**PROGRAM: (CONVERTING RS INTO DOLARS)**

const express = require('express'); const axios = require('axios');

const app = express();

const exchangeRateApi = 'https://api.exchangerate-api.com/v4/latest/INR';

app.get('/convert', async (req, res) => { const { amount, to } = req.query;

try {

const response = await axios.get(exchangeRateApi); const exchangeRate = response.data.rates[to];

if (!exchangeRate) {

return res.status(400).json({ error: 'Invalid currency code' });

}

const convertedAmount = (amount \* exchangeRate).toFixed(2); res.json({ convertedAmount });

} catch (error) { console.error(error);

res.status(500).json({ error: 'An error occurred while converting currency' });

}

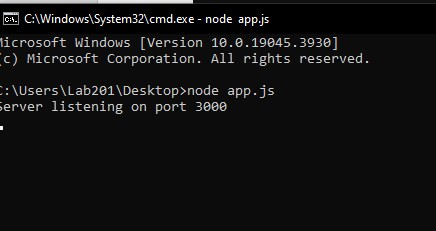
});

const PORT = process.env.PORT || 3000; app.listen(PORT, () => { console.log(`Server listening on port ${PORT}`);

});

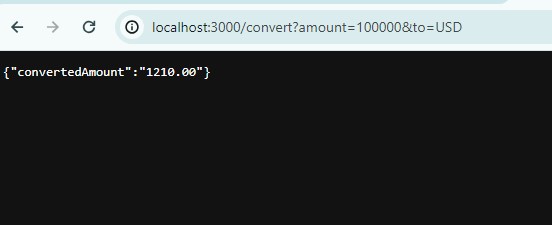
Go to desktop>cmd> type node filename.js (node app.js)

Copy port no (3000)



Go to browser type--- localhost:3000/convert?amount=100000&to=USD

**OUTPUT:**



**PROGRAM: (SIMPLE SOAP SERVICE USING NODE.js)**

const express = require("express");

var app = express();

app.get("/add/:num\_1/:num\_2", function(req, res) { const num1 = req.params.num\_1; const num2 = req.params.num\_2; const result = parseInt(num1) + parseInt(num2); res.json({result: result});

})

app.get("/sub/:num\_1/:num\_2", function(req, res) { const num1 = req.params.num\_1; const num2 = req.params.num\_2; const result = parseInt(num1) - parseInt(num2); res.json({result: result});

})

app.get("/mult/:num\_1/:num\_2", function(req, res) { const num1 = req.params.num\_1; const num2 = req.params.num\_2; const result = parseInt(num1) \* parseInt(num2); res.json({result: result});

})

app.get("/div/:num\_1/:num\_2", function(req, res) { const num1 = req.params.num\_1;

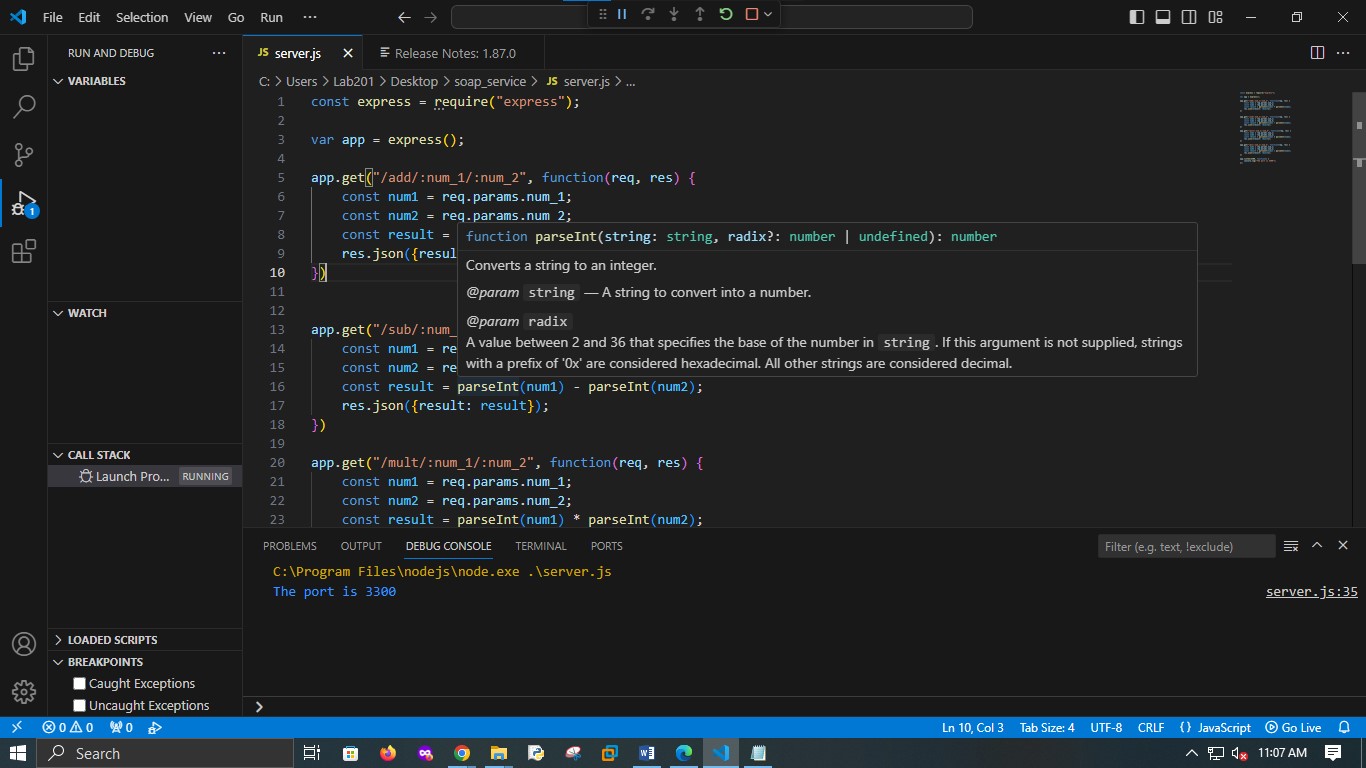
const num2 = req.params.num\_2; const result = parseInt(num1) / parseInt(num2); res.json({result: result});

})

app.listen(3300, function() { console.log("The port is 3300");

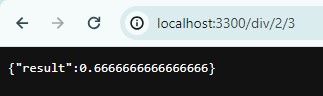
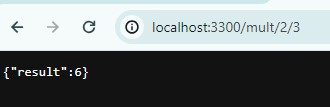
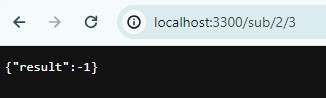
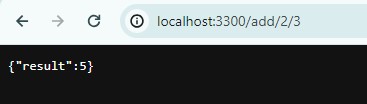
});

**Run in vs code copy the local host port number**



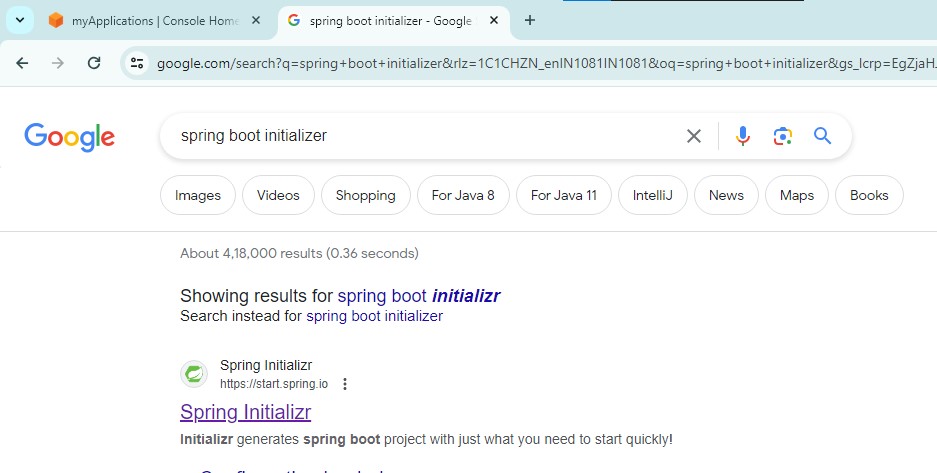
**Open google chrome type--** <http://localhost:3300/add/2/3><http://localhost:3300/sub/2/3><http://localhost:3300/mult/2/3><http://localhost:3300/div/2/3>

**OUTPUT:**



**PROGRAM:**

**OPEN GOOGLE CROME >search spring boot initializer**

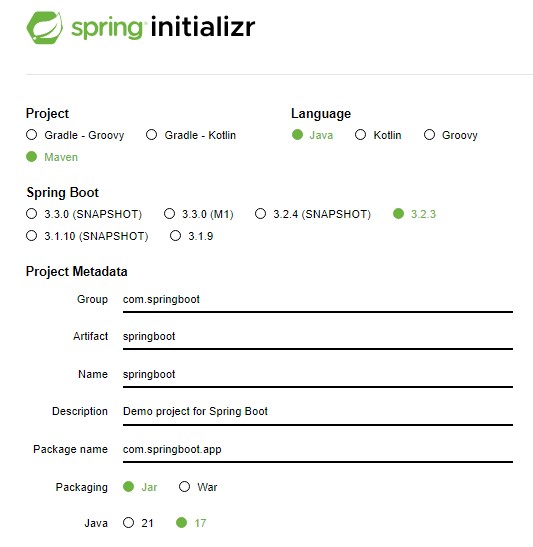


**SELECT>PROJECT:MAVEN**

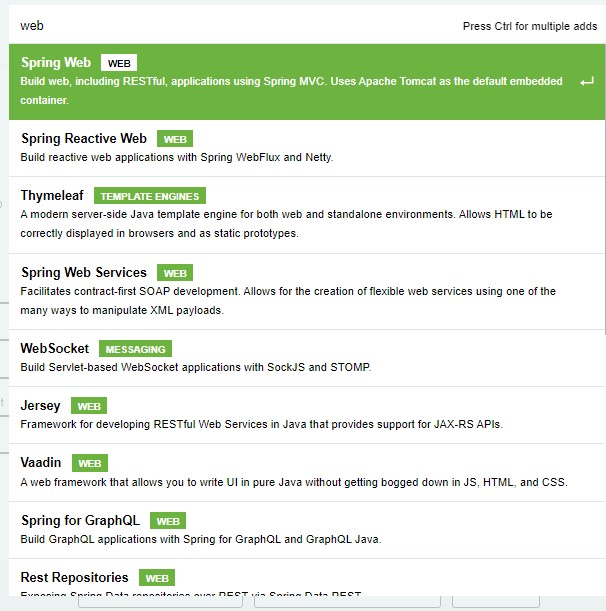
**LANGUAGE:JAVA**

**SPRING BOOT :3.2.3**

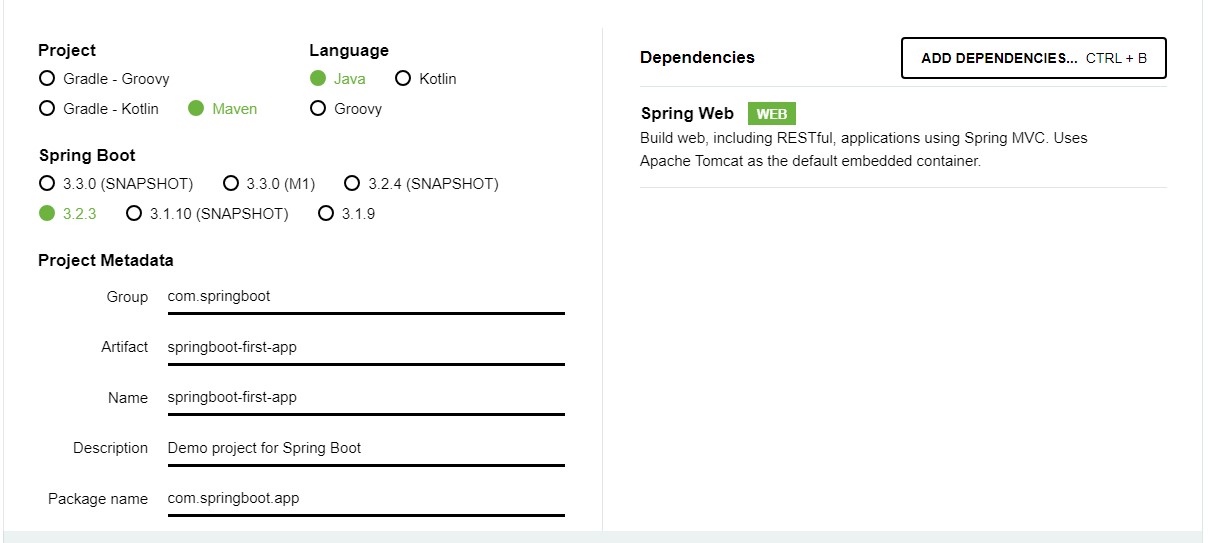
**FILL UP PROJECT METADATA**



**GO TO DEPENDENCIES >SEARCH WEB**



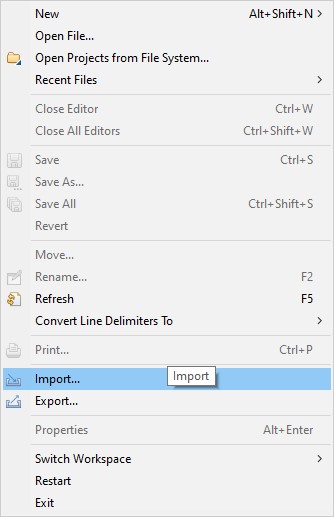
**ADD DEPENDENCIES**



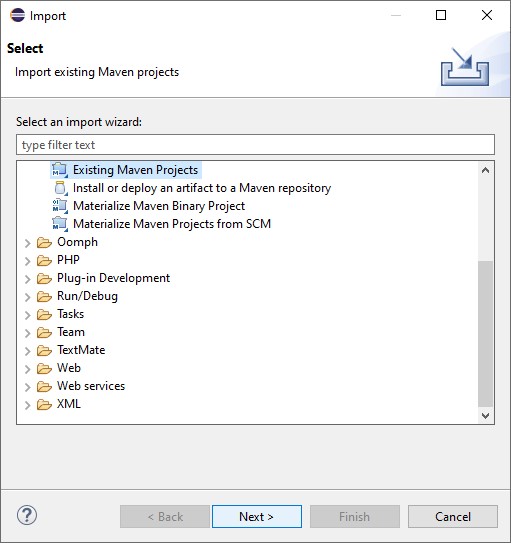
**CLICK ON GENERATE BUTTON (BELOW) TO DOWNLOAD SPRING BOOT PROJECT AS ZIP FILE**

**EXTRACT THE ZIP FILE**

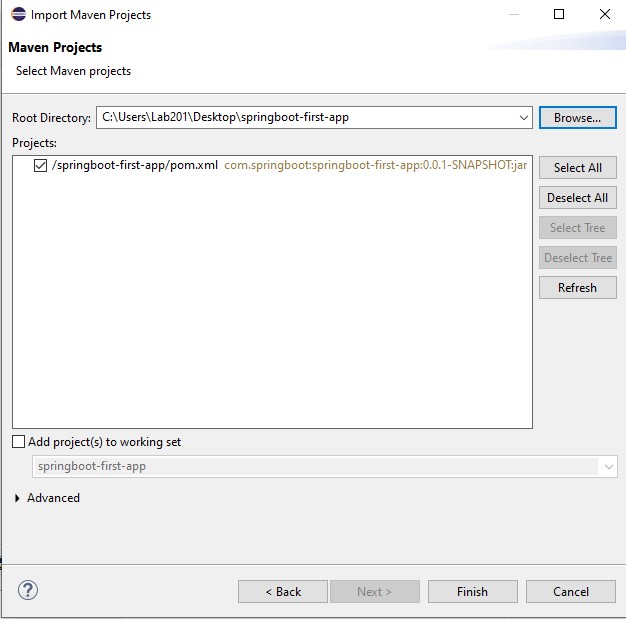
**OPEN ECLIPSE>GO TO FILE>IMPORT**



**SELECT EXISTING MAVEN PROJECT**

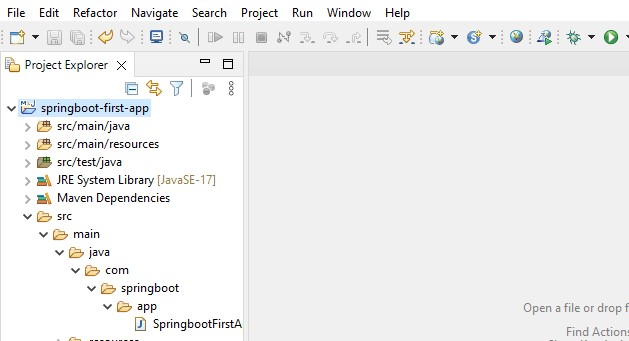


**BROWSE SPRINGBOOT-FIRAT-APP >CLICK ON FINISH**

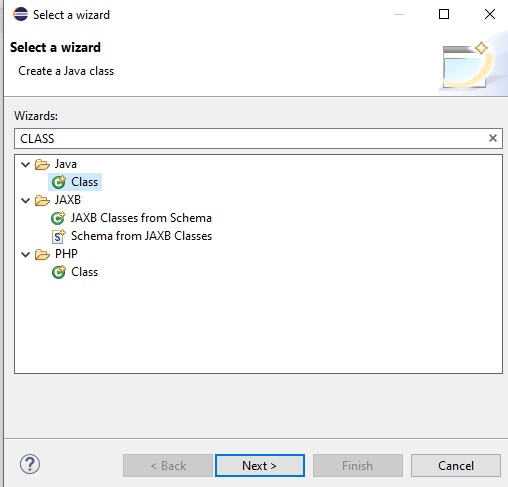
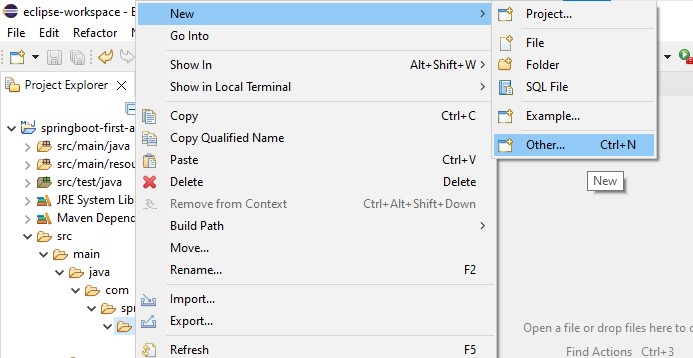


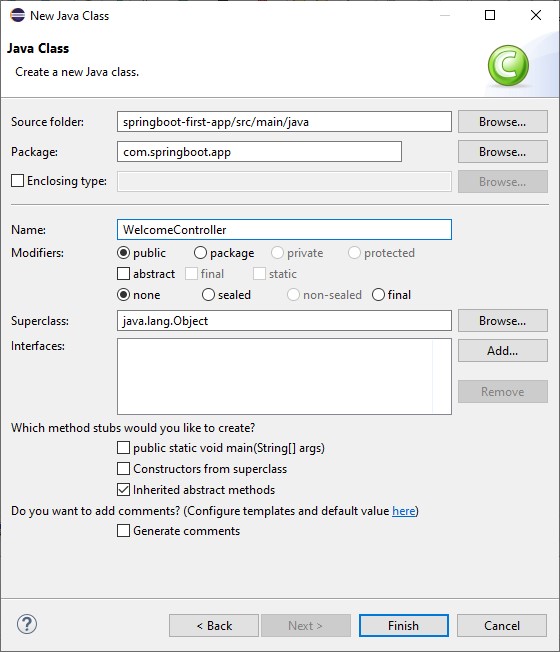
**SELECT SPRINGBOOT-FIRST-**

**APP>SRC>MAIN>JAVA>COM>SPRINGBOOT>APP**



**RIGHT CLICK ON APP>NEW>OTHERS>SELECT CLASS>NEXT**

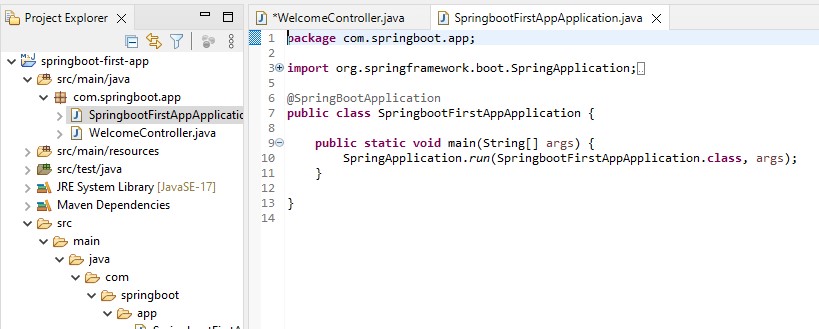




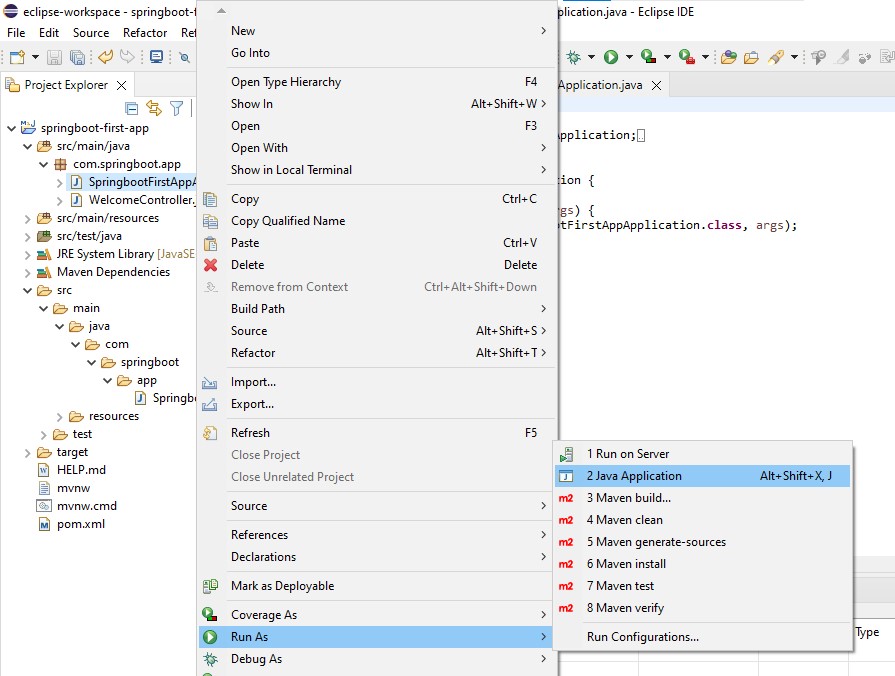
**WRITE CODE IN WELCOMECONTROLLER.JAVA FILE**



**NOW GO TO springbootfirstappapplication.java**

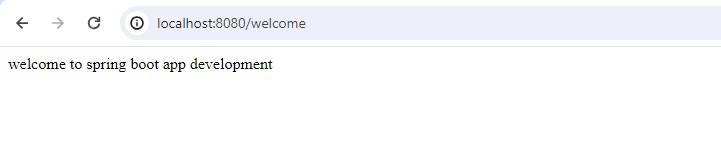


**RUN THE PROGRAM (AS JAVA APPLICATION)**



**GO TO BROWSER TYPE>localhost:8080/welcome**

**OUTPUT:**



**PROGRAM: (GOOGLE’s MAP RESTFUL WEB SERVICE)**

import requests

def get\_geolocation(api\_key, search\_string):

base\_url = "https://us1.locationiq.com/v1/search" params = {

'key': api\_key,

'q': search\_string,

'format': 'json',

}

response = requests.get(base\_url, params=params) data = response.json()

if response.status\_code == 200 and data: result = {

'place\_id': data[0].get('place\_id', ''),

'lat': data[0].get('lat', ''),

'lon': data[0].get('lon', ''),

'display\_name': data[0].get('display\_name', ''),

} return result else:

print(f"Error: {response.status\_code} - {data.get('error', 'No error message')}") return None

api\_key = 'pk.71c93f03731ac10faf75d7e071df51c1 ' search\_string = input("Enter the location : ")

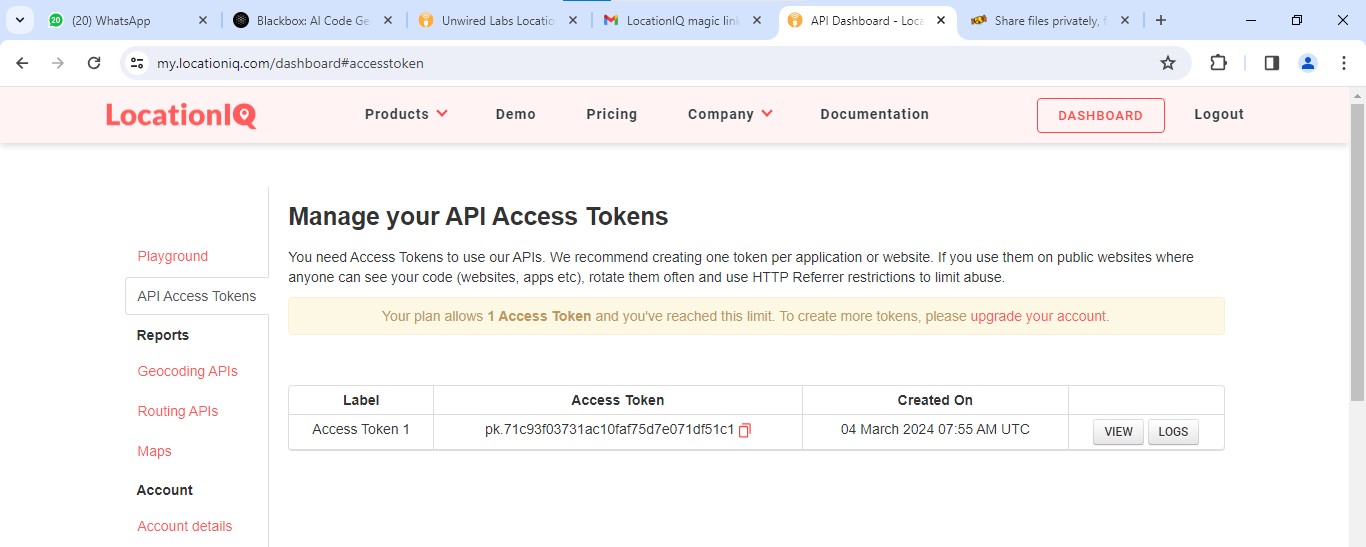
result = get\_geolocation(api\_key, search\_string)

if result:

print("Output:") for key, value in result.items():

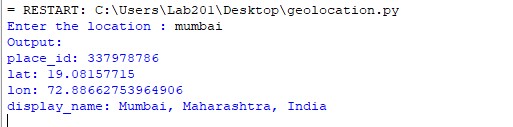
print(f"{key}: {value}")

BROWSE LOCATION IQ>SIGNUP WITH EMAIL> CLICK ON THE LINK PROVIDED BY LOCATION IQ (on your email)>YOU WILL GET YOUR ACCESS TOKEN COPY THE KEY AND PASTE IN THE PYTHON CODE:



RUN THE CODE

**Output:**



**PROGRAM: (INSTALLATION AND CONFIGURATION OF VIRTUALIZATION USING KVM) COMMANDS:**

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

**2.sudo apt install qemu-kvm libvirt-daemon-system virt-manager brid**

**3.sudo apt-get update**

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

**5.sudo apt install qemu-kvm libvirt-clients libvirt-daemon-system bridge-utils**

**6.sudo systemctl start libvirtd**

**7.sudo usermod -aG kvm $USER**

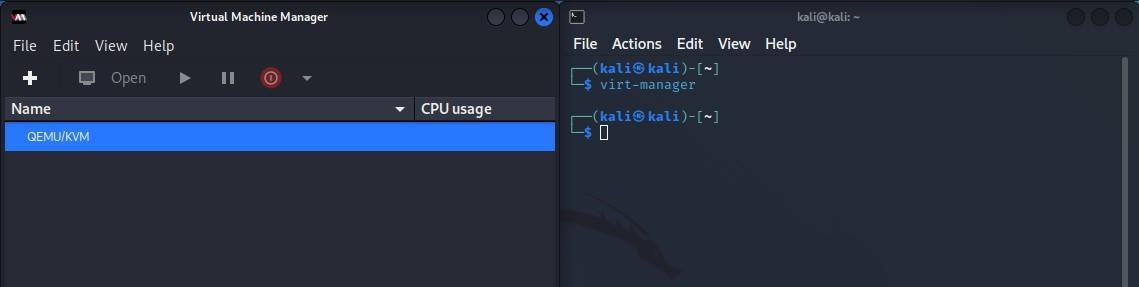
**8.sudo systemctl is-active libvirtd**

**9.sudo usermod -aG libvirt $USER sudo usermod -aG kvm $USER**

**10.virt-manager**

**11.kvm-ok**

**OUTPUT:**



**PROGRAM: ( application to download image/video from server or upload image/video to server using MTOM techniques) (node.js) code:**

const express = require('express'); const multer = require('multer'); const path = require('path'); const fs = require('fs'); const app = express(); const port = 3000;

// Define storage using multer.diskStorage const storage = multer.diskStorage({ destination: (req, file, cb) => {

// Set the destination folder where the file will be saved const uploadFolder = 'uploads'; fs.mkdirSync(uploadFolder, { recursive: true }); cb(null, uploadFolder);

}, filename: (req, file, cb) => {

// Set the filename to the original filename cb(null, file.originalname);

},

});

const upload = multer({ storage: storage }); app.post('/upload', upload.single('file'), (req, res) => { const file = req.file;

// Check if file is present if (!file) { return res.status(400).json({ success: false, message: 'No file uploaded.' }); }

// Process the file as needed (save to disk, database, etc.) res.json({ success: true, message: 'File uploaded successfully.' });

}); app.get('/download/:filename', (req, res) => { const filename = req.params.filename; const filePath = path.join(\_\_dirname, 'uploads', filename);

// Check if file exists if (fs.existsSync(filePath)) {

// Implement logic to send the file as a response res.sendFile(filePath);

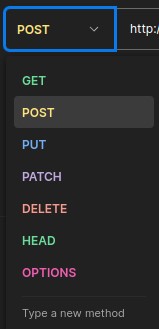
} else { res.status(404).json({ success: false, message: 'File not found.' }); } }); app.listen(port, () => { console.log(Server is running on http://localhost:${port});

});

**RUN THE CODE: IT WILL START THE SERVER AT localhost:3000**

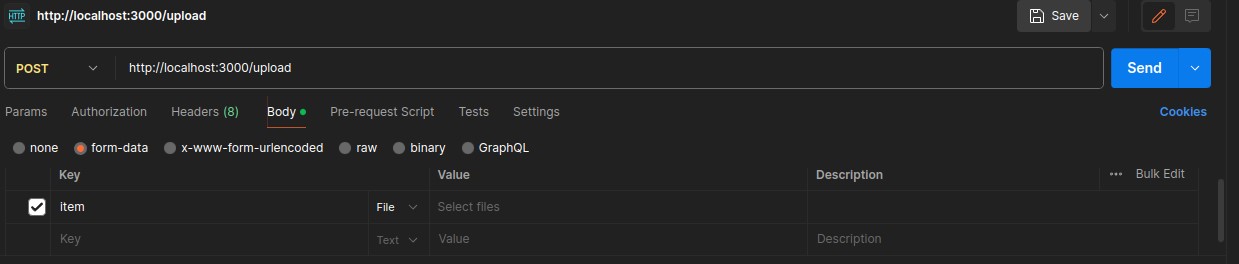


**OPEN POSTMAN OPEN A NEW PAGE & CHOOSE POST METHOD**

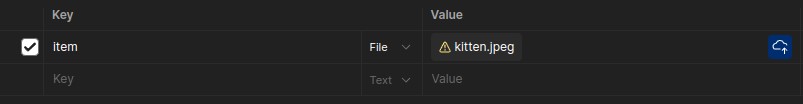


**ENTER THE URL OF THE SERVER:** [**http://localhost:3000/upload**](http://localhost:3000/upload)

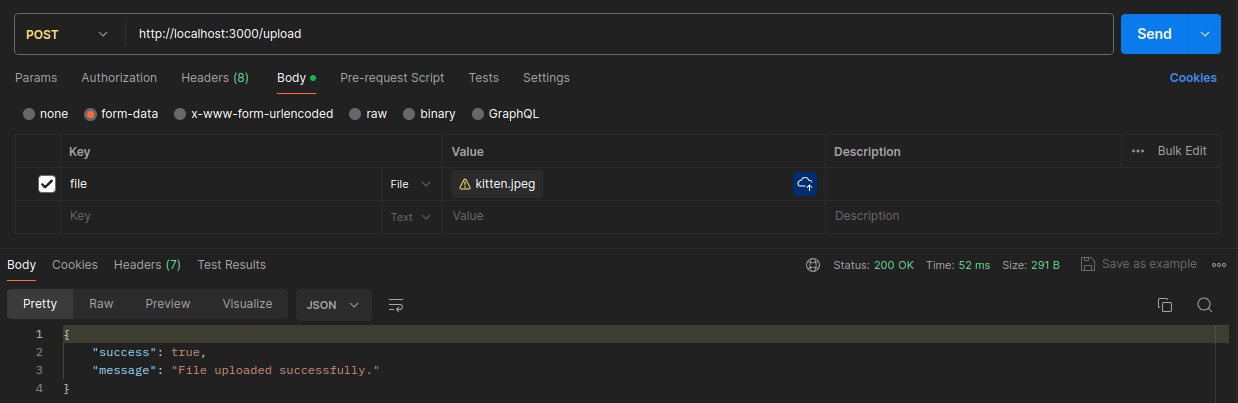
**THEN CLICK ON BODY>FORM DATA>NAME THE KEY ITEM AND FILE TYPE: FILE**



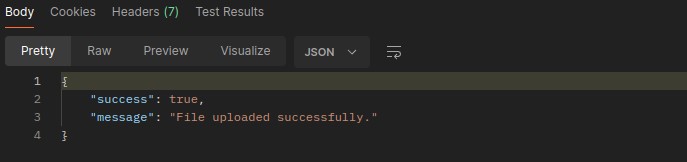
IN VALUE TAB ENTER THE FILE YOU WANT TO UPLOAD TO THE SERVER



CLICK ON SEND

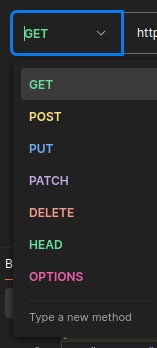


**Upload Output:**

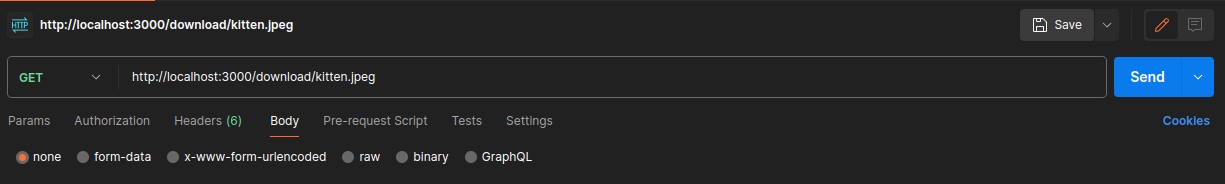


**DOWNLOAD:**

**CHOOSE GET METHOD**

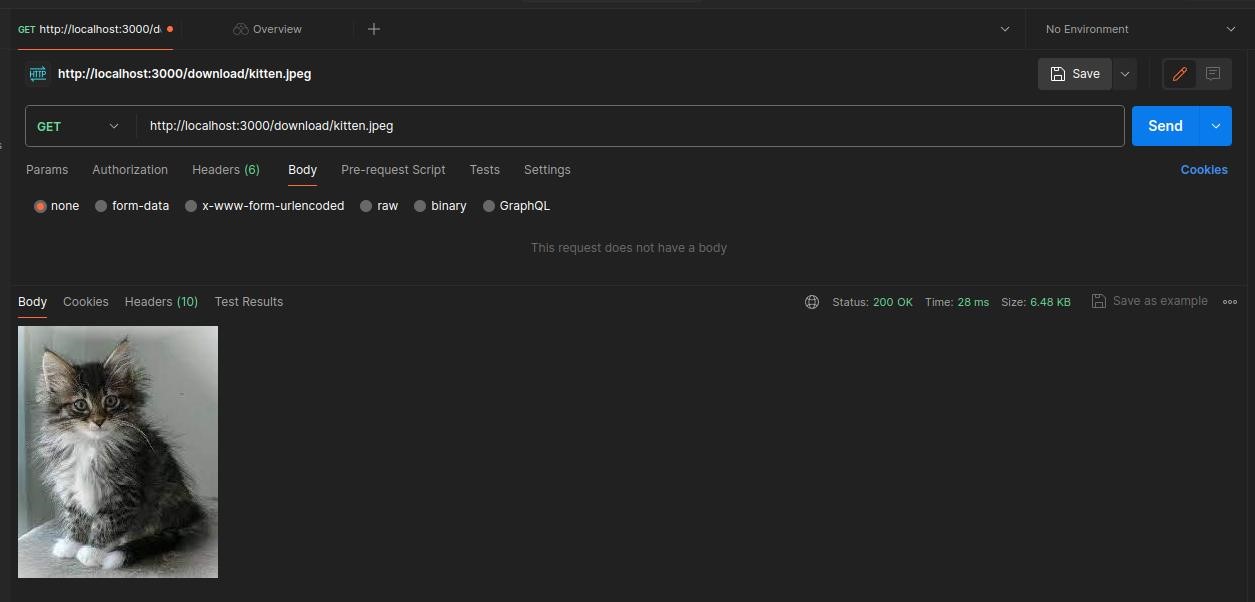


**ENTER THE URL OF THE SERVER:** [**http://localhost:3000/download/kitten.jpg**](http://localhost:3000/download/kitten.jpg)



CLICK ON SUBMIT

DOWNLOAD OUTPUT :



**PROGRAM: (Cloud Functionality VSI (Virtual Server Infrastructure) Infrastructure as a Service (IaaS), Storage)**

**AFTER INSTALLATION, IT WILL SHOW YOU AN IP ADDRESS. PUT IT IM**

**YOUR BROWSER TO ACCESS YOUR ADMINISTRATOR PAGE THE DEFAULT USER CREDENTIALS ARE USER: ADMIN AND PASSWORD: ADIMIN. FOR ROOT LOGIN - USERNAME- ROOT AND PASSWORD - PASSWORD**.



**THE FIRST SCREEN AFTER LOGIN SHOWS MANY OPTIONS TO INSTALL**

**AND DEPLOY ANY VIRTUAL MACHINE. TO INSTALL A VIRTUAL MACHINE**

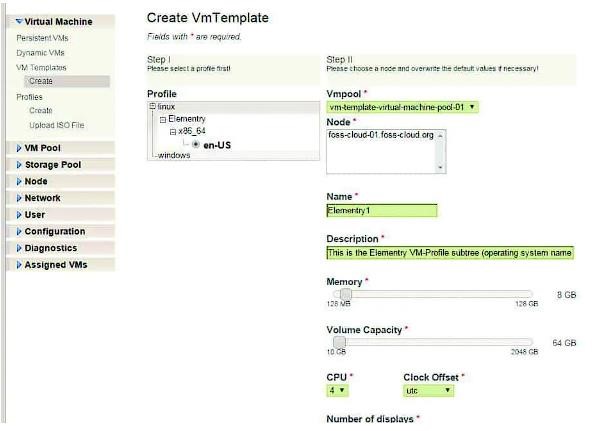
**CLICK ON VIRTUAI MACHINE-> UPLOAD ISO FILE OPTION AND UPLOAD THE BOOTABLE ISO FILE. HERE, WE ARE GOING TO UPLOAD LINUX ELEMENTARY 0S IS0**.



**ONCE YOU UPLOADED THE FILE, CREATE VIMTEMPLATE. IN THIS OPTION**

**YOU ARE BASICALLY CONFIGURING YOUR VIRTUAL MACHINE'S STORAGE**

**LOCATION, CPU, MEMORY, NODE ETC. HERE, YOU WILL FIND SINGLE NODES ANA VW PO01 1 RESPECTIVE OPTIONS BECAUSE EVERYTHING WAS INSTALLED AT THE SINGLE SERVER.**



**NOW, CLICK ON VMTEMPLATES AND YOU WILL SEE A TEMPLATE WHICH**

**YOU HAVE CREATED IN STEP 4. TO START YOUR MACHINE GO TO RUN**

**ACTION TAB AND CLICK ON THE GREEN ARROW. UNDER STATUS TAB, IT**

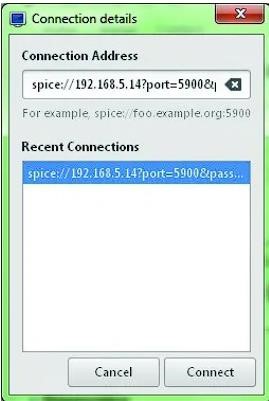
**SHOWS THE RUNNING TEXT WITH THE GREEN CIRCLE WHICH SHOWS**

**THAT YOUR MACHINE IS RUNNING WITHOUT ANY ERRORS. TO VIEW YOUR VIRTUAL MACHINE CLICK ON A BLUE SQUARE BOX UNDER ACTION TAB.**

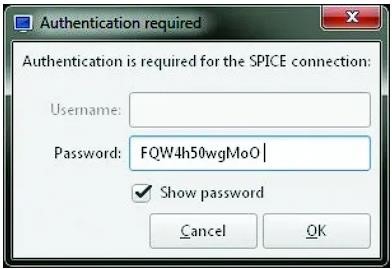


**TO VIEW YOUR VIRTUAL MACHINE YOU HAVE TO DOWNLOAD SPICE CLIENT TOOL. THE DOWNLOAD LINK CAN BE FOUND UNDER THE LINKS OPTION. AFTER DOWNLOADING THE APPLICATION, CLICK ON THE BLUE SQUARE TO VIEW MACHINE. WHEN YOU CLICK ON IT, THE BROWSER WILL POP-UP FOR LAUNCHING THE APPLICATION.**

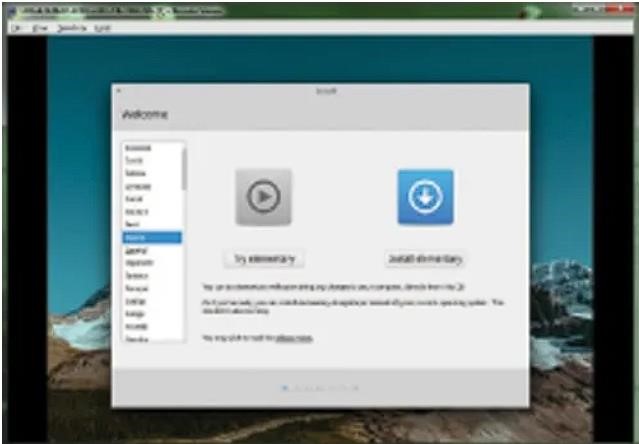
**IF THE APPLICATION DOES NOT LAUNCH AUTOMATICALLY, LAUNCH IT MANUALLY BY ENTERING THE LINK AND PASSWORD IN THE REMOTE VIEWER TOOL WHICH YOU WILL SEE IN THE POP-UP MESSAGE**.



**ONCE IT CONNECTS, ENTER THE PASSWORD WHICH WAS GIVEN IN THE LINK AND CLICK OK**.

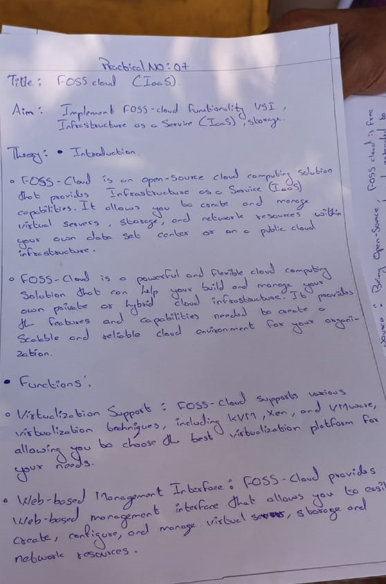


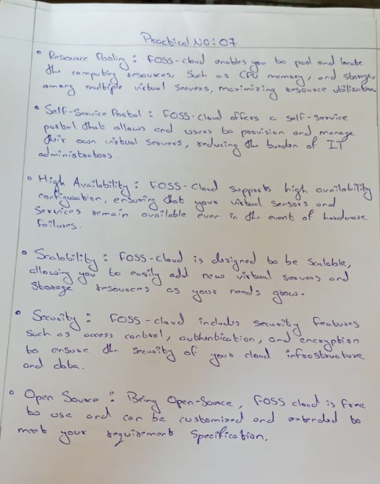
**FINALLY YOU WILL ABLE TO VIEW AND CONTROL YOUR VIRTUAL MACHINE.**

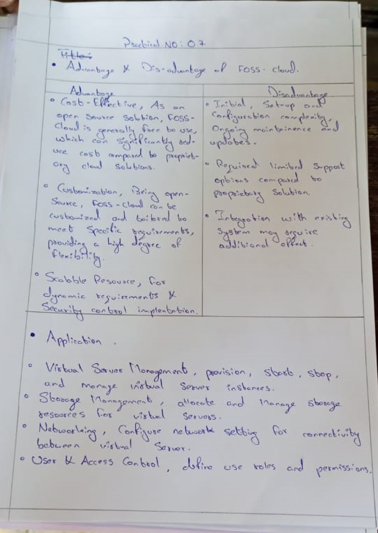


**Practical 7**

**Aim:foss cloud (Iaas)**





****

Practical 8

Aim:foss cloud (Paas)

