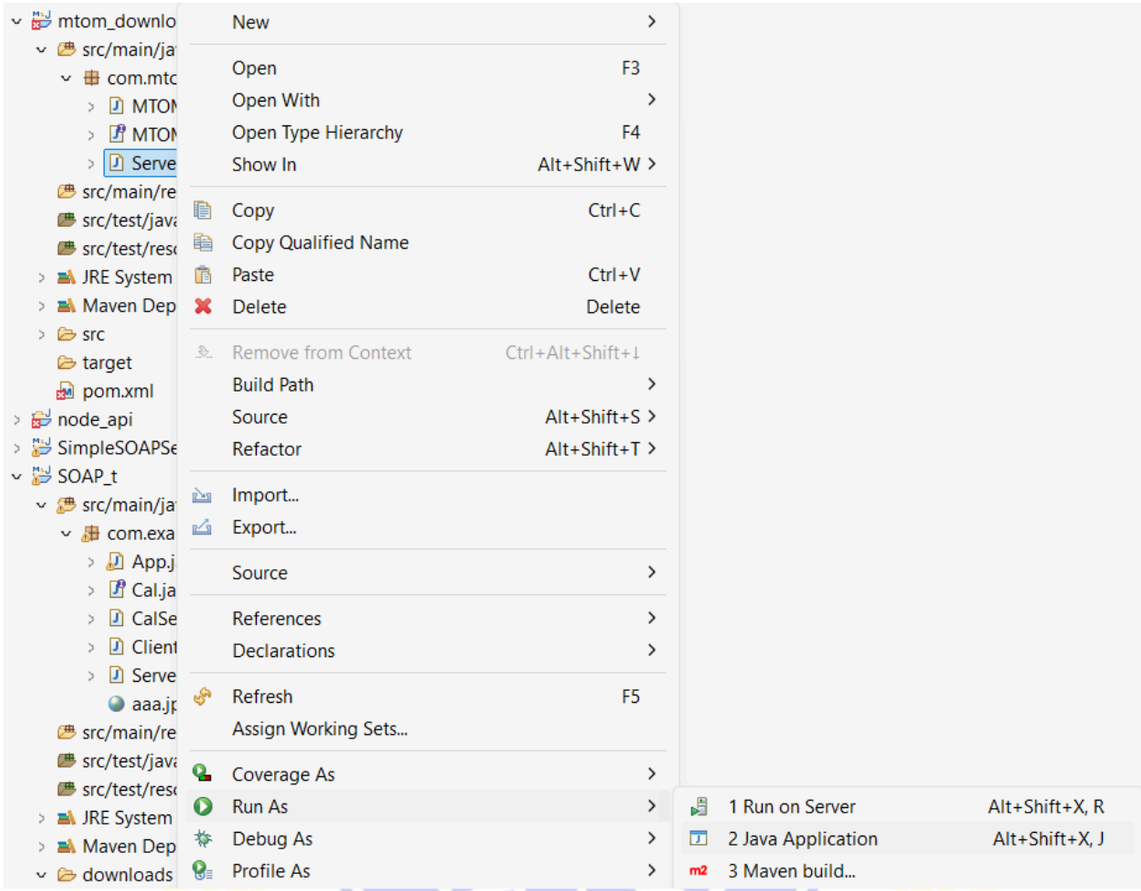
Type clean install in goals and click run:



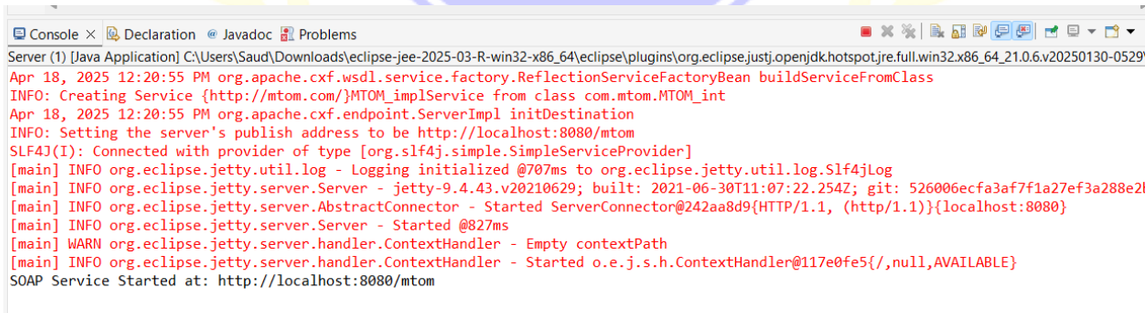
It will build the app: build successful



10. Run the server: Right click on Server.java -> Run as -> Java Application

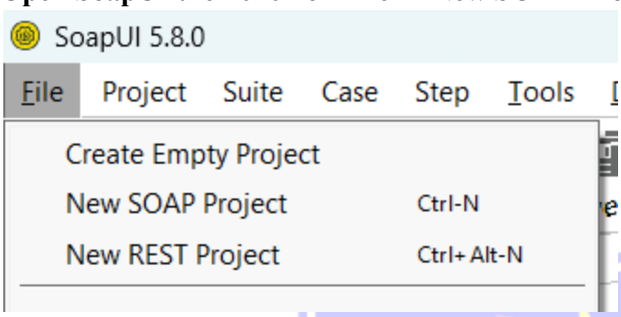


Server will start:

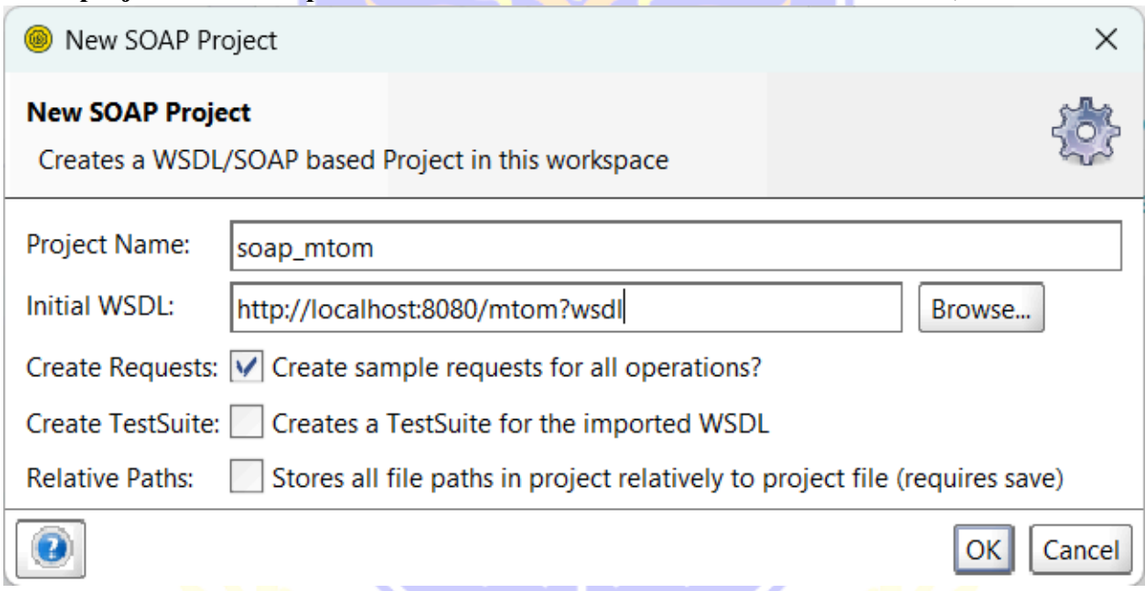


Copy the link: http://localhost:8080/mtom

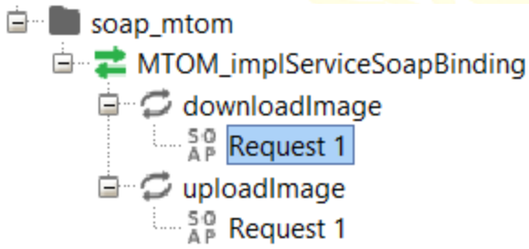
Open SoapUI then click on File -> New SOAP Project:



Write project name and paste the link in initial wsdl then add `?wsdl` in link, then click ok:

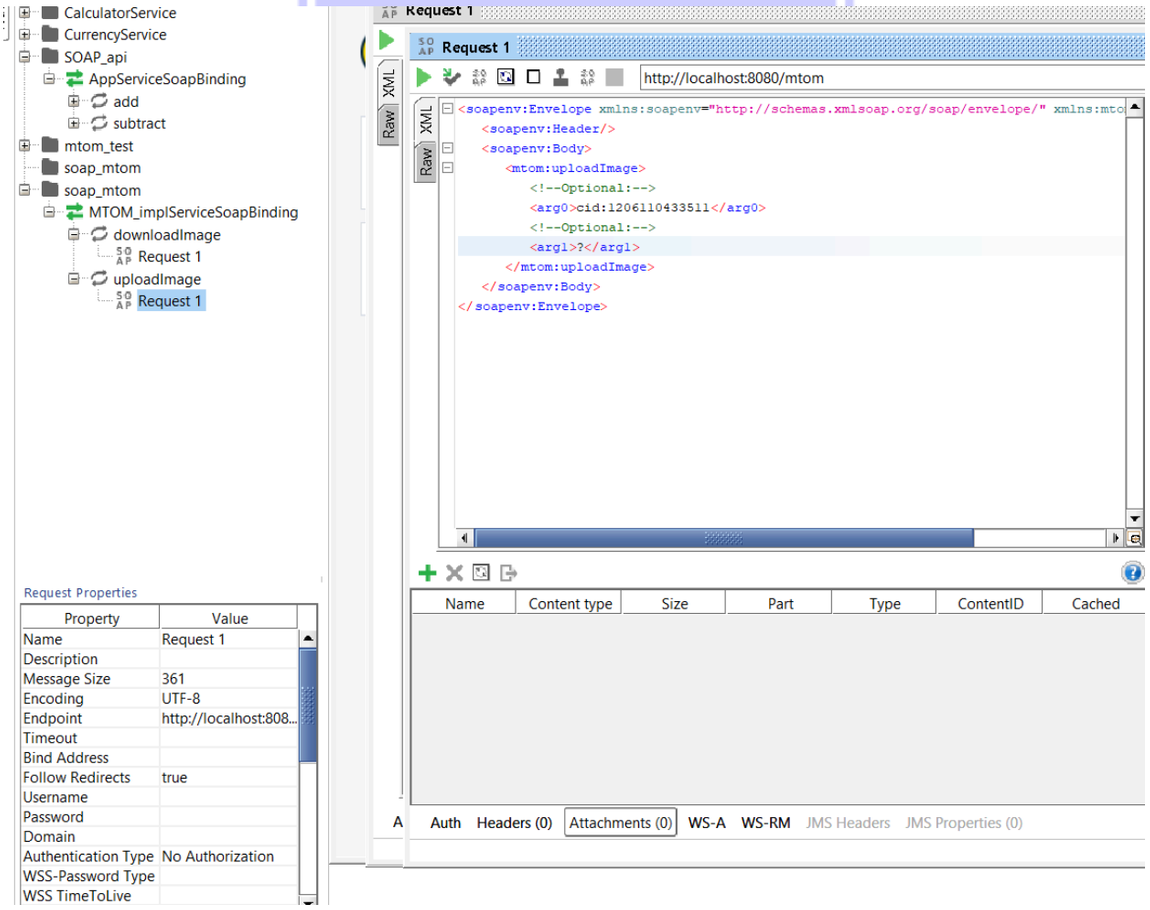


It will create the sample request for upload and download operations:

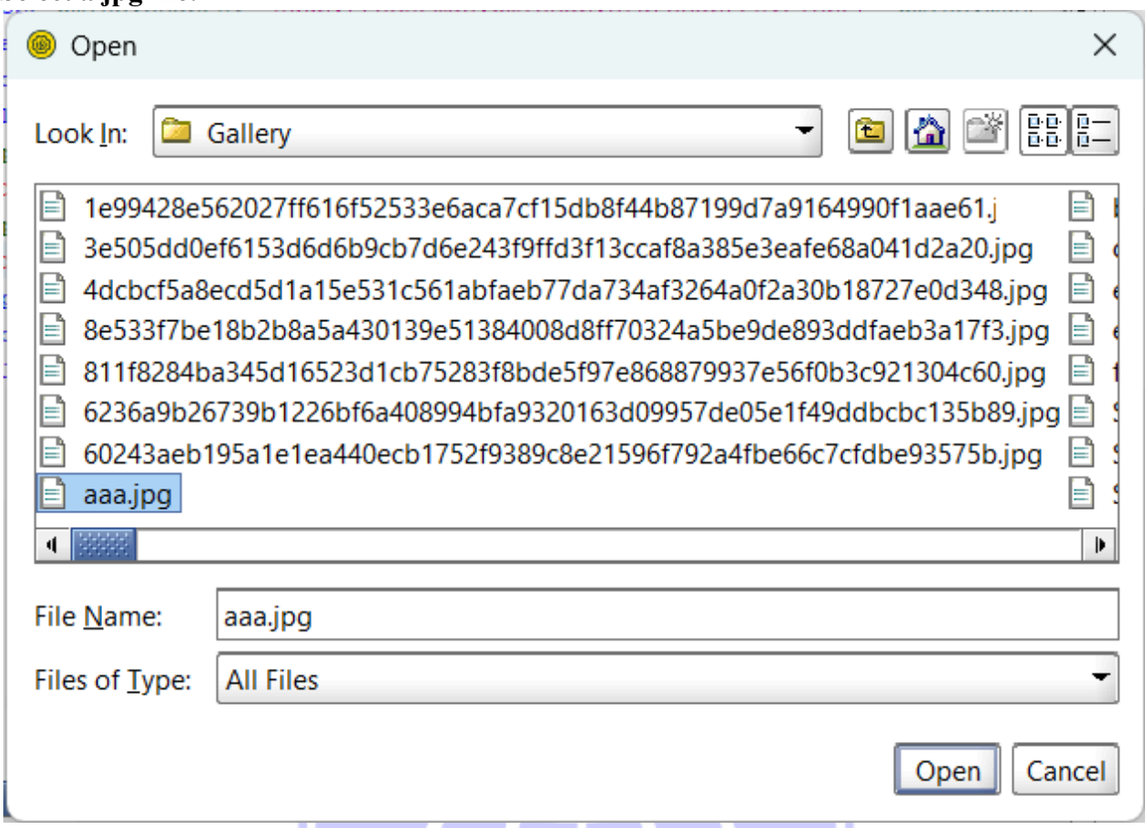


Project folder: See there is no media folder or image files

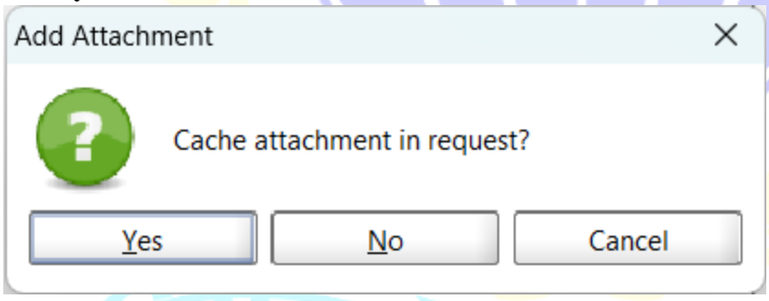
Click on `Request1` in `uploadImage` then request window will open then click on `Attachments` then click on `+` icon:



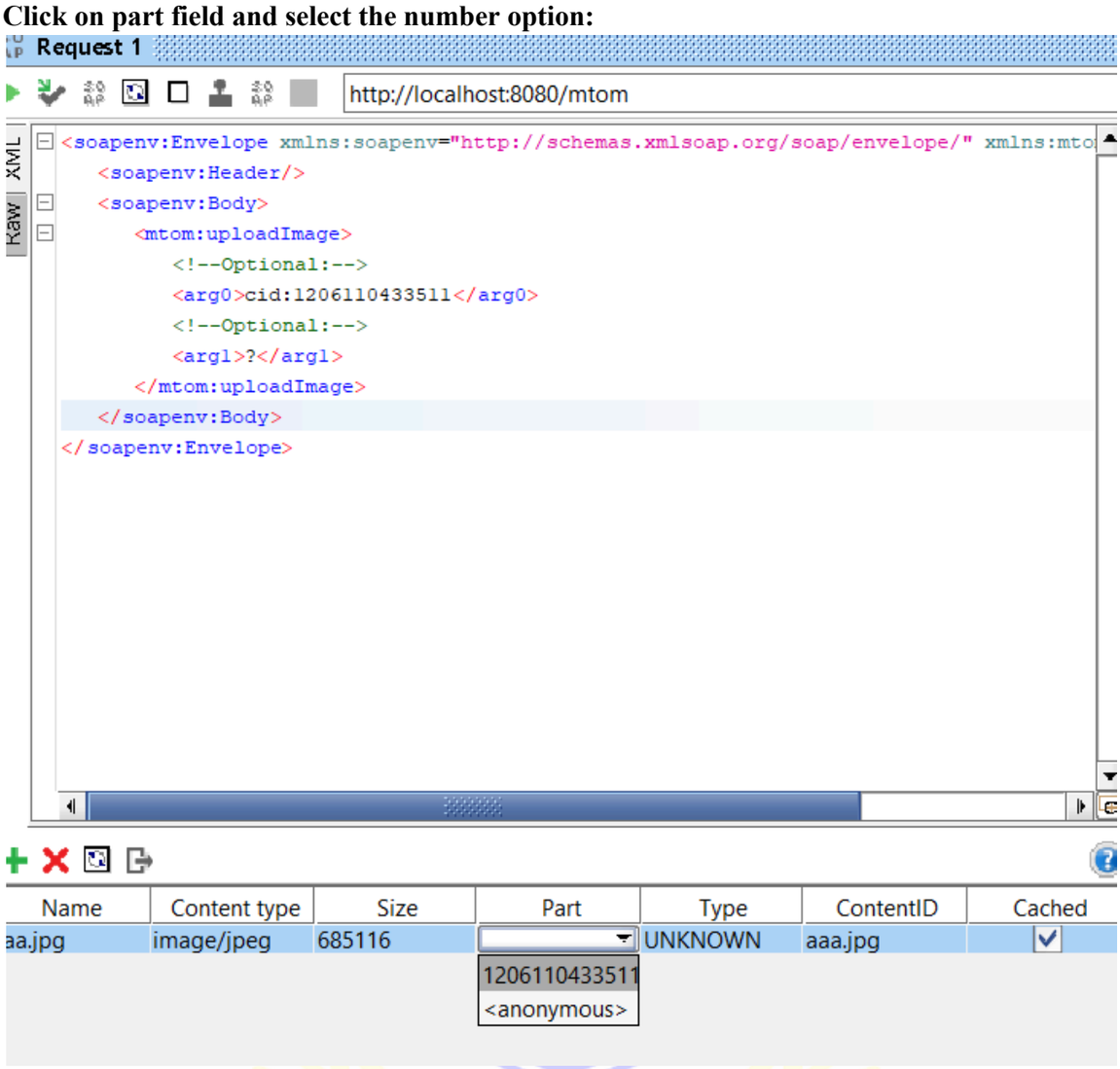
Select a jpg file:



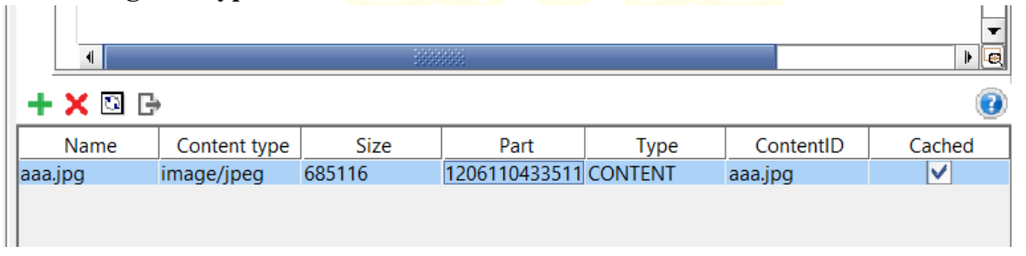
Click yes



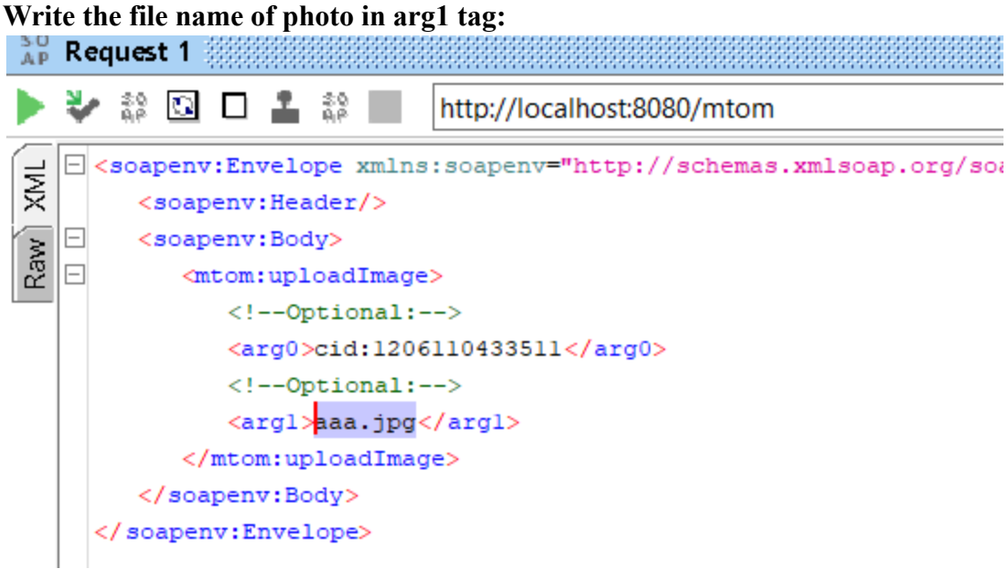
Click on part field and select the number option:



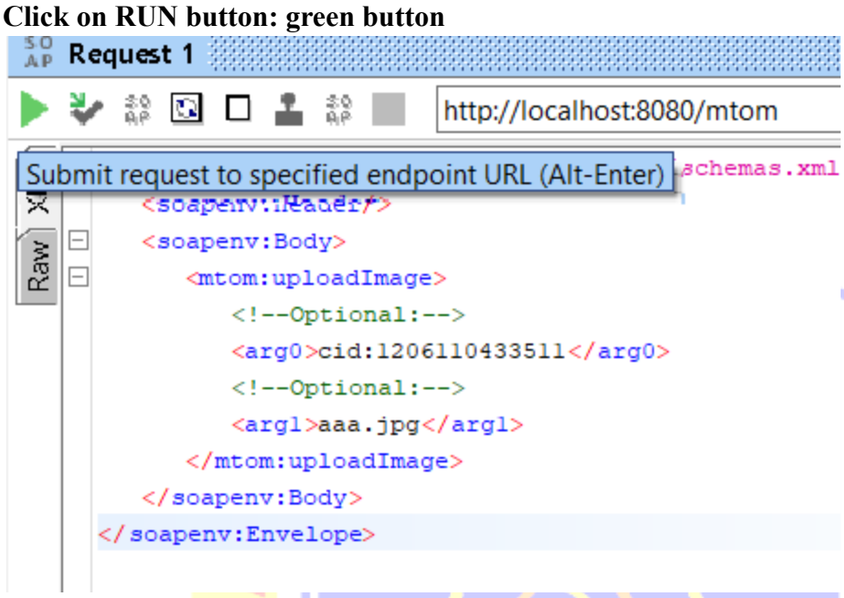
It will change the type:



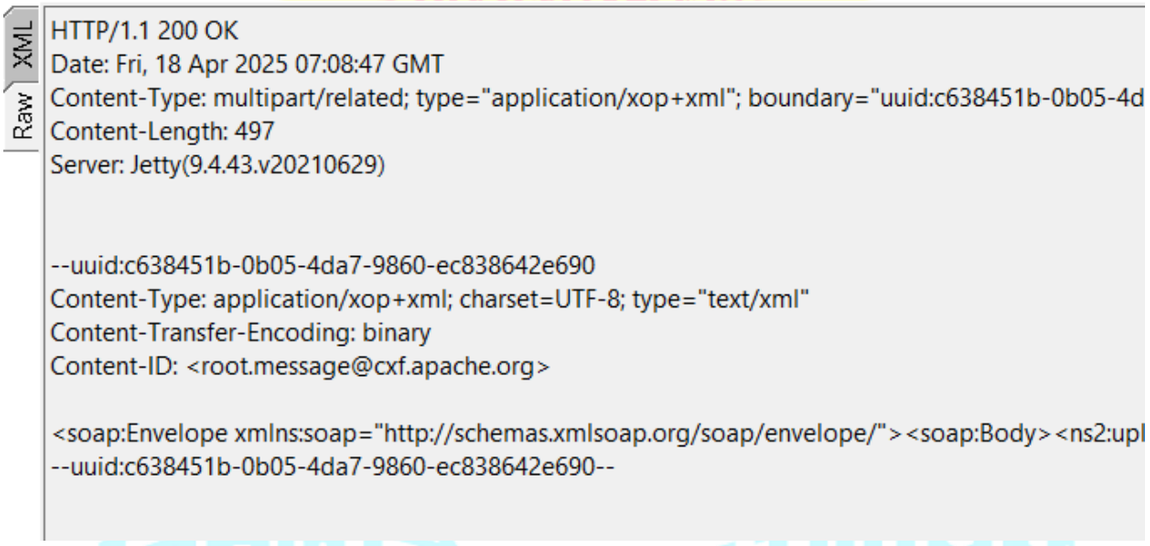
Write the file name of photo in arg1 tag:



Click on RUN button: green button

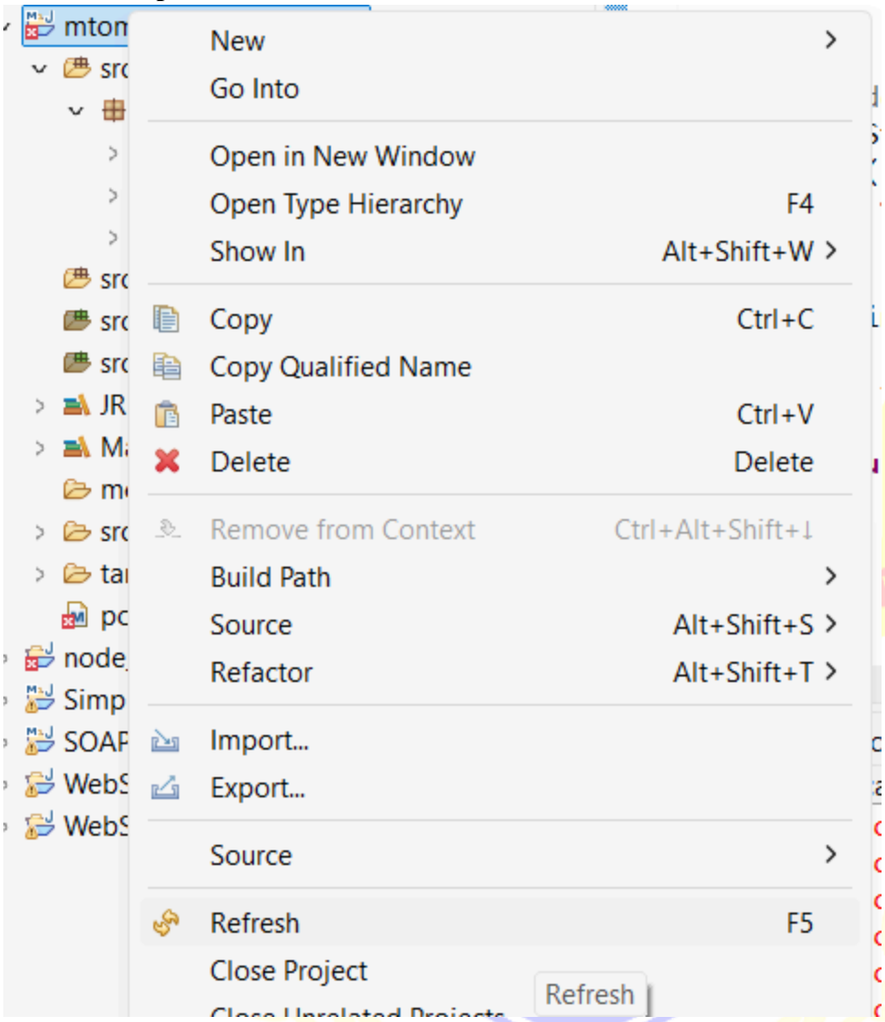


Raw Output:

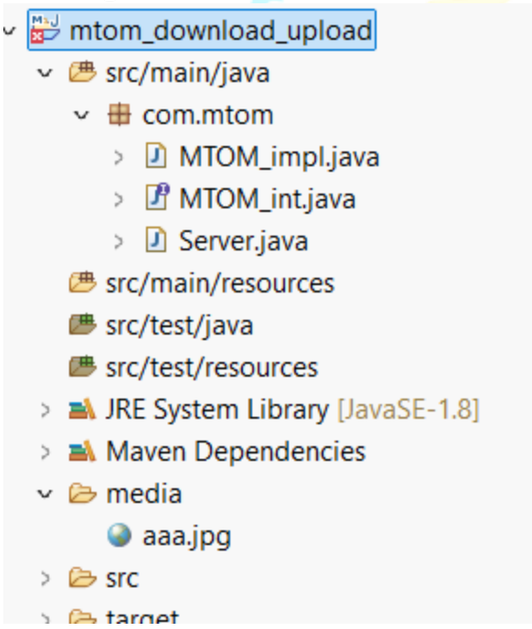


XML output:

Open Eclipse IDE then Refresh the project folder -> right click on the folder and click on the refresh option:

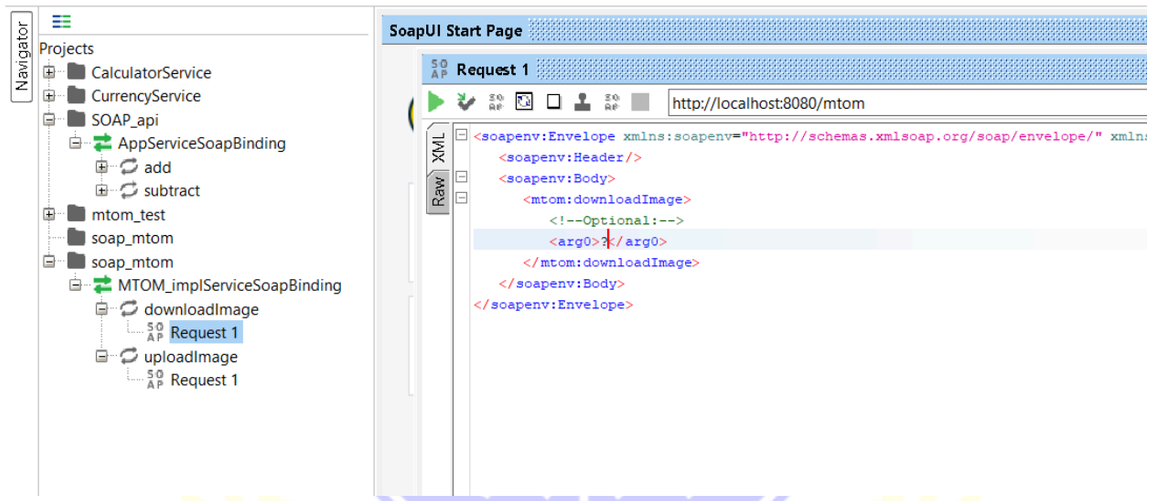


The image file will be available in the media folder:

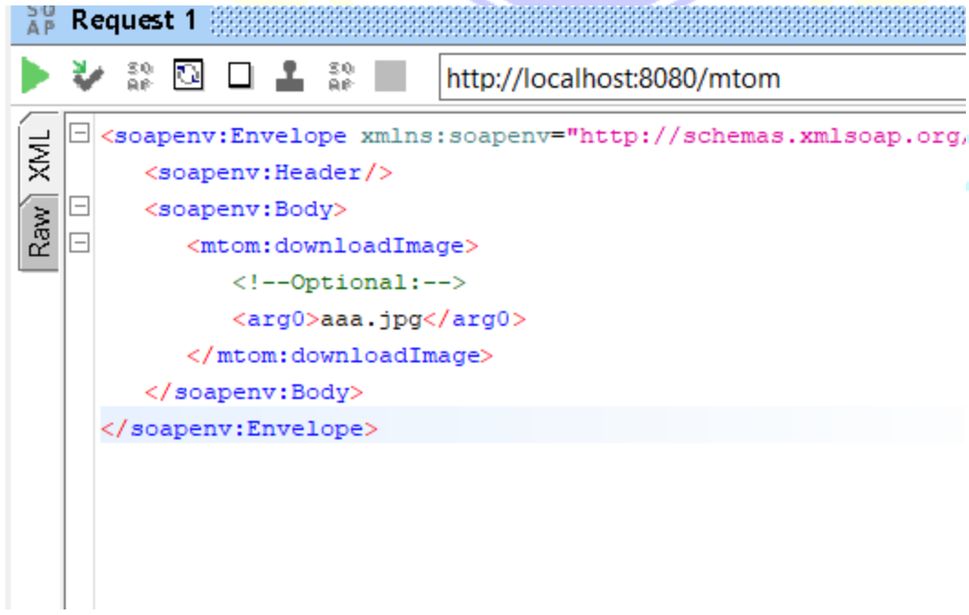


Click on the image file to see the image in editor:

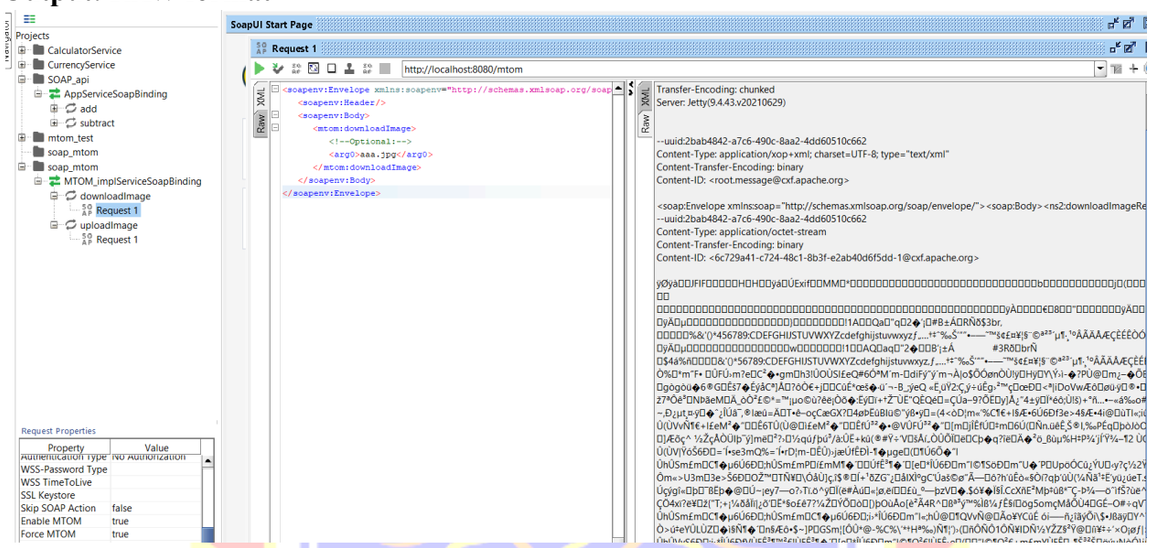
Open SOAPUI -> Click on the request1 in downloadImage:



Enter the file name to be downloaded and click on the run button:

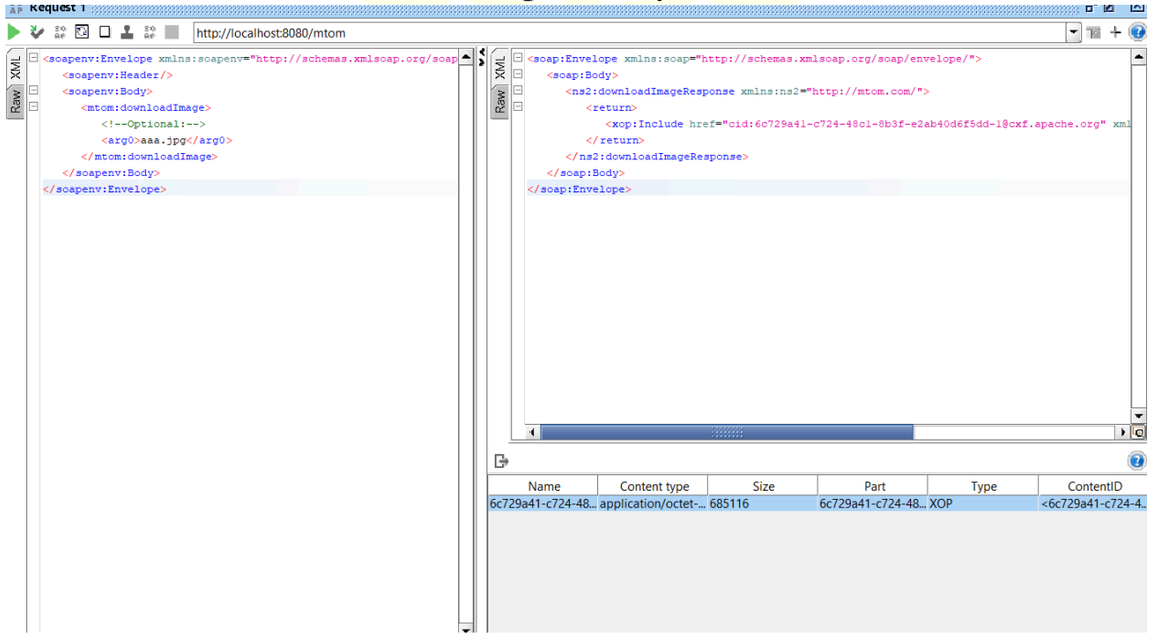


Output: RAW format

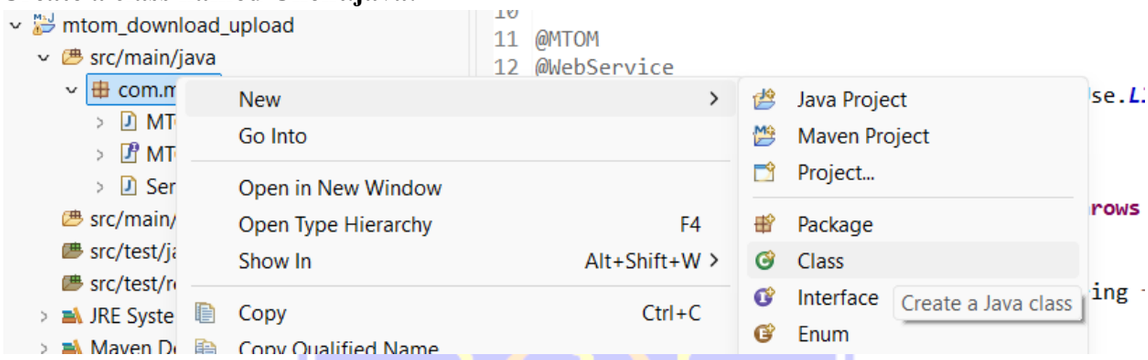


XML format:

Check the attachment, it contains the image in binary format:



Create a class named Client.java:



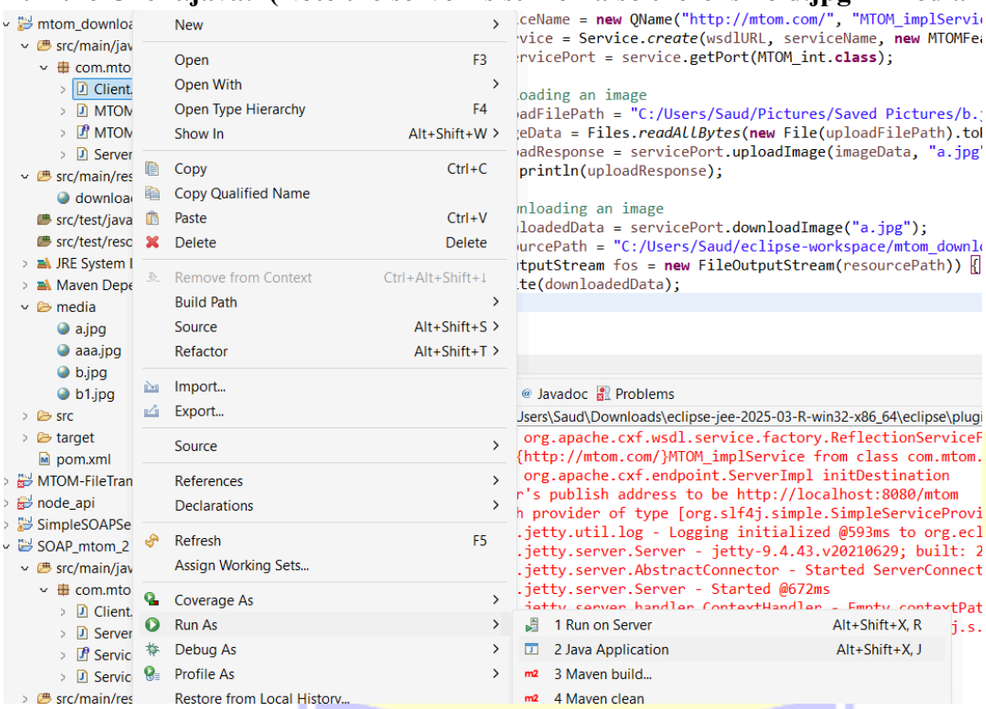
Client.java:

```
package com.mtom;
import java.io.File;
import java.io.FileOutputStream;
import java.nio.file.Files;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import javax.xml.ws.soap.MTOMFeature;
import java.net.URL;
import java.net.URI;
public class Client {
  public static void main(String[] args) throws Exception {
        URI uri = new URI("http://localhost:8080/mtom?wsdl");
    URL wsdlURL = uri.toURL();

    QName serviceName = new QName("http://mtom.com/", "MTOM_implService");
    Service service = Service.create(wsdlURL, serviceName, new MTOMFeature());
    MTOM_int servicePort = service.getPort(MTOM_int.class);

    // Test uploading an image
    String uploadFilePath = "C:/Users/Saud/Pictures/Saved Pictures/d.jpg"; // Specify the file to upload
    byte[] imageData = Files.readAllBytes(new File(uploadFilePath).toPath());
    String uploadResponse = servicePort.uploadImage(imageData, "d.jpg");
    System.out.println(uploadResponse);

//      Test downloading an image
    byte[] downloadedData = servicePort.downloadImage("d.jpg");
    String resourcePath =
"C:/Users/Saud/eclipse-workspace/mtom_download_upload/src/main/resources/downloaded_d.jpg";
    try (FileOutputStream fos = new FileOutputStream(resourcePath)) {
      fos.write(downloadedData);
    }
    System.out.println("File downloaded successfully: downloaded_image.jpg");
  }
}
```

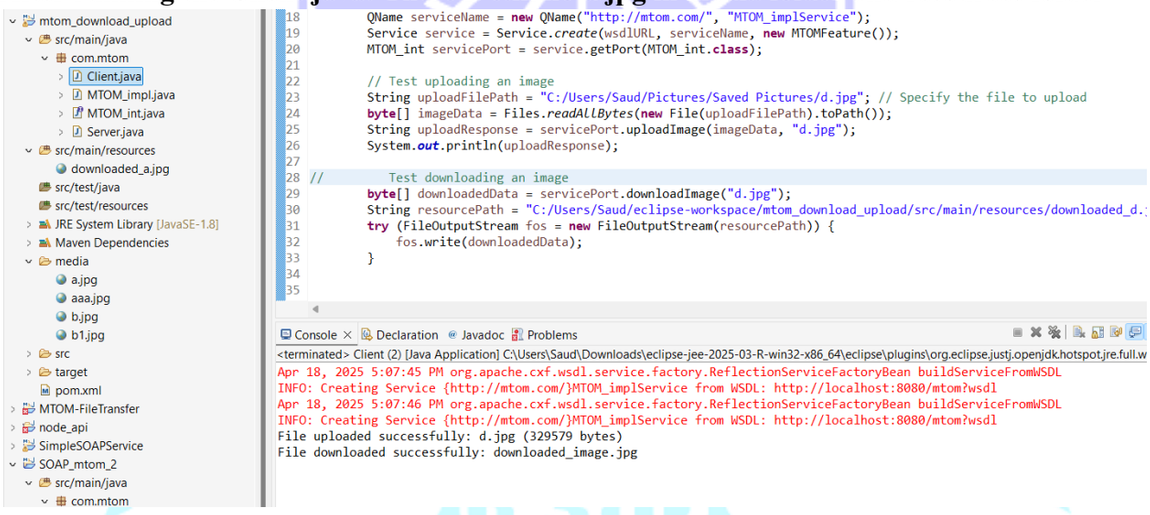
It will upload a file to media directory and download the file from media directory to the specified path.

Server is still on: if its off right-click on Server.java -> Run as -> Java Application

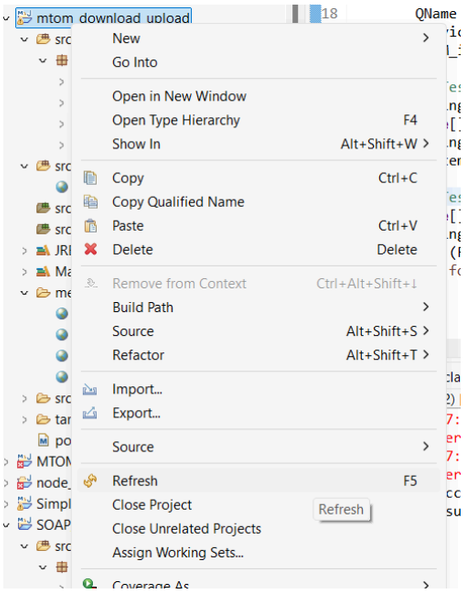
Run the Client.java: (Note the server is still on also there is no d.jpg in media folder)



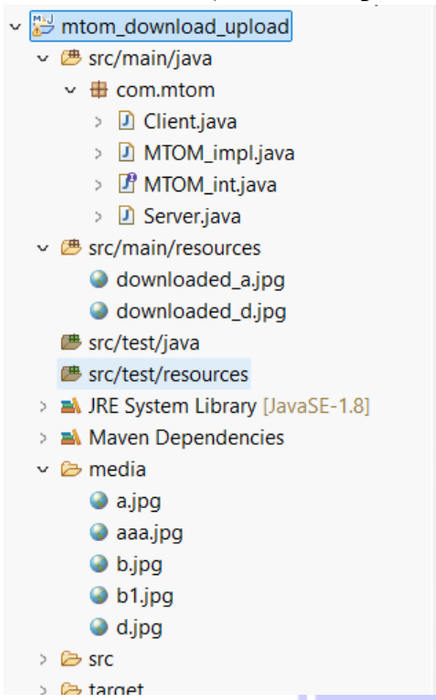
After running the Client.java: There is still no d.jpg in media folder and resources folder



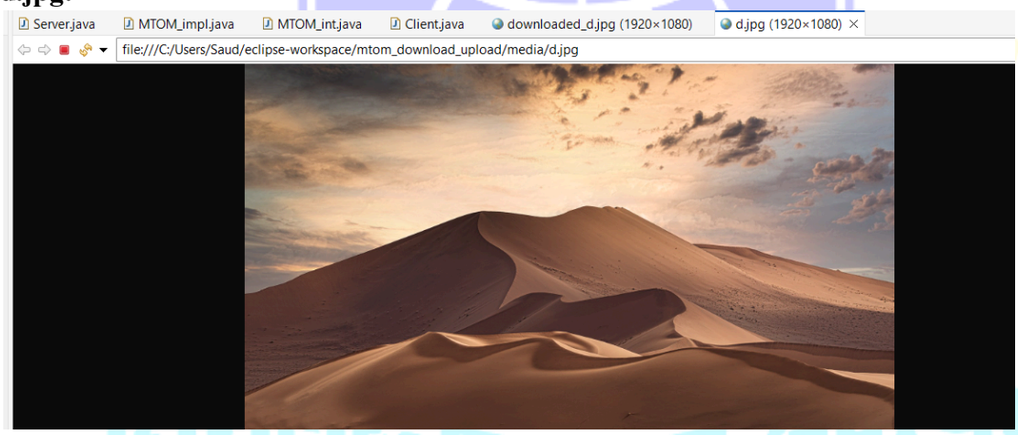
Refresh the folder:



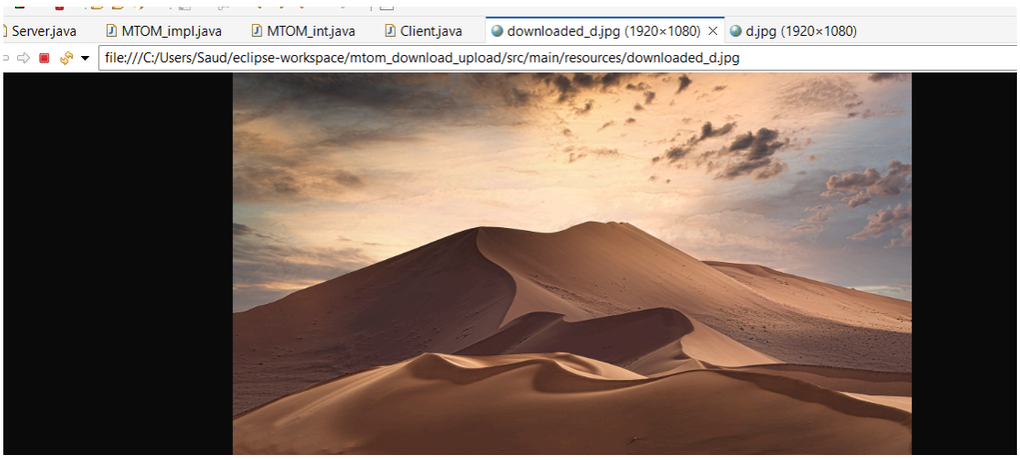
Output: d.jpg is there in media folder (upload operation) and downloaded_d.jpg in resources folder (download operation).



d.jpg:



downloaded_d.jpg:



Result and Discussion:
We have successfully implemented MTOM.

Learning Outcomes:
1. Successfully implemented MTOM
2. Understood MTOM

Course Outcomes:
1. We have learned about MTOM
2. We have learned how to upload and download media files using MTOM

Conclusion:
We have successfully implemented upload and download operations using MTOM.

Viva Question:
1. What is MTOM?
2. What is SOAP?
3. What is XML?
4. What is the use of MTOM6 ?

For Faculty Use

Correction Parameters	Formative Assessment [ ]	Timely completion practical [ ]	Attendance Learning Attitude[ ]

Theory-7
FOSS

FOSS Cloud stands for Free and Open Source Software Cloud, and it's a platform that provides virtualization and cloud services. Think of it as a tool that lets you create and manage virtual machines (computers that exist digitally) and other cloud-based resources without needing expensive proprietary software.

Intuition:
Imagine you have a computer that can create "virtual computers" inside it. These virtual computers can run different programs, store data, and even act like real computers—but everything happens digitally. FOSS Cloud is like a magic box that helps you do this easily and efficiently.

Uses:
- Virtualization: You can create virtual machines to test software, run applications, or simulate different environments.
- Cloud Services: It allows you to host services like websites, databases, or applications in the cloud.
- Cost Savings: Since it's free and open-source, you don't need to pay for expensive licenses.

Benefits:
1. Free to Use: No licensing fees—it's open-source.
2. Customizable: You can tweak it to fit your needs.
3. Community Support: A large community of developers and users can help you solve problems.
4. Transparency: You can see and modify the source code.
5. Security: Open-source software is often more secure because anyone can inspect and improve it.

What You Can Do with It:
- Host websites or applications.
- Create virtual machines for testing or development.
- Run cloud-based services for your business or personal projects.

Real-Life Examples:
- A small business might use FOSS Cloud to host its website and manage customer data.
- Developers can use it to test software in different environments without needing multiple physical computers.

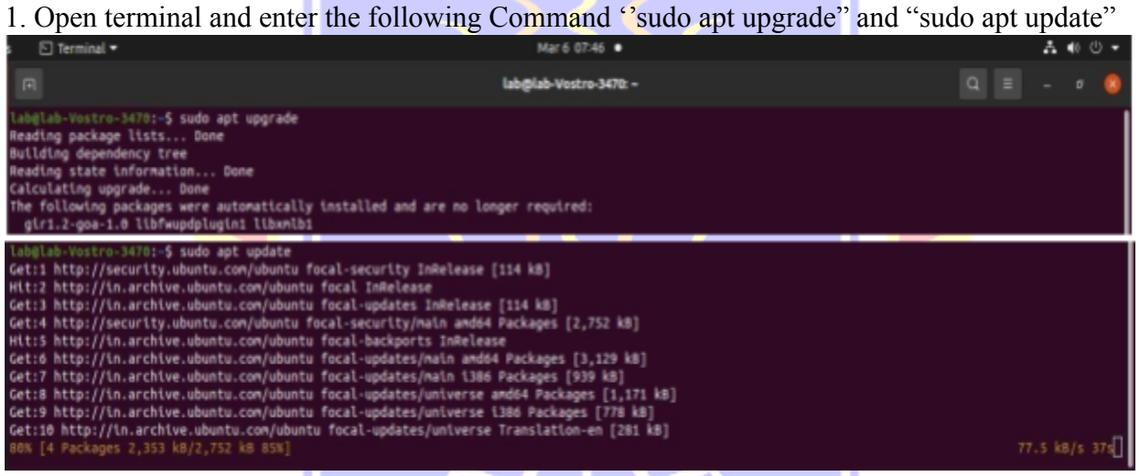
Alternatives:
If FOSS Cloud doesn't suit your needs, here are some alternatives:
- Proprietary Options: VMware, Microsoft Azure, Amazon Web Services (AWS).
- Other Open-Source Platforms: OpenStack, Proxmox, or Apache CloudStack.

Let me know if you'd like to explore any of these options further!

Theory-7 : FOSS
Practical-7: Implement FOSS-Cloud Functionality VSI (Virtual Server Infrastructure) Infrastructure as a Service (IaaS), Storage

Requiurements:
Memory: 4gb ram
Storage: 130gb
CPU with virtualization support

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

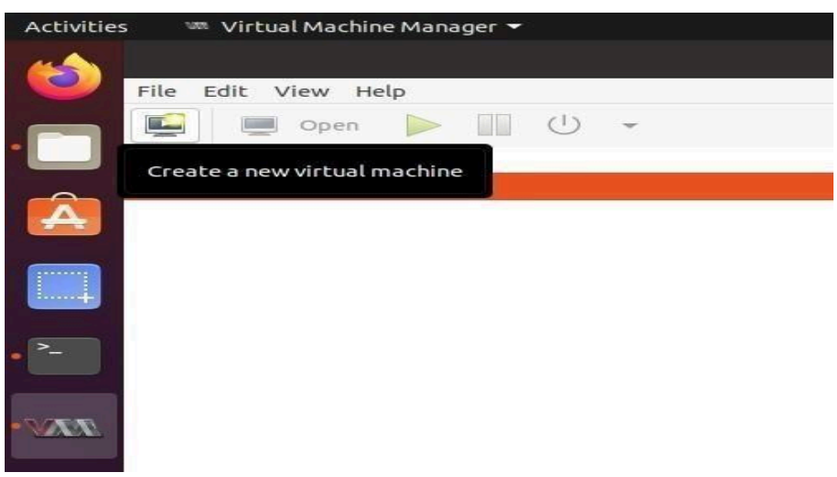


2. Enter the following command to install virt manager

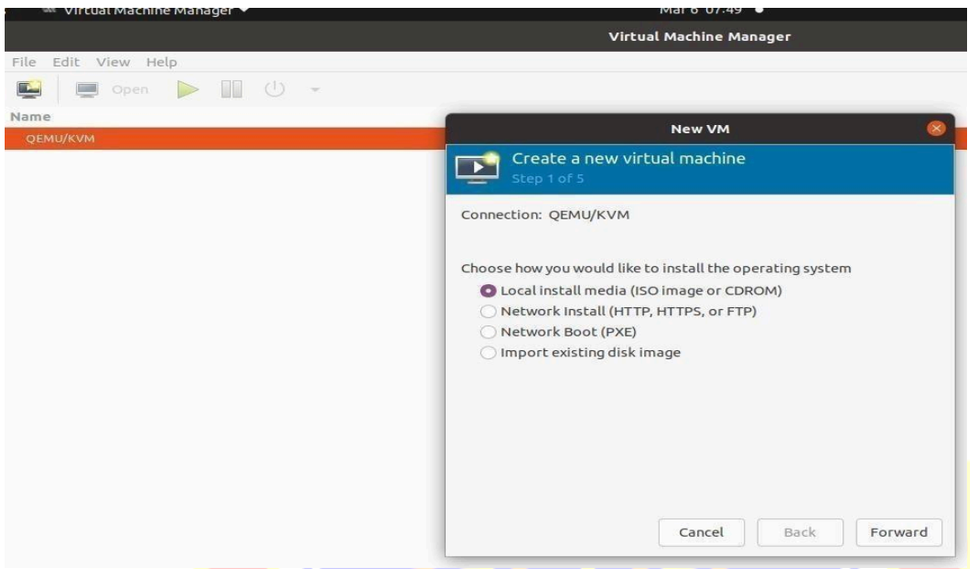
```
lab@lab-Vostro-3470:/etc$ sudo apt install virt-manager
[sudo] password for lab:
Reading package lists... Done
Building dependency tree
Reading state information... Done
virt-manager is already the newest version (1:2.2.1-3ubuntu2.2).
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 libfwupdplugin1 libxmlb1 linux-headers-5.13.0-30-generic
  linux-hwe-5.13-headers-5.13.0-30 linux-image-5.13.0-30-generic
  linux-modules-5.13.0-30-generic linux-modules-extra-5.13.0-30-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

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

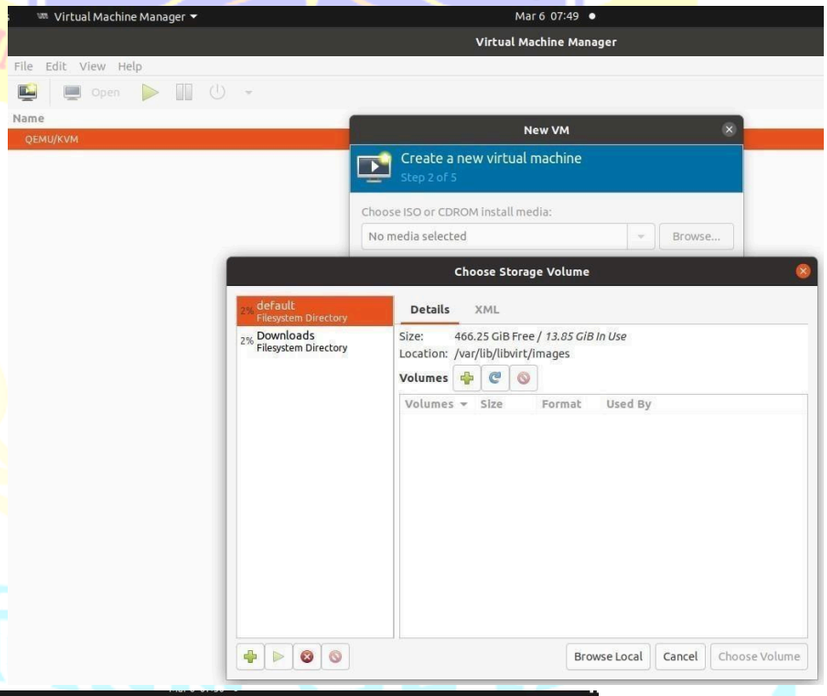
4. Once the virt manager is launched click on file➔ new virtual machine

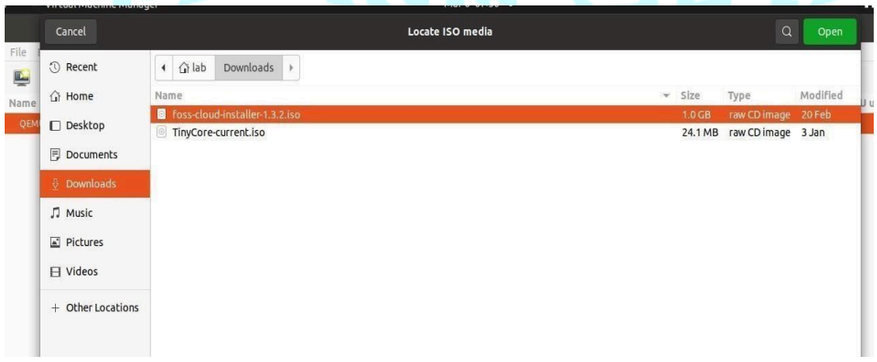


Select local install media (ISO image or CDROM) and click Forward

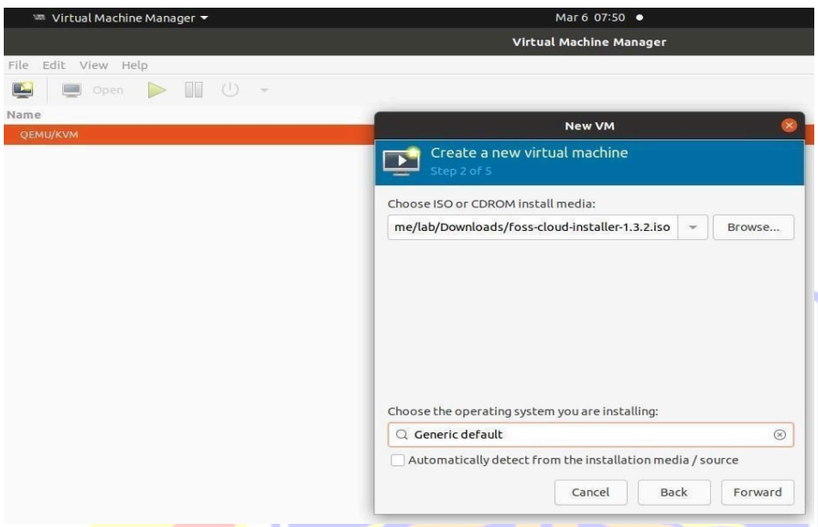


Click on Browse➔ Browse local ➔ select downloaded iso file ➔ Open

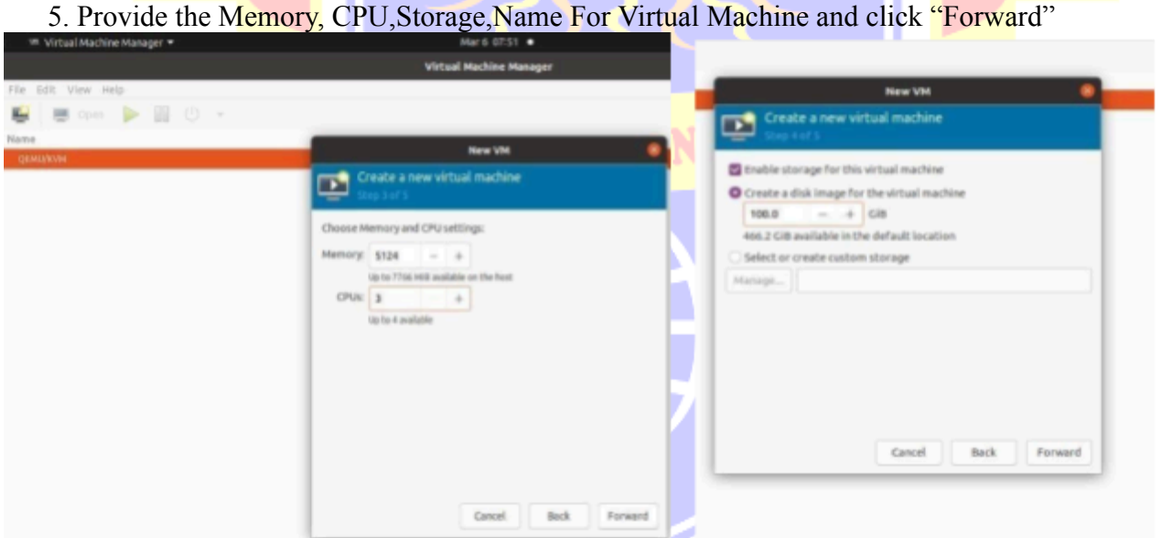


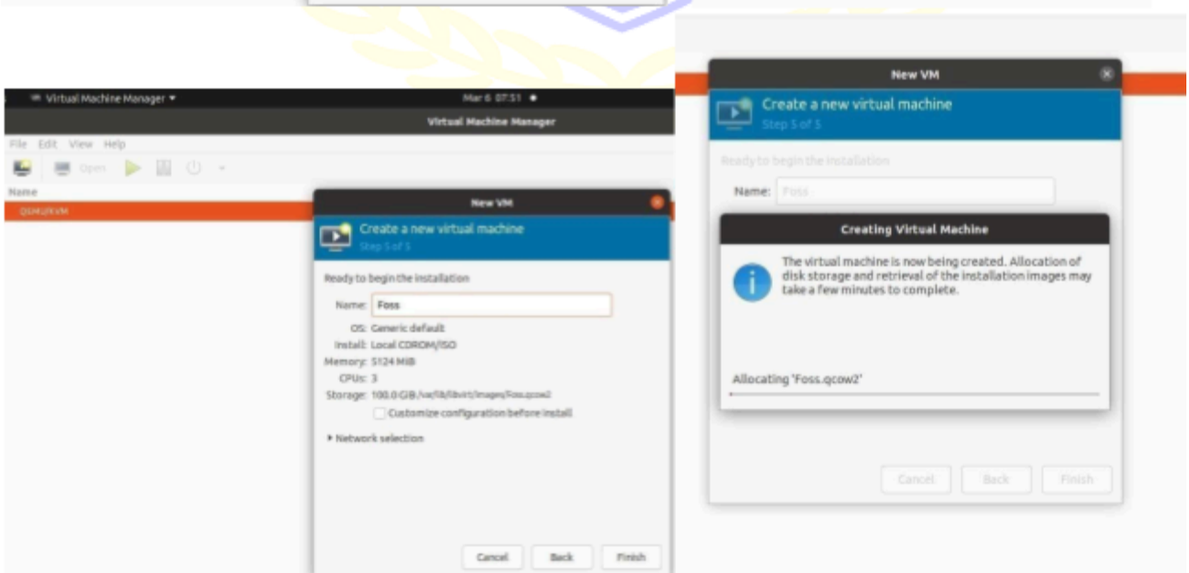


Uncheck The checkbox and Choose the OS you are installing as “Generic default” and click“Forward”

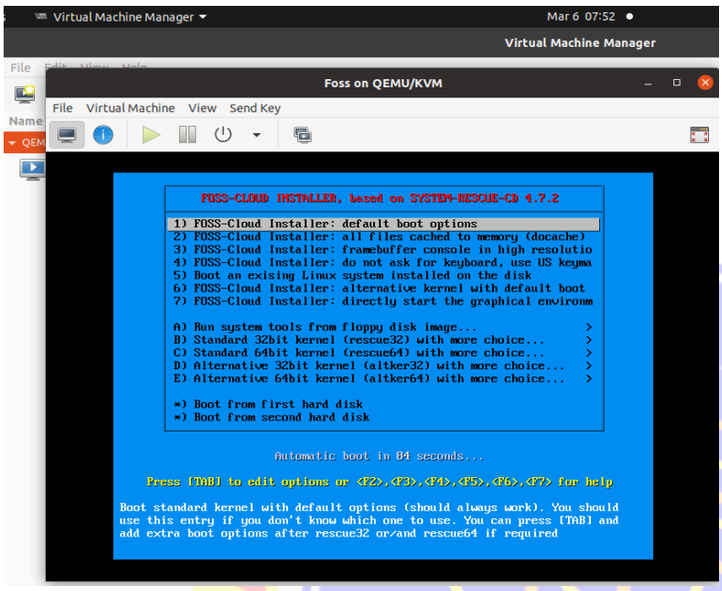


5. Provide the Memory, CPU,Storage,Name For Virtual Machine and click “Forward”



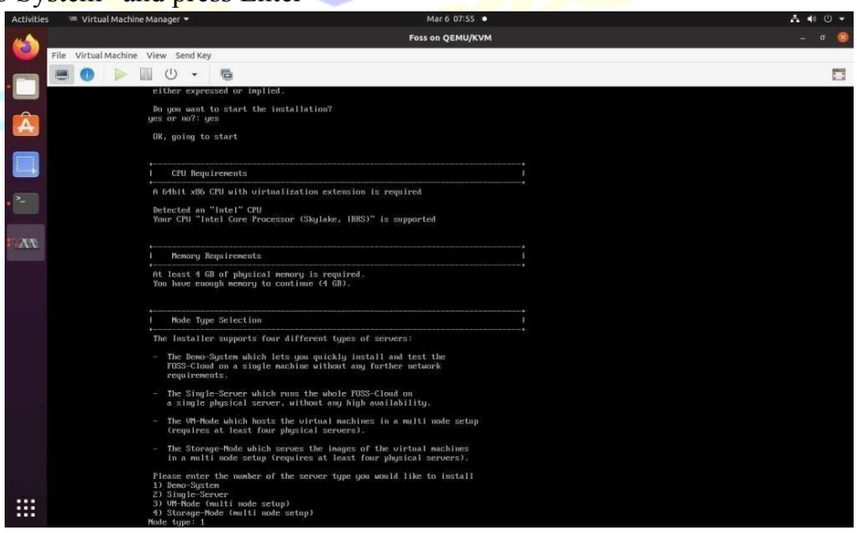


6. Once the foss Cloud is launched select “Foss-Cloud installer :default boot options” and press Enter



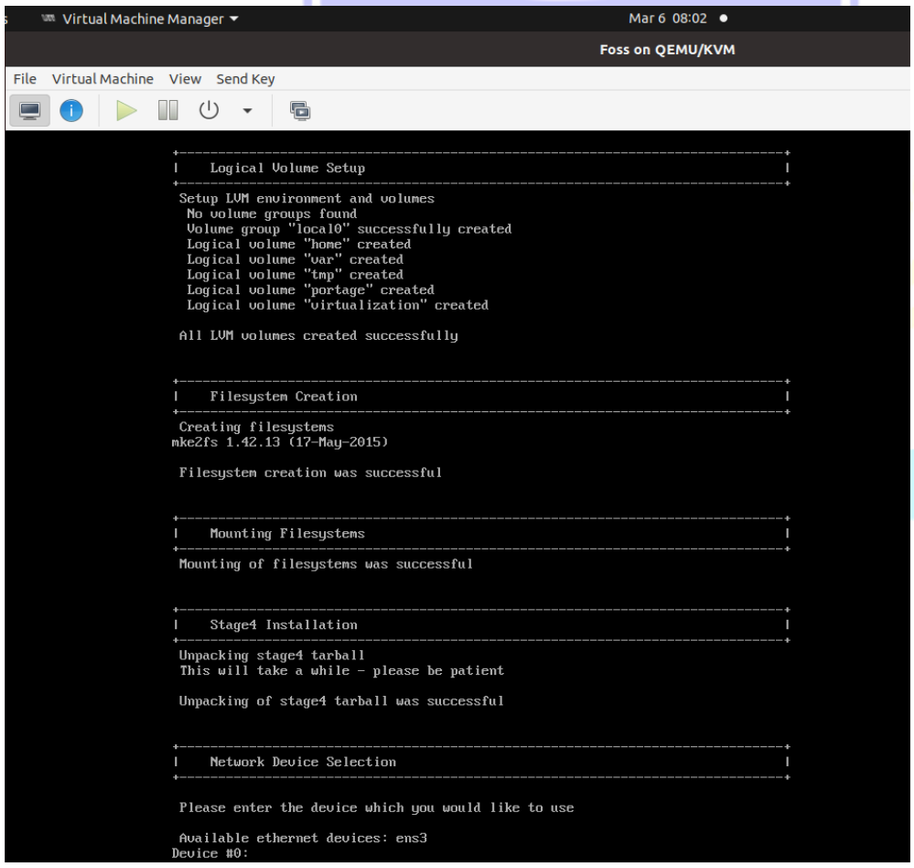
Types yes to start the installation

1. Select “Demo System” and press Enter



Enter the Device name as “sda” and press enter

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

 Found sda (150 GB). Size is OK

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

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


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


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



Result and Discussion:
We have successfully implemented FOSS cloud functionality

Learning Outcomes:
1. Successfully implemented FOSS cloud functionality
2. Understood the FOSS platform.

Course Outcomes:
1. We have learned about FOSS
2. We have understood FOSS.

Conclusion:
We have successfully implemented the FOSS cloud functionality.

Viva Question:
1. What is FOSS?
2. What is hypervisor?
3. What is virtualization?
4. What is IaaS?

For Faculty Use

Correction Parameters	Formative Assessment [ ]	Timely completion practical [ ]	Attendance Learning Attitude[ ]

Theory-9

AWS Flow Framework

Amazon Web Services (AWS) is a cloud computing platform that provides a wide range of services, including computing power, storage, databases, machine learning, and more. It allows businesses and developers to build, deploy, and scale applications without managing physical infrastructure.

The **AWS Flow Framework** is a Java-based library that simplifies the development of distributed, asynchronous applications using Amazon Simple Workflow Service (SWF). It abstracts the complexities of managing workflows, activities, and coordination logic.

Here's how you can develop a simple workflow using the AWS Flow Framework to print "Hello World" to the console:

Contracts : Contracts are Java interfaces that define the workflow and activity methods.

Activities: Activities are the tasks executed as part of the workflow.

Workflow: The workflow defines how activities are executed and coordinated.

Worker: Workers host the workflow and activity implementations and poll Amazon SWF for tasks

Starter: A workflow starter initiates the workflow execution.

Steps:
Defining Contracts: Create interfaces for activities and workflows.
Implementing Activities: Write the logic for individual tasks.
Workflow Coordination Logic: Define the sequence and coordination of activities.
Worker Programs: Host the workflow and activity implementations.
Workflow Starter: Initiate the workflow execution.

Practical-9: Using AWS Flow Framework develop application that includes a simple workflow. Workflow calls an activity to print hello world to the console. It must define the basic usage of AWS Flow Framework, including defining contracts, implementation of activities and workflow coordination logic and worker programs to host them

Steps:

```
art@art-VMware-Virtual-Platform:~$ sudo apt-get upgrade && apt-get update
[sudo] password for art:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
```

```
art@art-VMware-Virtual-Platform:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 226 kB of archives.
After this operation, 534 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu noble-updates/main amd64 curl amd64 8.
5.0-2ubuntu10.6 [226 kB]
Fetched 226 kB in 1s (161 kB/s)
Selecting previously unselected package curl.
(Reading database ... 153600 files and directories currently installed.)
Preparing to unpack .../curl_8.5.0-2ubuntu10.6_amd64.deb ...
Unpacking curl (8.5.0-2ubuntu10.6) ...
Setting up curl (8.5.0-2ubuntu10.6) ...
Processing triggers for man-db (2.12.0-4build2) ...
```

Install Docker:

sudo apt-get install ca-certificates curl

```
art@art-VMware-Virtual-Platform:~/sam-test-app$ sudo apt-get install ca-certific
ates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
art@art-VMware-Virtual-Platform:~/sam-test-app$
```

sudo install -m 0755 -d /etc/apt/keyrings

```
art@art-VMware-Virtual-Platform:~/sam-test-app$ sudo install -m 0755 -d /etc/apt
/keyrings
art@art-VMware-Virtual-Platform:~/sam-test-app$
```

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

```
art@art-VMware-Virtual-Platform:~/sam-test-app$ sudo curl -fsSL https://download
.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

sudo chmod a+r /etc/apt/keyrings/docker.asc

```
art@art-VMware-Virtual-Platform:~/sam-test-app$ sudo chmod a+r /etc/apt/keyrings
/docker.asc
art@art-VMware-Virtual-Platform:~/sam-test-app$
```

**echo **
**"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu **
$(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```
art@art-VMware-Virtual-Platform:~/sam-test-app$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc
] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" |
 \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
art@art-VMware-Virtual-Platform:~/sam-test-app$
```

sudo apt-get update

```
art@art-VMware-Virtual-Platform:~/sam-test-app$ sudo apt-get update
Get:1 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:2 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [24.0
 kB]
Hit:3 http://in.archive.ubuntu.com/ubuntu noble InRelease
Get:4 http://in.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 http://in.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:7 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages
[1,056 kB]
Fetched 1,255 kB in 2s (727 kB/s)
Reading package lists... Done
art@art-VMware-Virtual-Platform:~/sam-test-app$
```

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

```
art@art-VMware-Virtual-Platform:~/sam-test-app$ sudo apt-get install docker-ce d
ocker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras pigz slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli
  docker-ce-rootless-extras docker-compose-plugin pigz slirp4netns
0 upgraded, 8 newly installed, 0 to remove and 1 not upgraded.
Need to get 120 MB of archives.
After this operation, 439 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

```
art@art-VMware-Virtual-Platform:~/sam-test-app$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:c41088499908a59aae84b0a49c70e86f4731e588a737f1637e73c8c09d995654
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash
```

sudo usermod -aG docker $USER

Install AWS-CLI:

```
art@art-VMware-Virtual-Platform:~$ curl "https://awscli.amazonaws.com/awscli-exe
-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 67.1M  100 67.1M    0     0  12.0M      0  0:00:05  0:00:05 --:--:-- 12.1M
```

curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip.sig

```
art@art-VMware-Virtual-Platform:~$ curl -o awscliv2.sig https://awscli.amazonaw
.com/awscli-exe-linux-x86_64.zip.sig
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   566  100   566    0     0    758      0 --:--:-- --:--:-- --:--:--   757
art@art-VMware-Virtual-Platform:~$
```

unzip awscliv2.zip

```
 inflating: aws/dist/awscli/examples/ec2/delete-ipam-pool.rst
 inflating: aws/dist/awscli/examples/ec2/modify-verified-access-trust-prov
st
 inflating: aws/dist/awscli/examples/ec2/delete-transit-gateway-peering-at
nt.rst
 inflating: aws/dist/awscli/examples/ec2/describe-spot-fleet-requests.rst
 inflating: aws/dist/awscli/examples/ec2/disable-image.rst
 inflating: aws/dist/awscli/examples/ec2/describe-verified-access-instance
ng-configurations.rst
 inflating: aws/dist/awscli/examples/ec2/create-local-gateway-route-table.
 inflating: aws/dist/awscli/examples/ec2/restore-snapshot-from-recycle-bin

 inflating: aws/dist/awscli/examples/ec2/delete-vpc.rst
 inflating: aws/dist/awscli/examples/ec2/lock-snapshot.rst
 inflating: aws/dist/awscli/examples/ec2/describe-instance-connect-endpoin
```

sudo ./aws/install

```
art@art-VMware-Virtual-Platform:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

sudo apt install git

```
art@art-VMware-Virtual-Platform:~$ sudo apt install git
[sudo] password for art:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk git
  git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 1 not upgraded.
Need to get 4,804 kB of archives.
After this operation, 24.5 MB of additional disk space will be used.
```

sudo apt install python3-pip

```
art@art-VMware-Virtual-Platform:~$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2
  dpkg-dev fakeroot g++ g++-13 g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu
  gcc gcc-13 gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu javascript-common
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
  libasan8 libbinutils libcc1-0 libctf-nobfd0 libctf0 libdpkg-perl
  libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-13-dev libgprofng0
  libhwasan0 libitm1 libjs-jquery libjs-sphinxdoc libjs-underscore liblsan0
  libpython3-dev libpython3.12-dev libquadmath0 libsframe1 libstdc++-13-dev
  libtsan2 libubsan1 lto-disabled-list make python3-dev python3-setuptools
  python3-wheel python3.12-dev zlib1g-dev
Suggested packages:
  binutils-doc gprofng-gui bzip2-doc debian-keyring g++-multilib
  g++-13-multilib gcc-13-doc gcc-multilib autoconf automake libtool flex bison
  gcc-doc gcc-13-multilib gcc-13-locales gdb-x86-64-linux-gnu apache2
  libtsan2 libubsan1 lto-disabled-list make python3-dev python3-pip
  python3-setuptools python3-wheel python3.12-dev zlib1g-dev
0 upgraded, 51 newly installed, 0 to remove and 1 not upgraded.
Need to get 64.8 MB of archives.
After this operation, 237 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

```
Setting up python3-dev (3.12.3-0ubuntu2) ...
Setting up gcc-13 (13.3.0-6ubuntu2~24.04) ...
Setting up g++-13-x86-64-linux-gnu (13.3.0-6ubuntu2~24.04) ...
Setting up gcc-x86-64-linux-gnu (4:13.2.0-7ubuntu1) ...
Setting up gcc (4:13.2.0-7ubuntu1) ...
Setting up g++-x86-64-linux-gnu (4:13.2.0-7ubuntu1) ...
Setting up g++-13 (13.3.0-6ubuntu2~24.04) ...
Setting up g++ (4:13.2.0-7ubuntu1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mo
de
Setting up build-essential (12.10ubuntu1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.4) ...
art@art-VMware-Virtual-Platform:~$
```

apt install pipx

```
art@art-VMware-Virtual-Platform:~$ sudo apt install pipx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-argcomplete python3-packaging python3-pip-whl python3-platformdirs
  python3-psutil python3-setuptools-whl python3-userpath python3-venv
  python3.12-venv
The following NEW packages will be installed:
  pipx python3-argcomplete python3-packaging python3-pip-whl
  python3-platformdirs python3-psutil python3-setuptools-whl python3-userpath
  python3-venv python3.12-venv
0 upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 3,508 kB of archives.
After this operation, 7,832 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

pipx install aws-sam-cli

```
art@art-VMware-Virtual-Platform:~$ pipx install aws-sam-cli
⣷ installing aws-sam-cli
```

```
art@art-VMware-Virtual-Platform:~$ pipx install aws-sam-cli
  installed package aws-sam-cli 1.137.1, installed using Python 3.12.3
  These apps are now globally available
    - sam
⚠️ Note: '/home/art/.local/bin' is not on your PATH environment variable.
    These apps will not be globally accessible until your PATH is updated. Run
    `pipx ensurepath` to automatically add it, or manually modify your PATH in
    your shell's config file (i.e. ~/.bashrc).
done! ✨ 🌟 ✨
art@art-VMware-Virtual-Platform:~$
```

pipx ensurepath

```
art@art-VMware-Virtual-Platform:~$ pipx ensurepath
Success! Added /home/art/.local/bin to the PATH environment variable.

Consider adding shell completions for pipx. Run 'pipx completions' for
instructions.

You will need to open a new terminal or re-login for the PATH changes to take
effect.

Otherwise pipx is ready to go! ✨ 🌟 ✨
art@art-VMware-Virtual-Platform:~$
```

sam init

```
art@art-VMware-Virtual-Platform:~$ sam init

        SAM CLI now collects telemetry to better understand customer needs.

        You can OPT OUT and disable telemetry collection by setting the
        environment variable SAM_CLI_TELEMETRY=0 in your shell.
        Thanks for your help!

        Learn More: https://docs.aws.amazon.com/serverless-application-model/lat
est/developerguide/serverless-sam-telemetry.html


You can preselect a particular runtime or package type when using the `sam init`
 experience.
Call `sam init --help` to learn more.

Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice:  1
```

Type 1:

```
Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Data processing
        3 - Hello World Example with Powertools for AWS Lambda
        4 - Multi-step workflow
        5 - Scheduled task
        6 - Standalone function
        7 - Serverless API
        8 - Infrastructure event management
        9 - Lambda Response Streaming
        10 - GraphQLApi Hello World Example
        11 - Full Stack
        12 - Lambda EFS example
        13 - Serverless Connector Hello World Example
        14 - Multi-step workflow with Connectors
        15 - DynamoDB Example
        16 - Machine Learning
Template: 1
```

```
Use the most popular runtime and package type? (python3.13 and zip) [y/N]: y

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

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

Would you like to set Structured Logging in JSON format on your Lambda functions
?  [y/N]: y
```

Enter project name:

```
Project name [sam-app]: sam-test-app
```

```
Project name [sam-app]: sam-test-app

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

    -----------------------
    Generating application:
    -----------------------
    Name: sam-test-app
    Runtime: python3.13
    Architectures: x86_64
    Dependency Manager: pip
    Application Template: hello-world
    Output Directory: .
    Configuration file: sam-test-app/samconfig.toml

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

cd sam-test-app

```
art@art-VMware-Virtual-Platform:~$ cd sam-test-app
art@art-VMware-Virtual-Platform:~/sam-test-app$ pwd
/home/art/sam-test-app
art@art-VMware-Virtual-Platform:~/sam-test-app$
```

sudo apt-get install ca-certificates curl

```
art@art-VMware-Virtual-Platform:~/sam-test-app$ sudo apt-get install ca-certific
ates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
art@art-VMware-Virtual-Platform:~/sam-test-app$
```

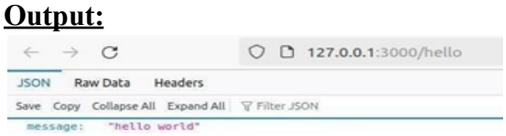
sam local invoke "HelloWorldFunction"

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

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

sam local start-api

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

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

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

Output:



Result and Discussion:
We have successfully implemented Hello World Flow Activity

Learning Outcomes:
1. Successfully implemented Hello World Program by AWS Flow Framework
2. Understood AWS Flow Framework

Course Outcomes:
1. We have learned about AWS Flow Framework
2. We have understood AWS Flow Framework

Conclusion:
We have successfully implemented Hello World Flow Activity

Viva Question:
1. What is AWS?
2. What is Cloud Service?
3. What is Worker in AWS Flow Framework?
4. What is Workflow in AWS Flow Framework?

For Faculty Use

Correction Parameters	Formative Assessment [ ]	Timely completion practical [ ]	Attendance Learning Attitude[ ]

Theory-10
OpenStack

OpenStack is an open-source cloud computing platform that provides Infrastructure-as-a-Service (IaaS). It allows users to create and manage private and public clouds by pooling virtual resources like compute, storage, and networking. Here's a breakdown:

What is OpenStack?
OpenStack is a modular platform made up of interrelated components, called "projects," that handle core cloud services:
- Compute (Nova): Manages virtual machines and other compute resources.
- Networking (Neutron): Provides network connectivity for instances.
- Storage (Swift and Cinder): Handles object storage and block storage.
- Identity (Keystone): Manages authentication and authorization.
- Image Services (Glance): Stores and retrieves virtual machine disk images.

OpenStack Working:
OpenStack relies on virtualization and a base operating system (often Linux) to create cloud environments. It uses APIs to abstract virtual resources into pools, which are then used to power cloud computing tools. Administrators and users interact with OpenStack through a web-based dashboard, command-line tools, or RESTful APIs.

Benefits of OpenStack
- Flexibility: Build custom cloud environments tailored to specific needs.
- Cost-Effective: Open-source nature eliminates licensing fees.
- Scalability: Easily scale resources up or down based on demand.
- Community Support: Backed by a large community of developers and contributors.

Real-Life Applications
- Hosting private clouds for businesses.
- Managing public clouds for service providers.
- Supporting hybrid cloud environments that combine private and public clouds.

Practical-10: Implementation of Openstack with user and private network creation.

Steps:

Install Linux

Start with a clean and minimal install of a Linux system. DevStack attempts to support the two latest LTS releases of Ubuntu, Rocky Linux 9 and openEuler.

Add Stack User (optional)

DevStack should be run as a non-root user with sudo enabled (standard logins to cloud images such as “ubuntu” or “cloud-user” are usually fine).

If you are not using a cloud image, you can create a separate stack user to run DevStack with

sudo useradd -s /bin/bash -d /opt/stack -m stack

Ensure home directory for the stack user has executable permission for all, as RHEL based distros create it with 700 and Ubuntu 21.04+ with 750 which can cause issues during deployment.

sudo chmod +x /opt/stack

Since this user will be making many changes to your system, it should have sudo privileges:

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

Then

sudo -u stack -i

Download DevStack

git clone https://opendev.org/openstack/devstack

Then

cd devstack

The devstack repo contains a script that installs OpenStack and templates for configuration files.

Create a local.conf

Create a local.conf file with four passwords preset at the root of the devstack git repo.

touch local.conf

Then

nano local.conf

Then paste this:

[[local|localrc]]
ADMIN_PASSWORD=secret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

This is the minimum required config to get started with DevStack.

Start the install

./stack.sh

This will take 15 - 30 minutes, largely depending on the speed of your internet connection. Many git trees and packages will be installed during this process.

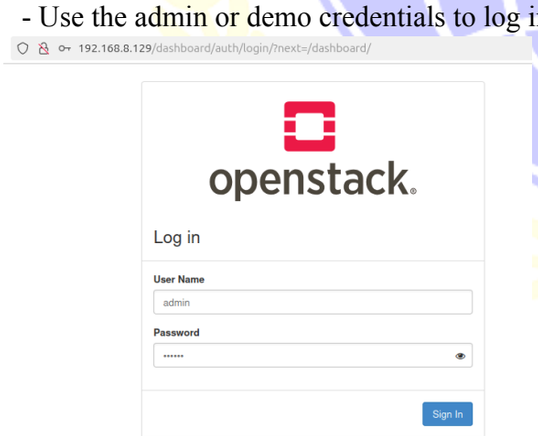
After successful installation:

```
This is your host IP address: 192.168.8.129
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.8.129/dashboard
Terminal is serving at http://192.168.8.129/identity/
The default users are: admin and demo
The password: secret

Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/devstack/latest/systemd.html

DevStack Version: 2025.2
Change: 608ae718fca61e557b315f87900a54a7fb40b07e Merge "Skip functional tests fo
r .gitreview update" 2025-03-28 19:25:37 +0000
OS Version: Ubuntu 24.04 noble

2025-04-20 06:30:59.311 | stack.sh completed in 890 seconds.
stack@art-VMware-Virtual-Platform:~/devstack$
```

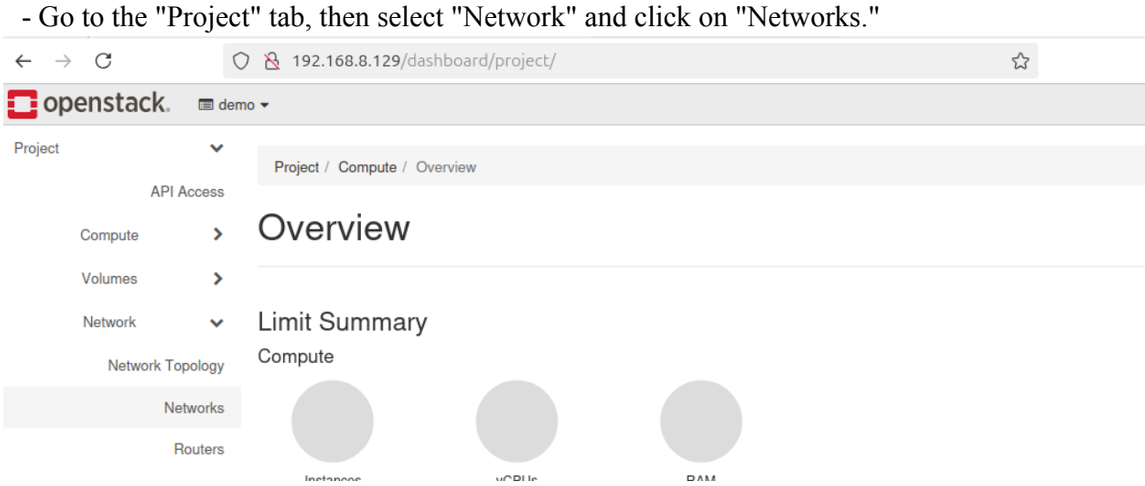
A private network in OpenStack is a virtual network that is isolated from external networks. It allows instances (virtual machines) to communicate with each other securely within the same project or tenant. This is particularly useful for internal communication between instances without exposing them to the public internet. Here's how you can create and use a private network in OpenStack:

Steps to Create a Private Network:

1. Log in to the OpenStack Dashboard (Horizon):
   - Use the admin or demo credentials to log in.

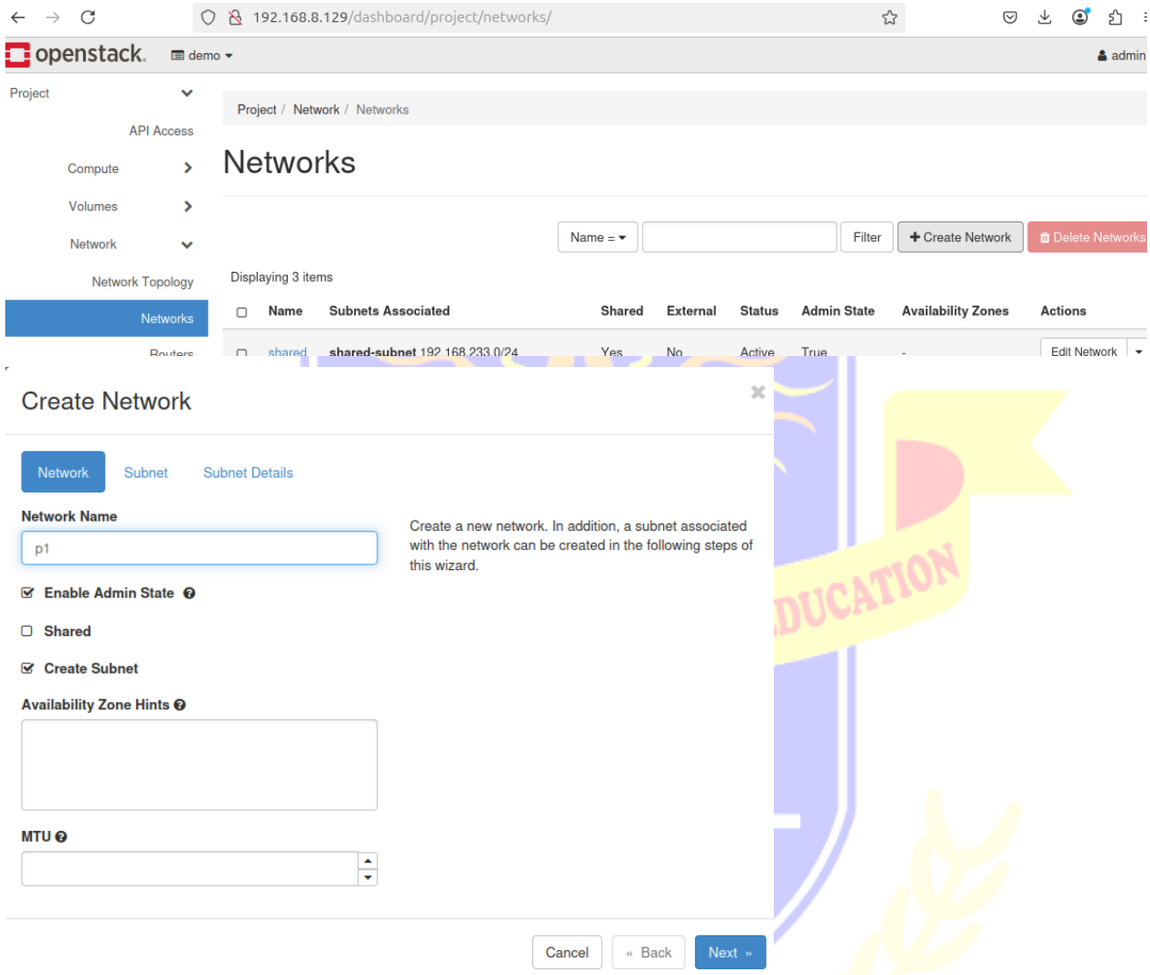


2. Navigate to the Network Section:
   - Go to the "Project" tab, then select "Network" and click on "Networks."

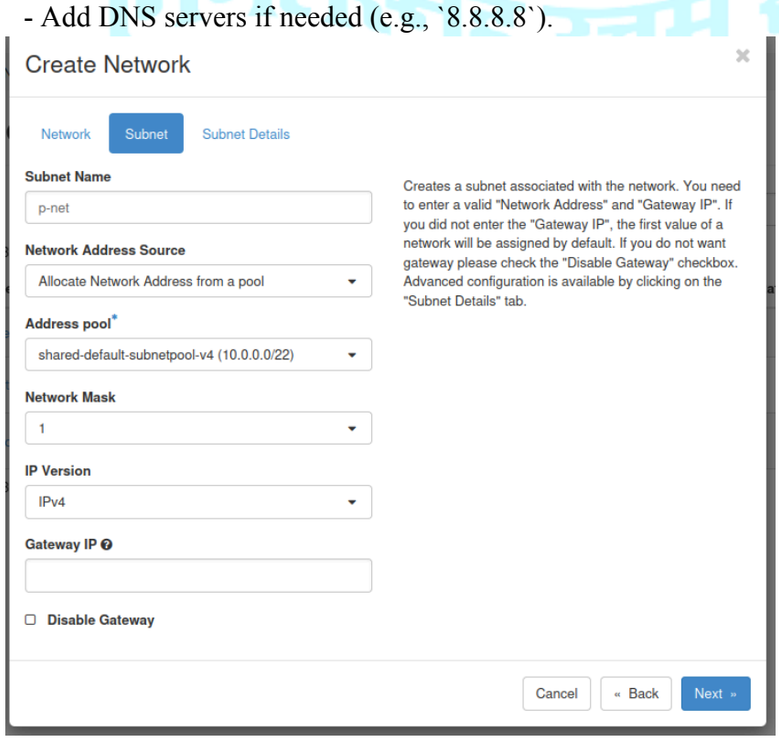


3. Create a New Network:
   - Click on the "Create Network" button.

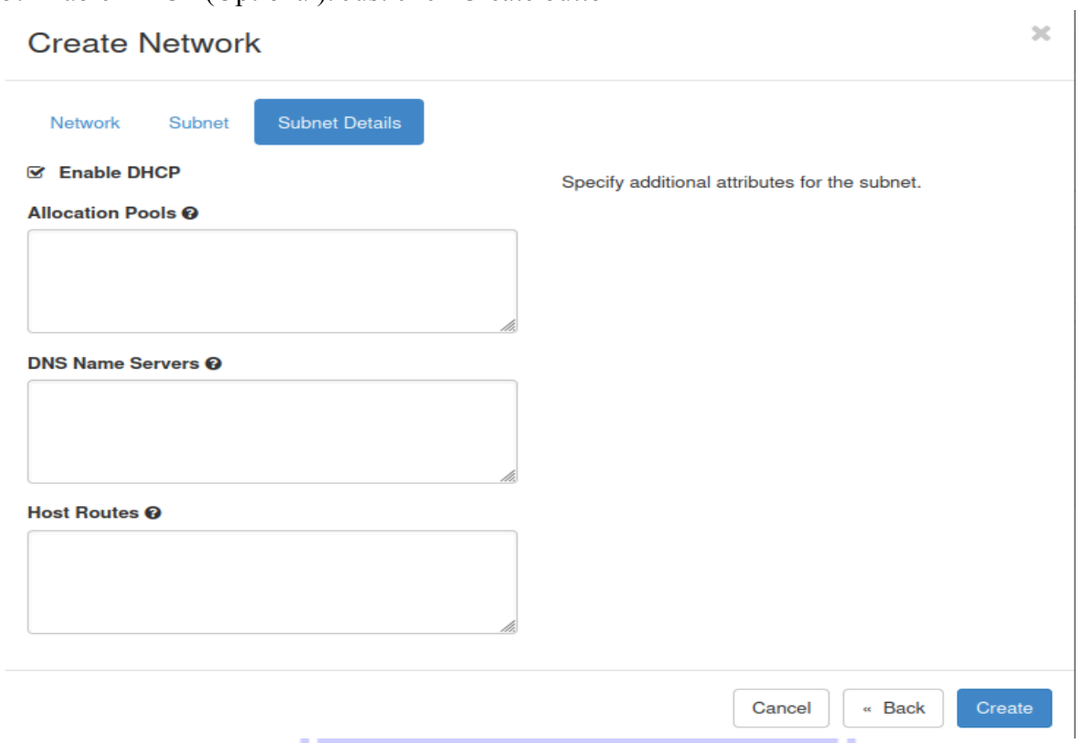
- Provide a name for your network (e.g., `private-network`).
- Leave the "Shared" option unchecked to keep it private.



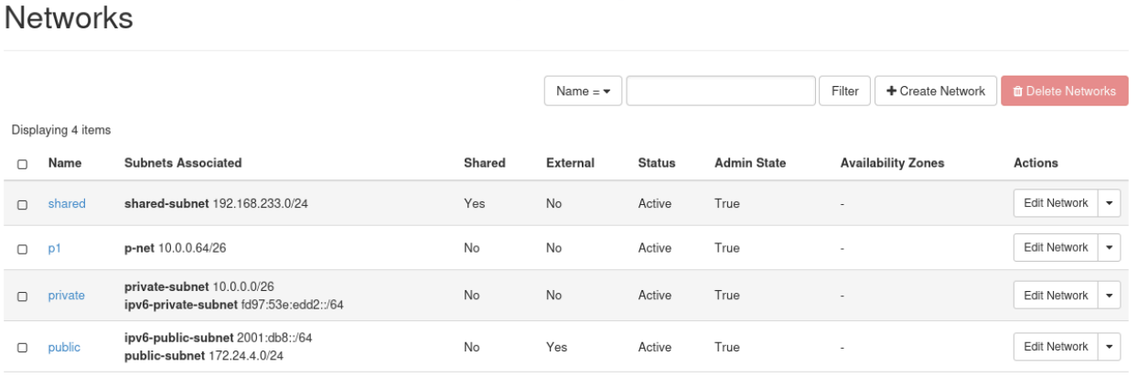
4. Add a Subnet:
   - In the "Subnet" tab, provide a name for the subnet (e.g., `private-subnet`).
   - Specify the network address (e.g., `192.168.10.0/24`).
   - Set the gateway IP (e.g., `192.168.10.1`) or leave it as default.
   - Add DNS servers if needed (e.g., `8.8.8.8`).



5. Enable DHCP (Optional): Just click Create button



Private Network is created:



Result and Discussion:
We have successfully demonstrated installation of openstack and creation of private network

Learning Outcomes:
1. Successfully installed openstack
2. Understood components of openstack

Course Outcomes:
1. We have learned about openstack
2. We have experimented with openstack

Conclusion:
We have successfully demonstrated the installation of openstack.

Viva Question:
1. What is OpenStack?
2. What is KeyStone in OpenStack?
3. What is Nova in OpenStack?
4. What is Neutron in OpenStack?

For Faculty Use

Correction Parameters	Formative Assessment [ ]	Timely completion practical [ ]	Attendance Learning Attitude[ ]