JAI SHRIRAM
ENGINEERING COLLEGE

(Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai, Accredited by NAAC, NBA Accredited for ECE & CSE)
Dharapuram Road, Avinashipalayam, Tirupur – 638 660.

Academic Year 2023-2024 (Even Semester)

LABORATORY RECORD

Certified that this is a bonafide record of work done by

Name : ……..

Reg. No. : ……..

Branch : **B. E - COMPUTER SCIENCE AND ENGINEERING**

Year & Semester : **III & VI**

Course Code & Name : **CCS335 & CLOUD COMPUTING**

Course In-Charge **Head of the Department**

Submitted for the University Practical Examination held on ……..

Internal Examiner **External Examiner**

JAI SHRIRAM ENGINEERING COLLEGE

TIRUPPUR – 638 660

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

Recognized by UGC & Accredited by NAAC and NBA (CSE and ECE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION AND MISSION

VISION

To produce a proficient software and hardware professionals, researchers along with entrepreneurs by imparting sound knowledge in Computer Science and Engineering with technical expertise to meet the global challenges

MISSION

M1: To enhance the students with practical knowledge in design and develop the innovative system to serve global demands and standards

M2: To create competent and high quality engineers by edify the state of the art tools and skills which are necessary to adopt emerging technological and societal changes

M3: To sensitize the graduates to become a responsible Computer Science Engineers with ethical and human values

INDEX

Ex. No.	Date	NameoftheExperiment	Page No.	Marks Awarded	Faculty Signature

EX NO. : 1 DATE:	**Install Virtualbox / VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.**

Aim:

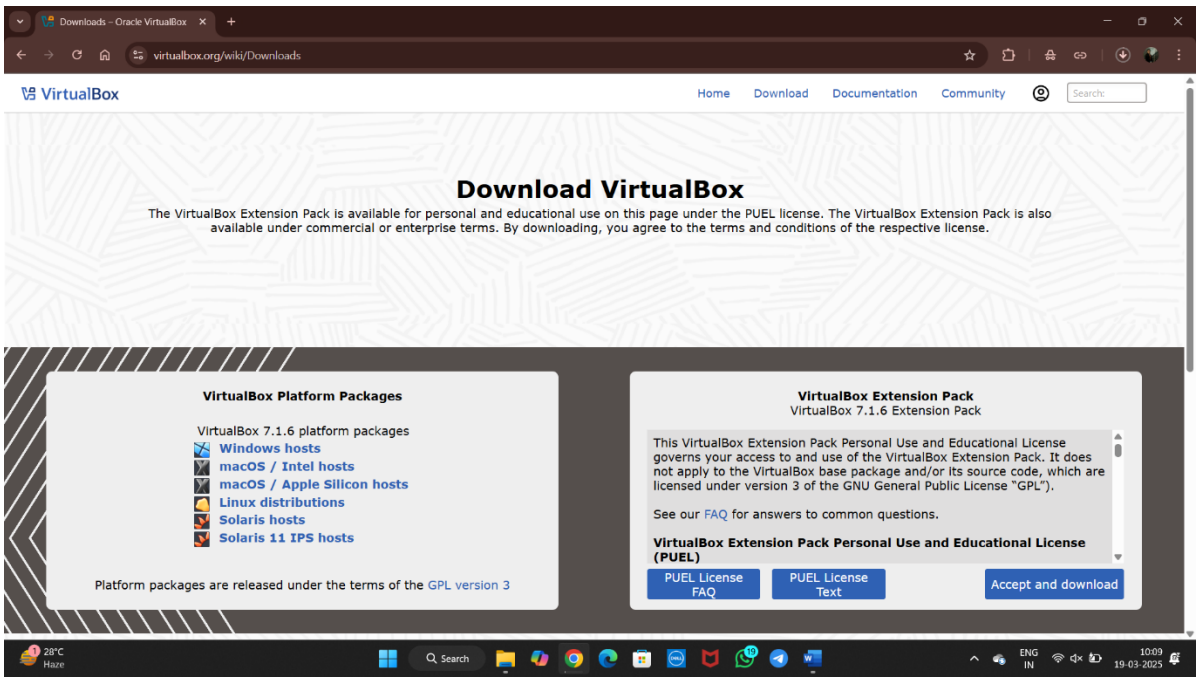
To Install Virtualbox / VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.

PROCEDURE:

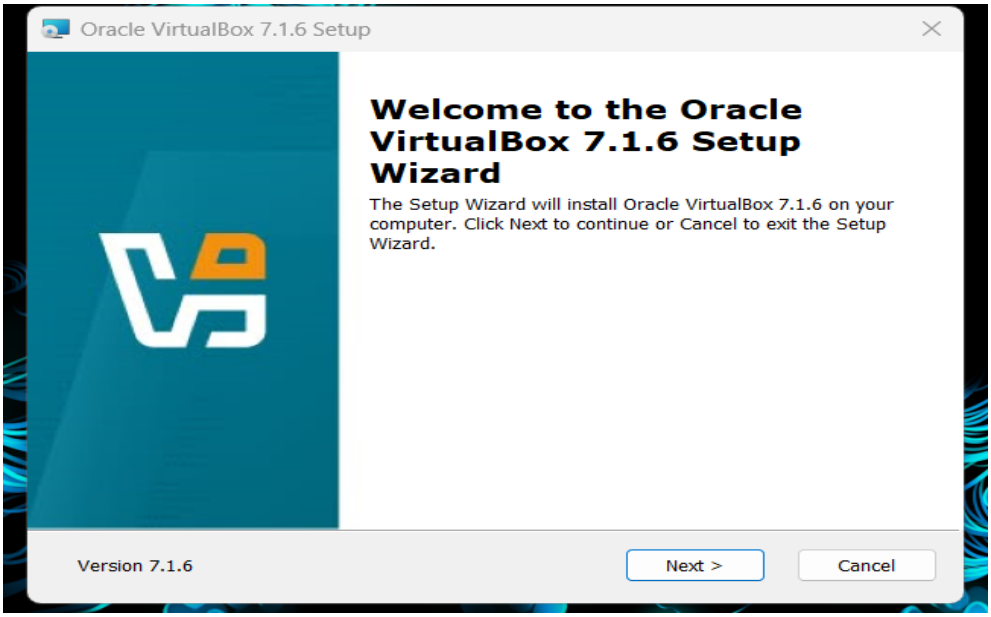
Steps to install Virtual Box:

1. Open the VirtualBox website. Go to https://www.virtualbox.org/ in your computer's Internet browser. This is the website from which you'll download the VirtualBox setup file.

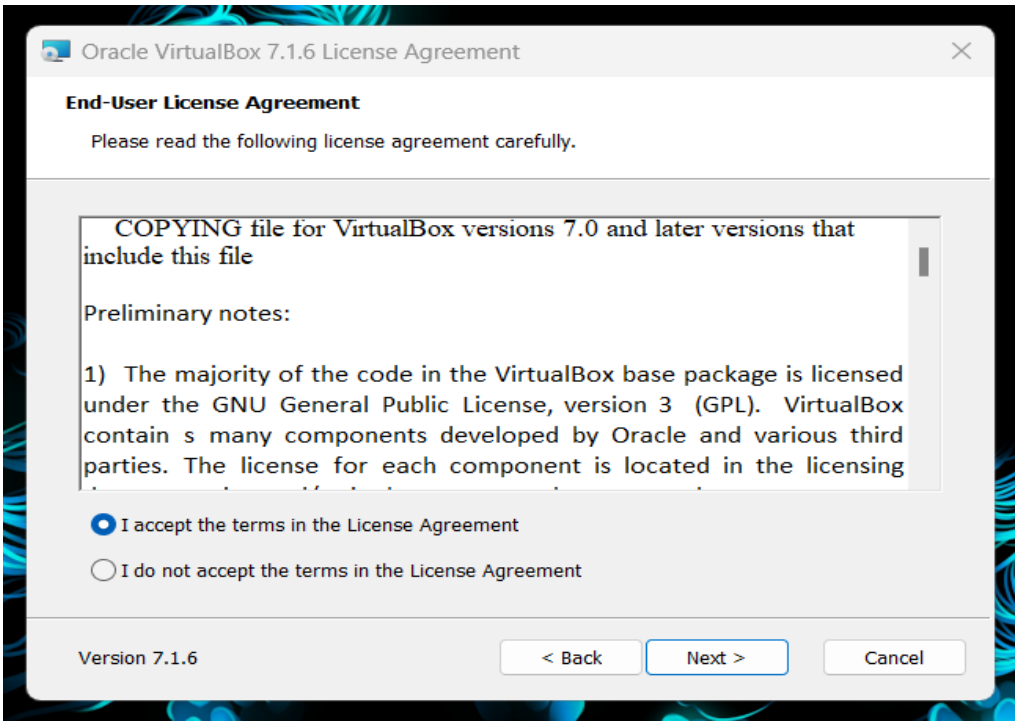
2. Click download virtual box. It's a blue button in the middle of the page. Doing so will open the downloads page.

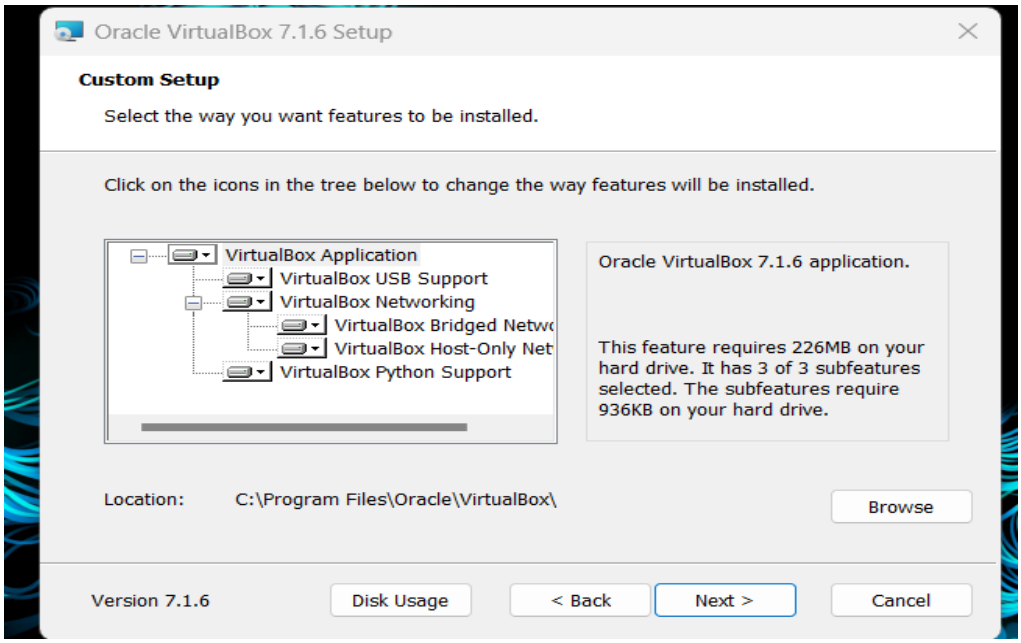


3. Click the next button

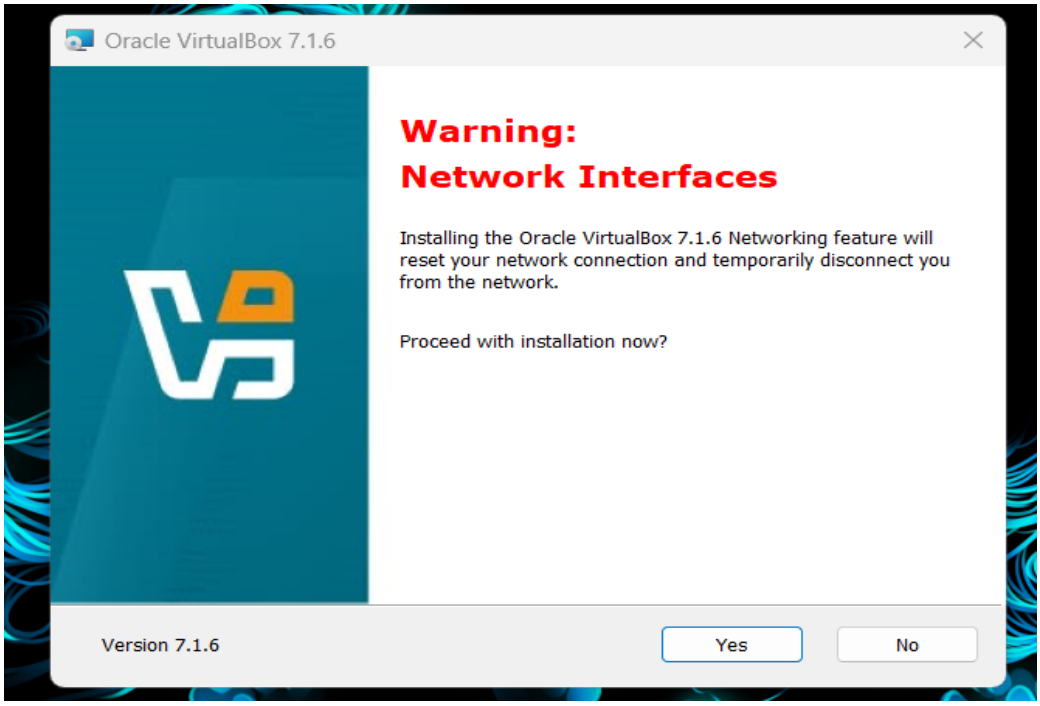


4. Click the next button

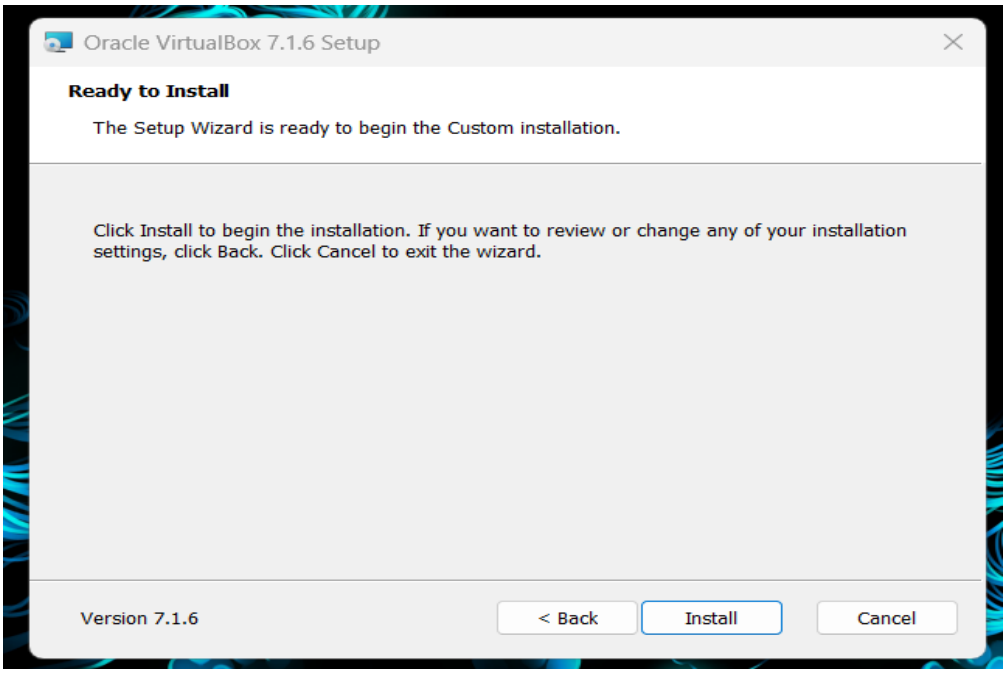




5. Click the yes button



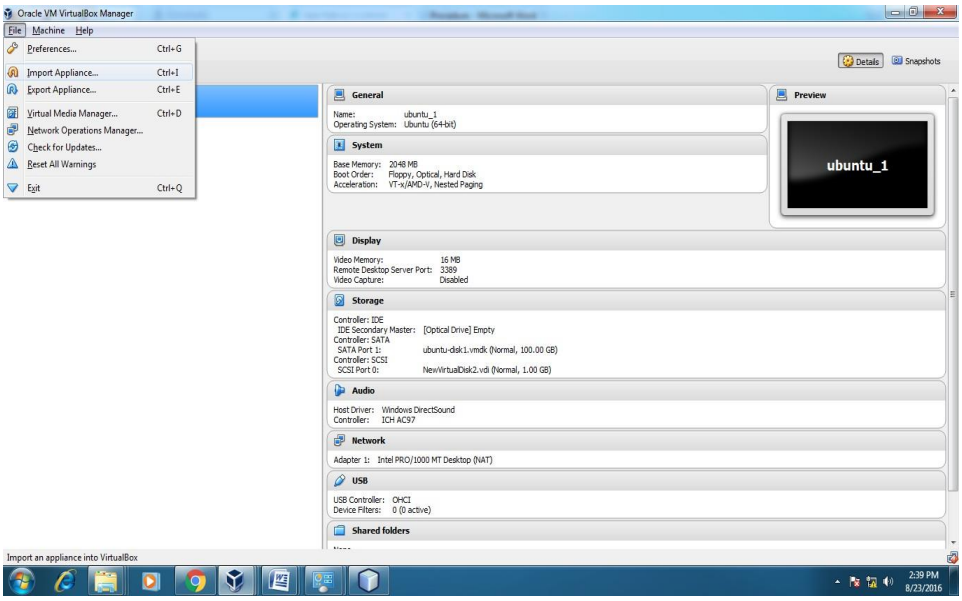
6. Click the next button

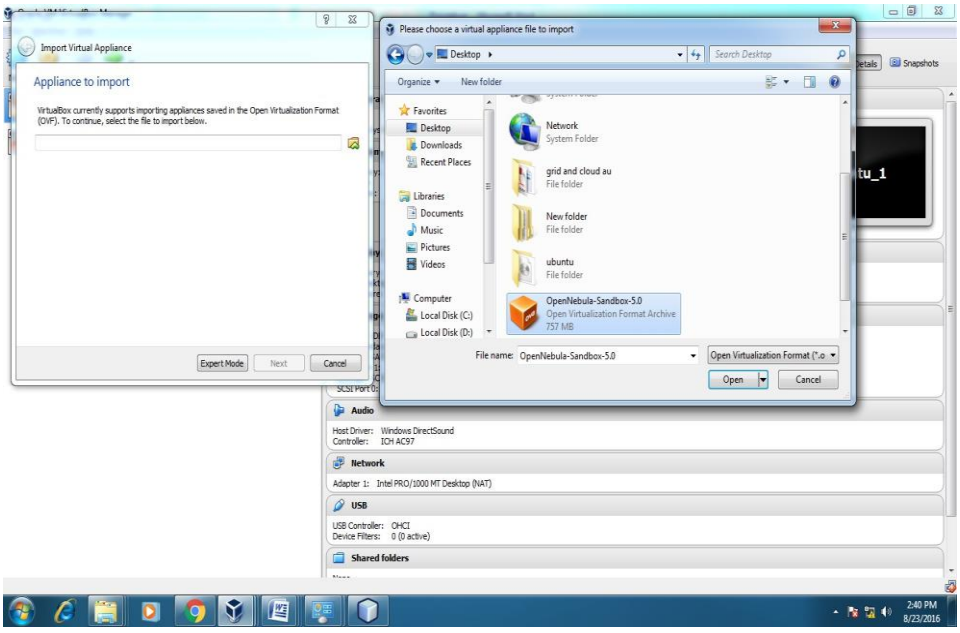


7. Then installation was completed..the show virtual box icon on desktop screen….

Steps to import Open nebula sandbox:

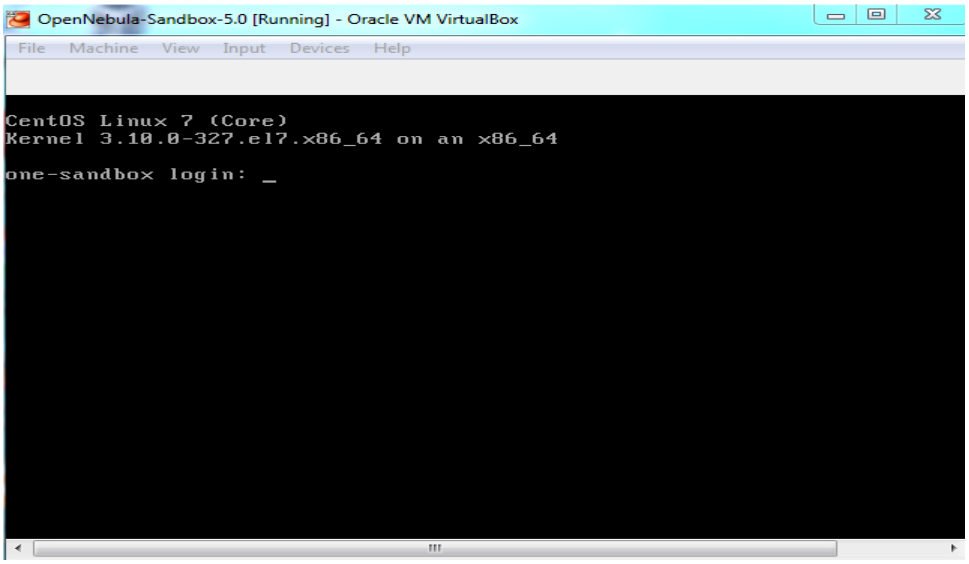
1. Open Virtual box
2. File □import Appliance
3. Browse OpenNebula-Sandbox-5.0.ova file
4. Then go to setting, select Usb and choose USB 1.1
5. Then Start the Open Nebula
6. Login using username: root, password:opennebula



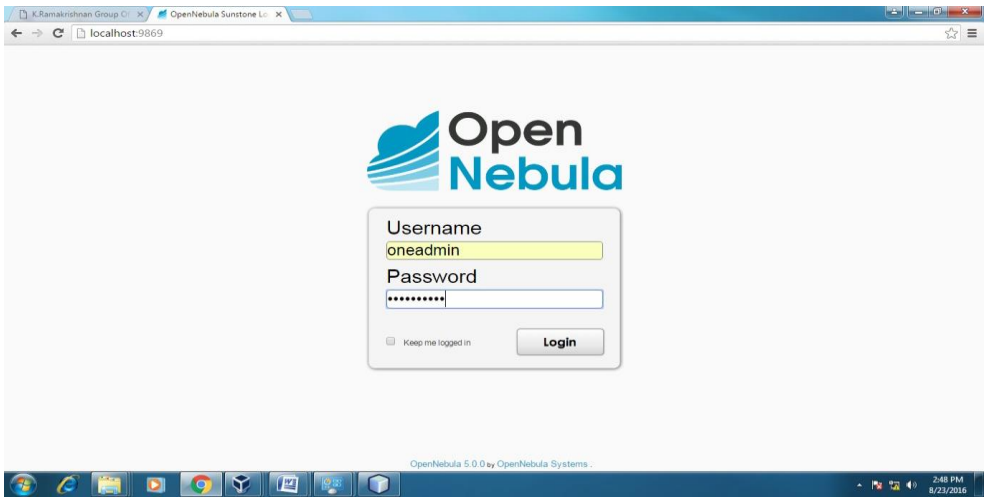


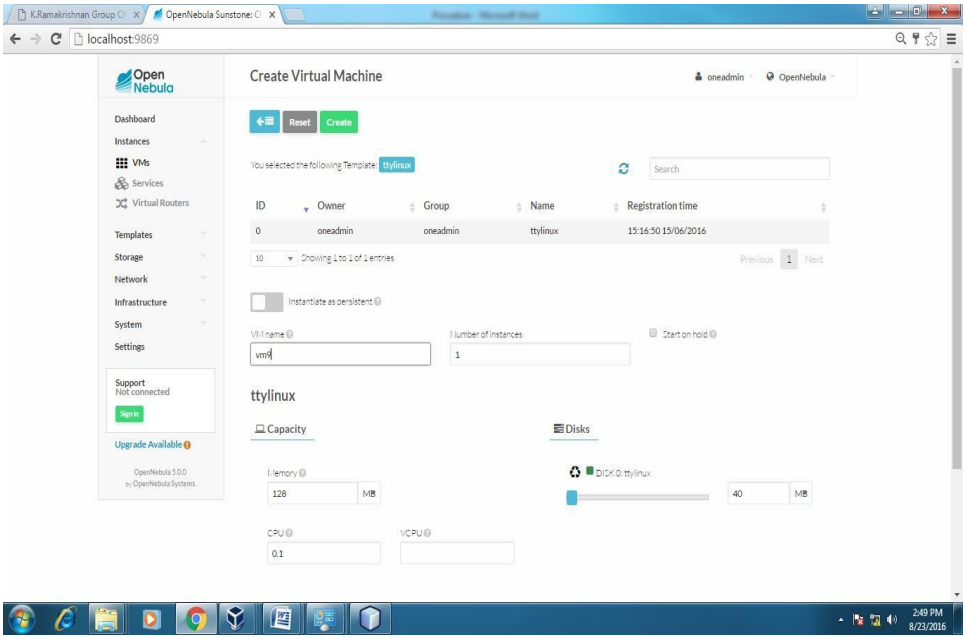
Steps to create Virtual Machine through opennebula

1. Open Browser, type localhost:9869
2. Login using username: oneadmin, password: opennebula
3. Click on instances, select VMs then follow the steps to create Virtaul machine
    a. Expand the + symbol
    b. Select user oneadmin
    c. Then enter the VM name,no.of instance, cpu.
    d. Then click on create button.



    e. Repeat the steps the C,D for creating more than one VMs.





APPLICATIONS:

There are various applications of cloud computing in today's network world. Many search engines and social websites are using the concept of cloud computing like www.amazon.com, hotmail.com, facebook.com, linkedln.com etc. the advantages of cloud computing in context to scalability is like reduced risk , low cost testing ,ability to segment the customer base and auto-scaling based on application load.

RESULT:

Thus the procedure to run the virtual machine of different configuration.

EX NO. : 2 DATE:	**Install a C compiler in the virtual machine created using virtual box and execute Simple Programs**

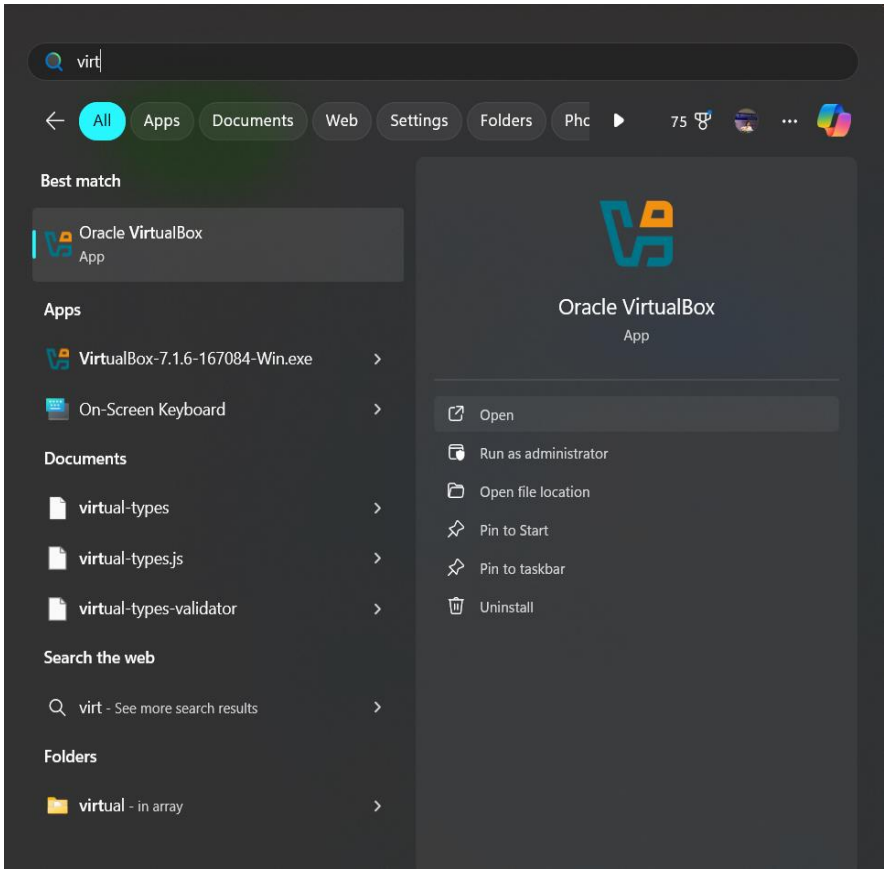
Aim:

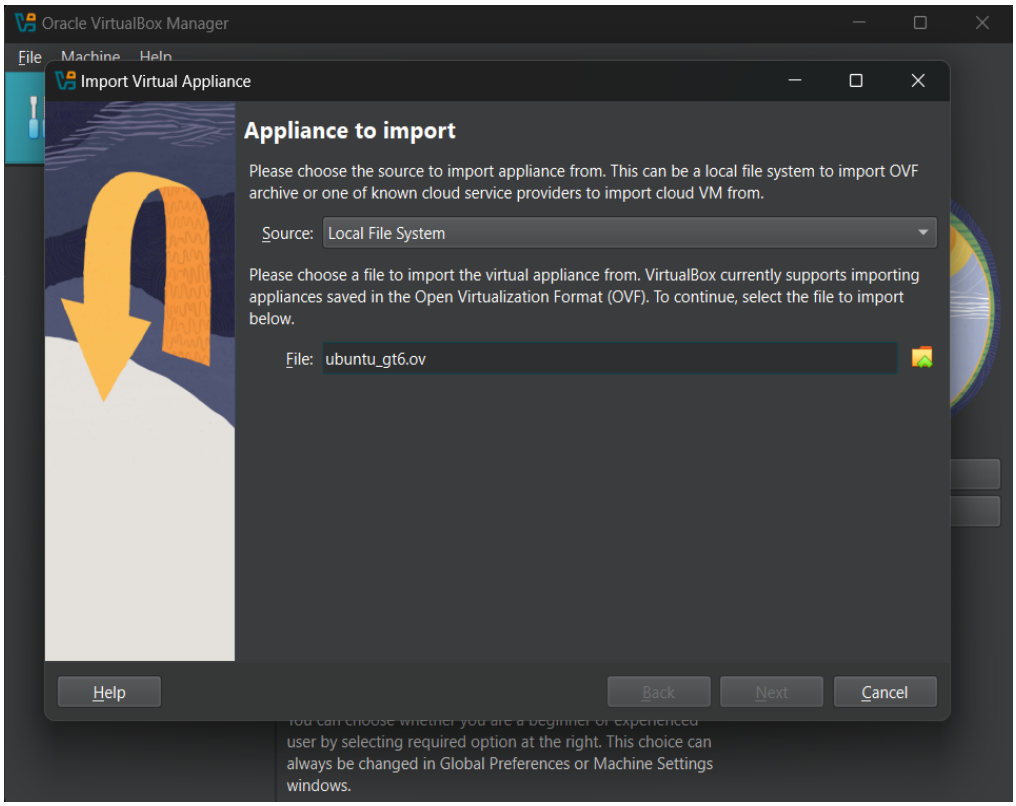
To Install a C compiler in the virtual machine created using virtual box and execute Simple Programs.

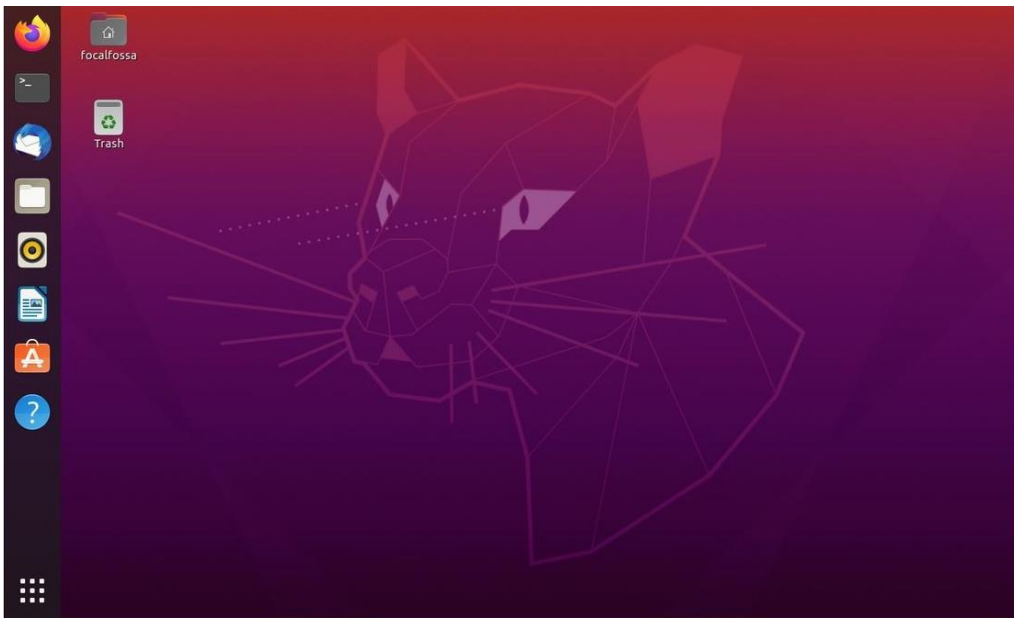
PROCEDURE:

Steps to import .ova file:

1. Open Virtual box
2. File >> import Appliance
3. Browse ubuntu_gt6.ova file
4. Then go to setting, select Usb and choose USB 1.1
5. Then Start the ubuntu_gt6
6. Login using username:Sri, password:76958.



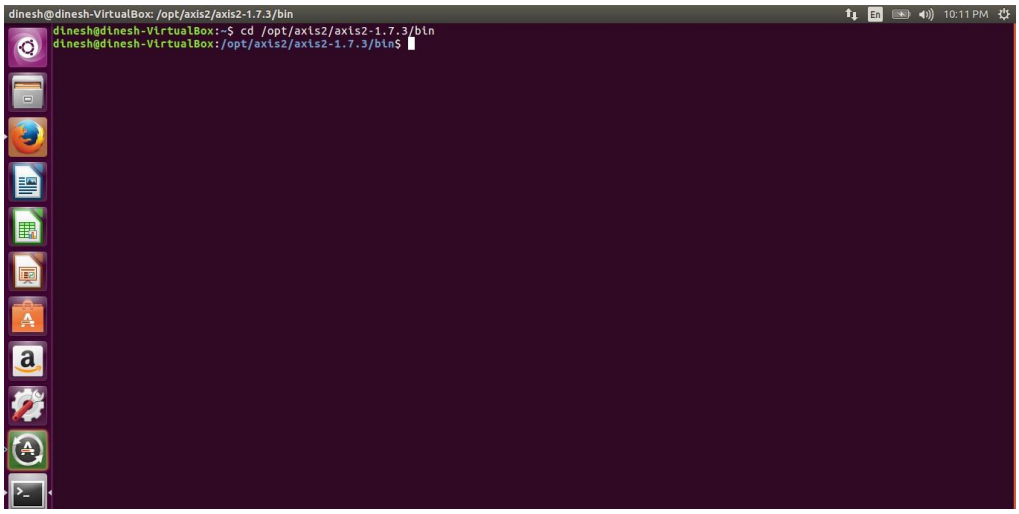

Oracle VirtualBox Manager
File
Machine
Help
Import Virtual Appliance
Appliance to import
Please choose the source to import appliance from. This can be a local file system to import OVF archive or one of known cloud service providers to import cloud VM from.
Source:
Local File System
Please choose a file to import the virtual appliance from. VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.
File:
ubuntu_gt6.ov
Help
Back
Next
Cancel
user by selecting required option at the right. This choice can always be changed in Global Preferences or Machine Settings windows.


focalfossa
Trash

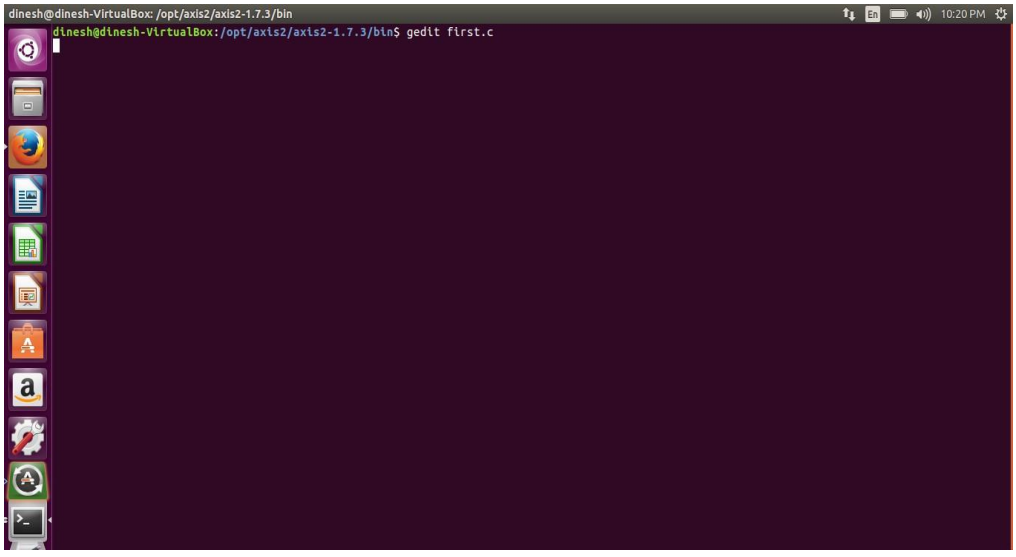
Steps to run c program:

1. Open the terminal
2. Type cd /opt/axis2/axis2-1.7.3/bin then press enter
3. gedit hello.c
4. gcc hello.c
5. ./a.out

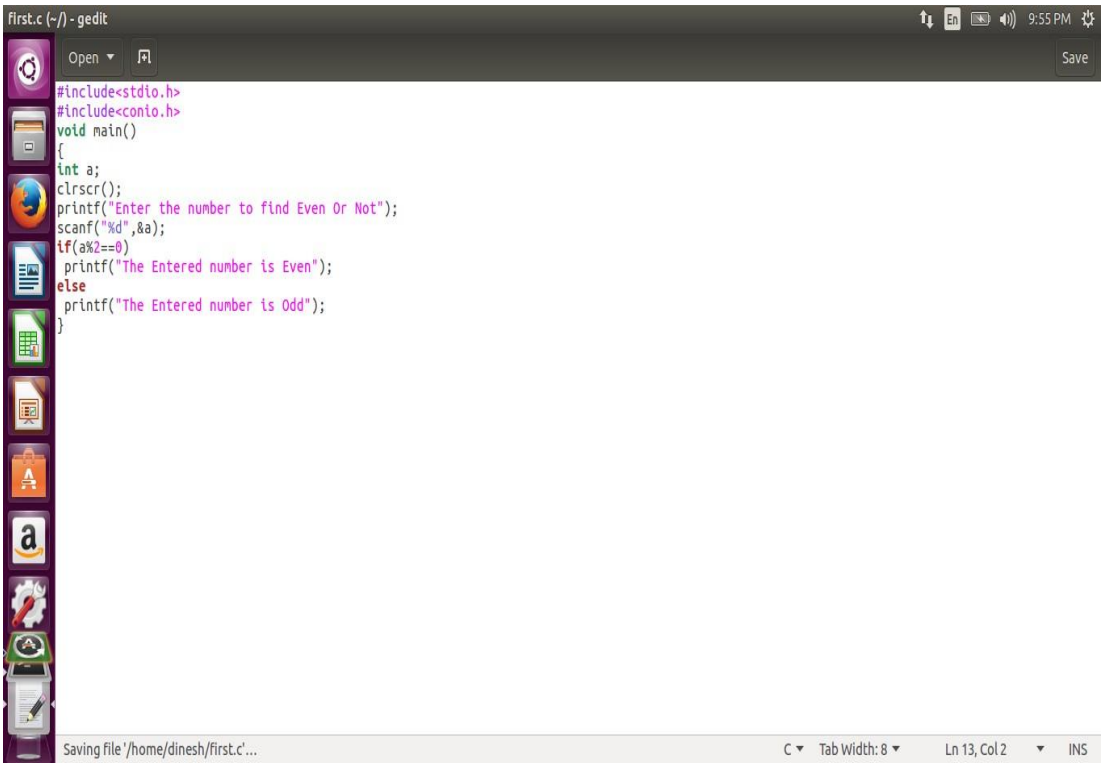
1. Type cd /opt/axis2/axis2-1.7.3/bin then press enter



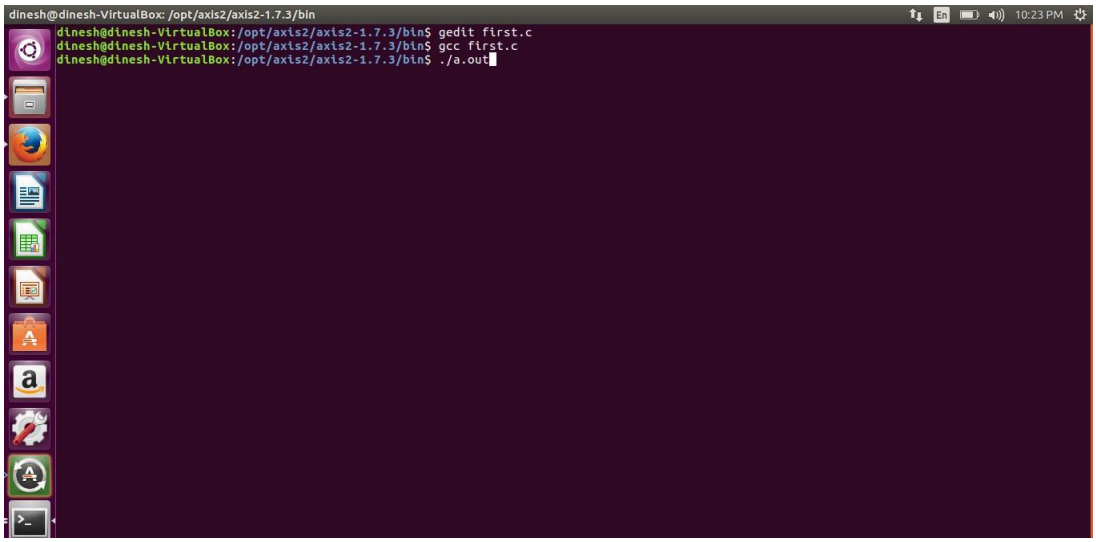
2. Type gedit first.c



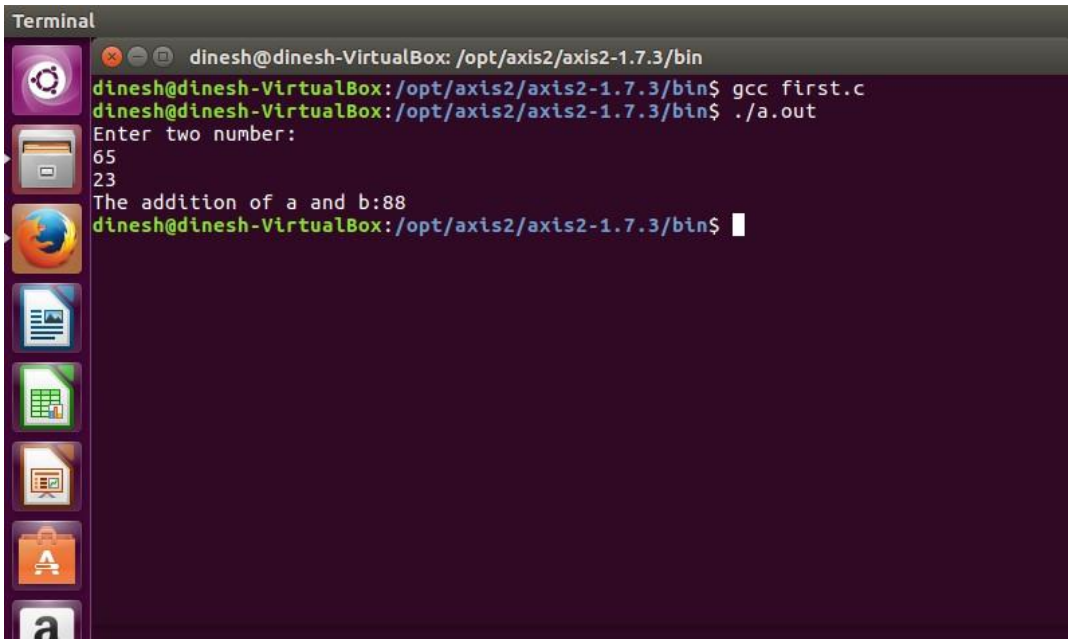
3. Type the c program



4. Running the C program



5. Display the output:



APPLICATIONS:

Simply running all programs in grid environment.

RESULT:

Thus the simple C programs executed successfully in VirtualBox .

Ex no:3 Date:	**3.Install Google App Engine. Create *hello world* app and other simple web applications using python/java**.

Aim:

To install google app engine . create hello world app and other simple web applications using python/java.

Install Google App Engine (GAE)

First, GAE is part of **Google Cloud**. So installing GAE basically means setting up **Google Cloud SDK**.

Install Google Cloud SDK:

- Download: https://cloud.google.com/sdk/docs/install
- After installing:

gcloud init

This will:

- Authenticate your account
- Set project and region

Now you have **GAE tools** ready.

Procedure for creating hello world app:

1.Create a new folder:

mkdir hello-python

cd hello-python

2.Create 3 files:

- main.py
- requirements.txt
- app.yaml

I. main.py

python

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello():
    return 'Hello, World from Python GAE!'

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080, debug=True)
```

II. requirements.txt

```
Flask
```

III. app.yaml

yaml

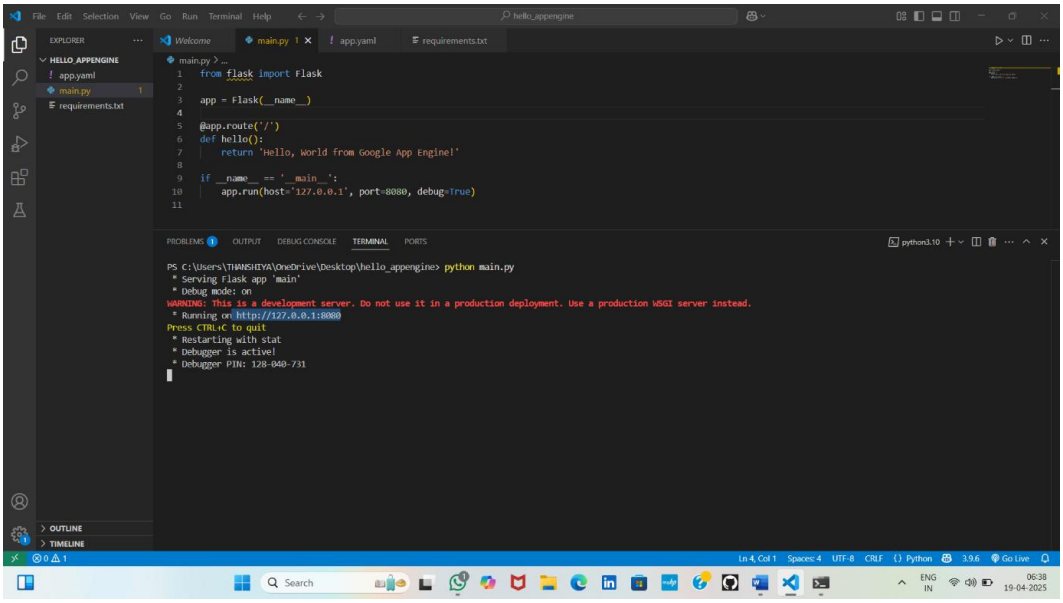
```
runtime: python39
entrypoint: gunicorn -b :$PORT main:app
```

3.Install dependencies:

```
pip install -r requirements.txt
```

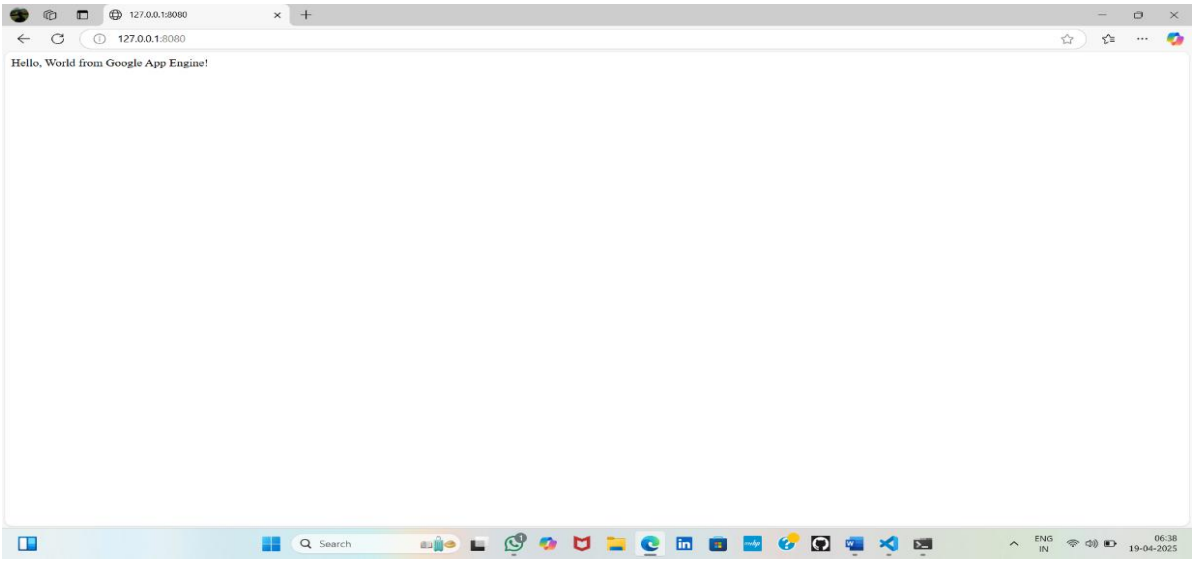
4.**Deploy to App Engine:**

```
gcloud app create
gcloud app deploy
```



5.**After deployment:**

gcloud app browse



Result:

Thus the installation of google app engine and the creation of hello world app using python/java has been successfully executed and output was verified.

Ex no:4 Date:	4. Use GAE launcher to launch the web applications

Aim:

To use GAE launcher to launch the web applications.

Procedure:

Installation Process:

Pre-Requisites: Python 2.7

If you don't already have Python 2.7 installed in your computer, download and install Python 2.7 from: https://www.python.org/download/releases/2.7/

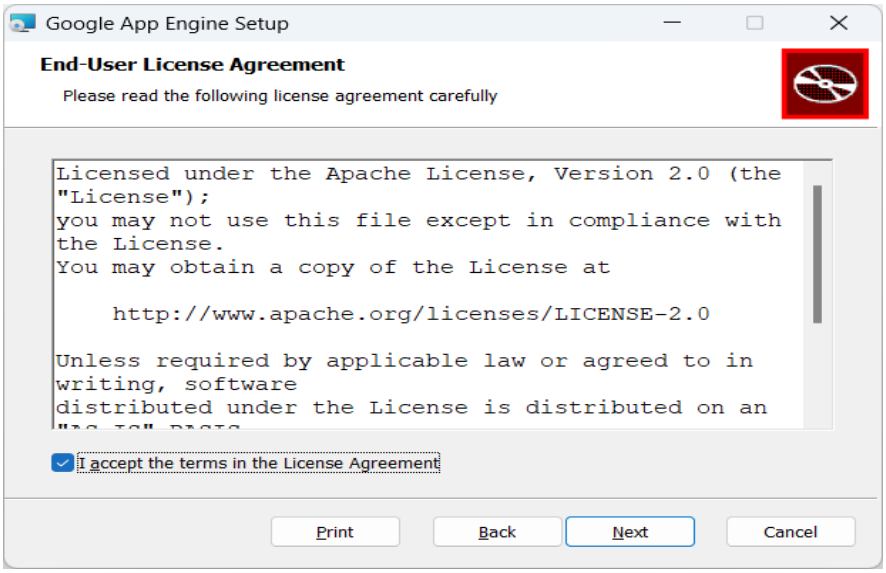
1. Download the Google App Engine (GAE) Launcher

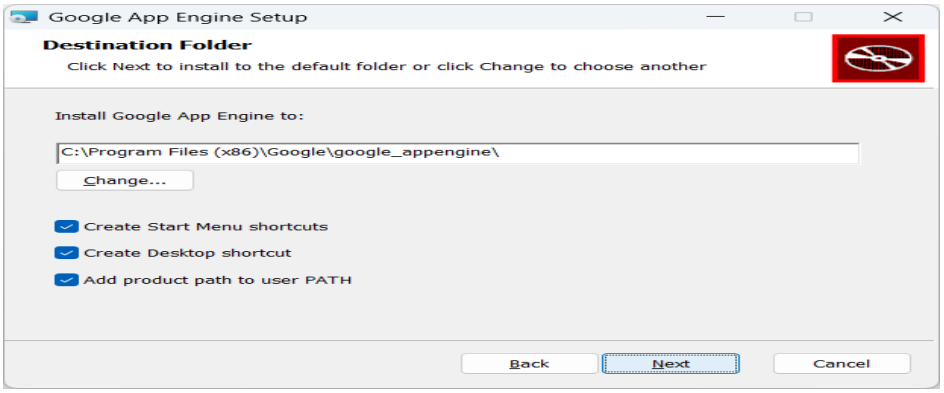
   using the following link

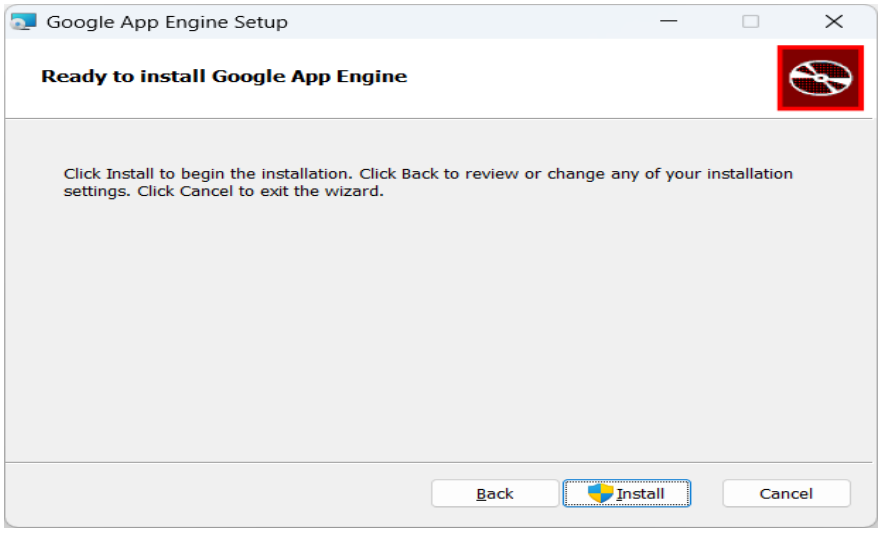
   https://www.npackd.org/p/com.google.AppEnginePyt

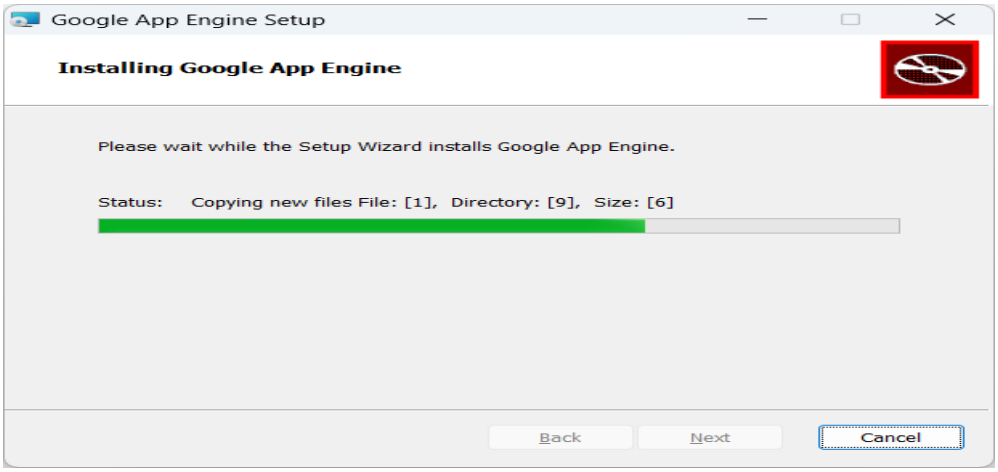
   honSDK/1.9.62

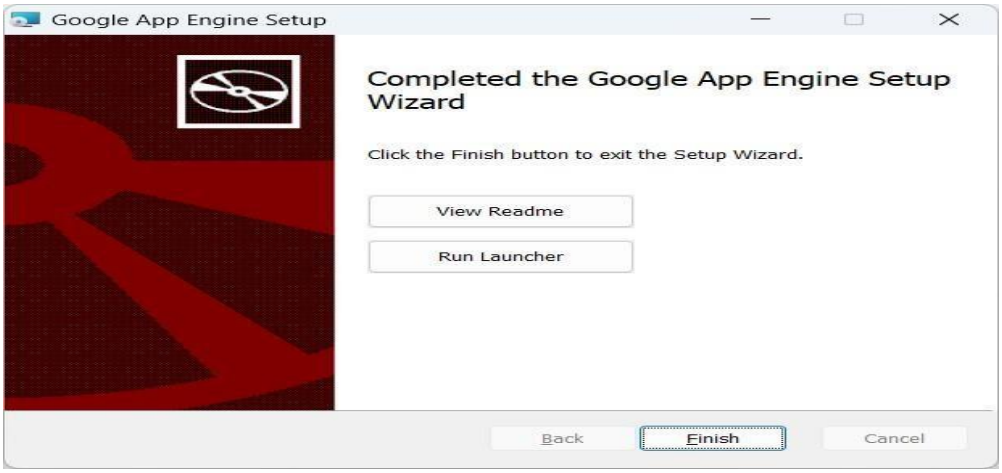
2. Install the Google App Engine Launcher using the following steps.


Google App Engine Setup
End-User License Agreement
Please read the following license agreement carefully
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an
I accept the terms in the License Agreement
Print
Back
Next
Cancel

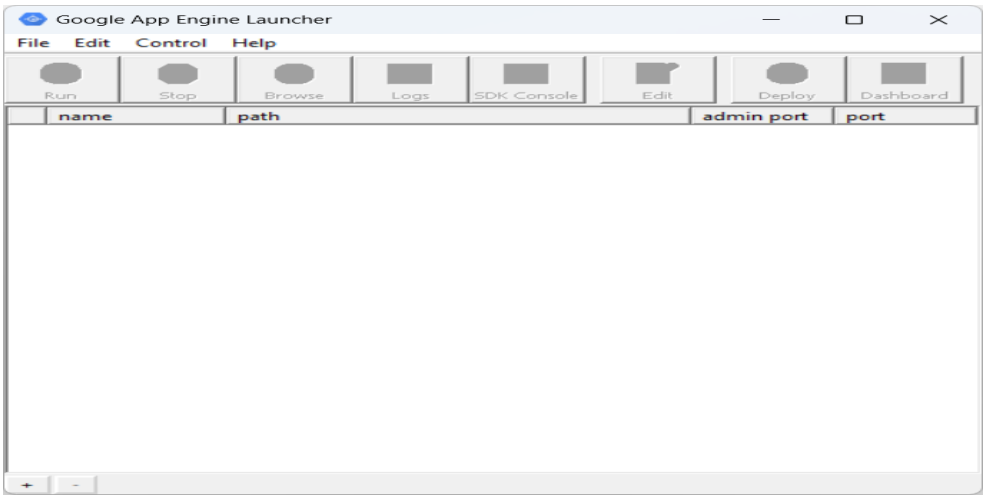

Google App Engine Setup
Destination Folder
Click Next to install to the default folder or click Change to choose another
Install Google App Engine to:
C:\Program Files (x86)\Google\google_appengine\
Change...
Create Start Menu shortcuts
Create Desktop shortcut
Add product path to user PATH
Back
Next
Cancel


Google App Engine Setup
Ready to install Google App Engine
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel



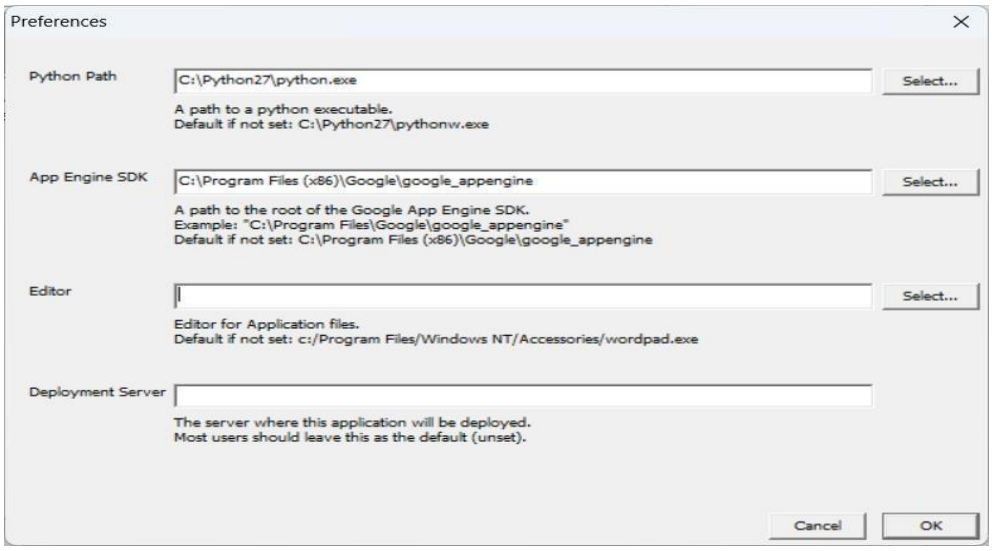


3. Click Run Launcher, the Google App Engine Launcher will open.

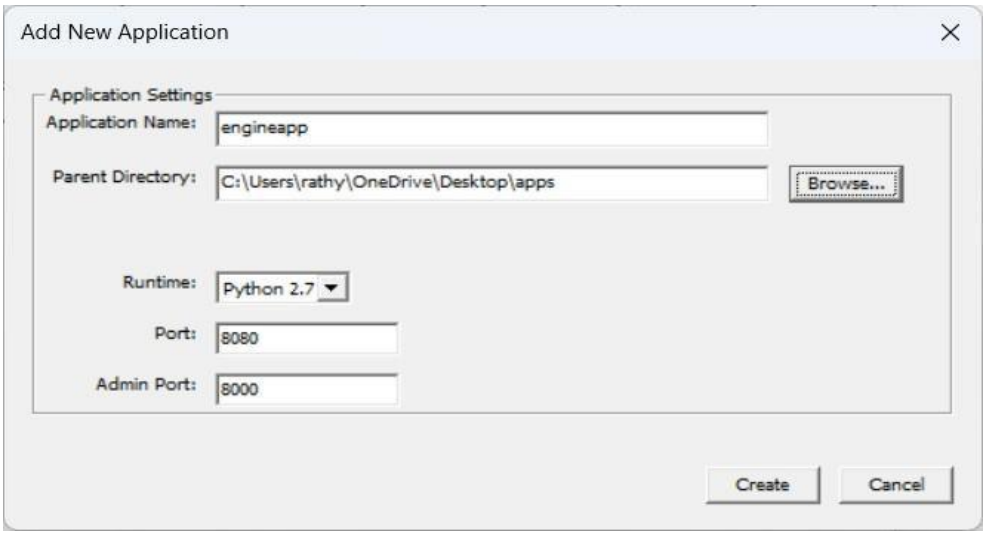


Making Your Applications:

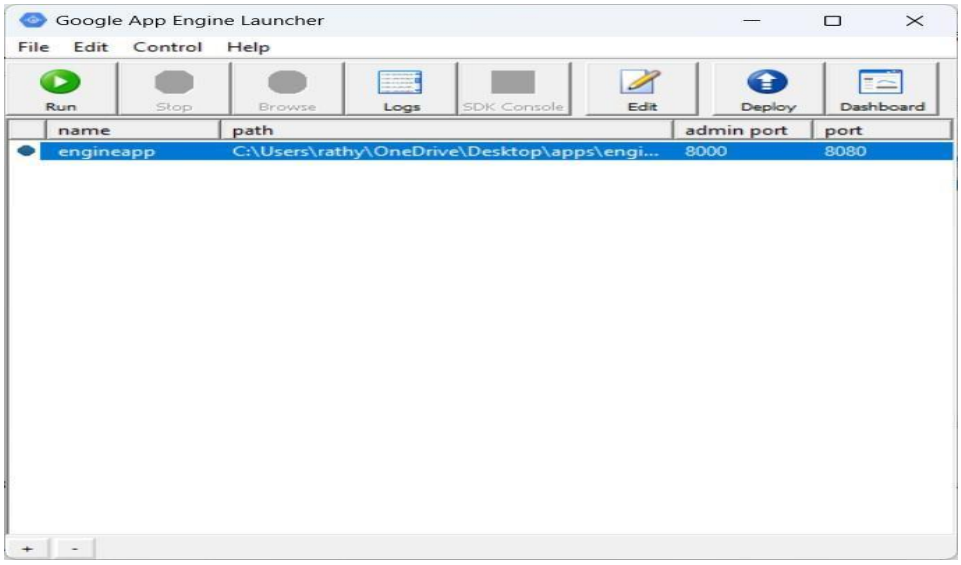
1. To set the paths for python and Google Application Engine, In GAE Launcher menu, Click Edit ->Preferences.



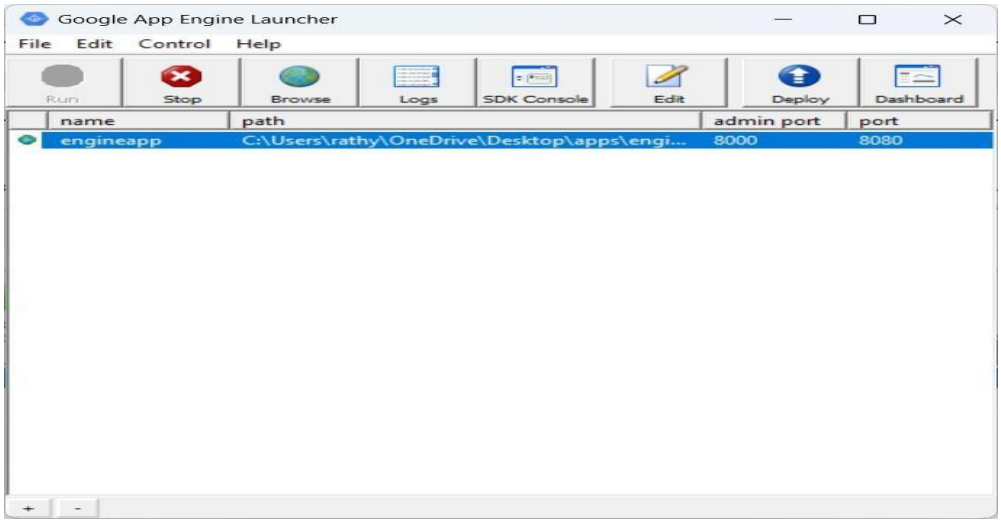
2. Create one folder in Desktop. (E.g., apps)
3. In Google App Engine Launcher Menu, Click File-> Create New Application, and pick a parent directory for the new applications, then click Create.



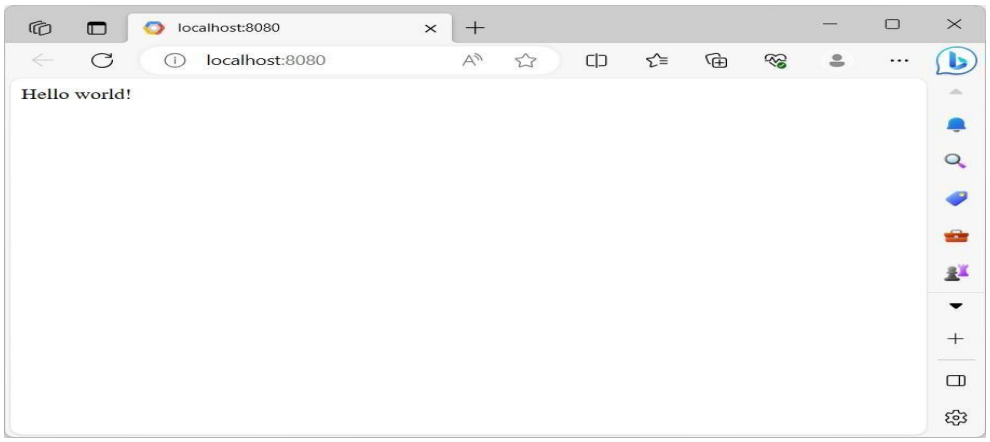
4. Once you have created the new application, the following files (app.yaml, favicon.ico, index.yaml, main.py) are automatically created in your apps folder.
5. Select the created application, so that you can control the application using the launcher.



6. Once you have selected your application and click **Run**. After a few moments your application will start and the launcher will show a little green icon next to your application.

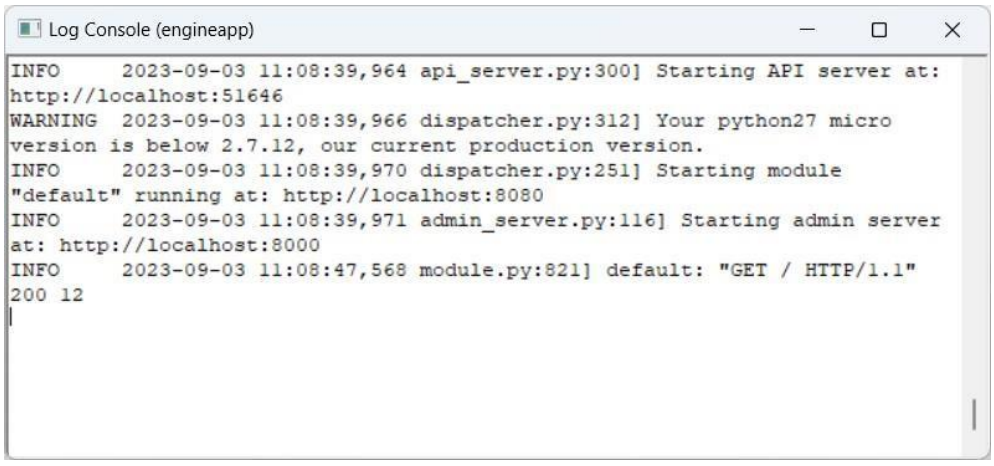


7. Then click **Browse** to open a browser pointing at your application which is running at **http://localhost:8080/ or** type **http://localhost:8080** into your browser and you should see your application as follows:



Watching the Log

You can watch the internal log of the actions that the web server is performing when you are interacting with your application in the browser. Select your application in the Launcher andpress the **Logs** button to bring up a log window:



Each time you press **Refresh** in your browser – you can see it retrieving the output with a **GET** request.

Shutting Down the Server

To shut down the server, use the Launcher, select your application and press the **Stop** button.

Result:

Thus, the installation of GAE launcher was launched and verified successfully.

Ex No. 5 Date:	Simulate a cloud scenario using CloudSim and running a scheduling algorithm that is not present in cloud sim

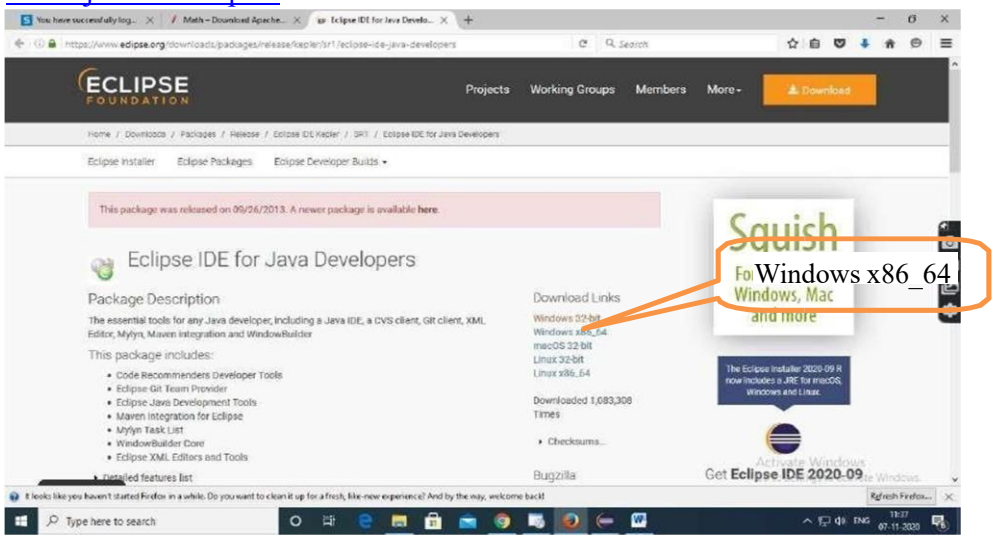
AIM:

To simulate a cloud scenario using cloud sim and run a scheduling algorithm that is

not present in Cloud sim

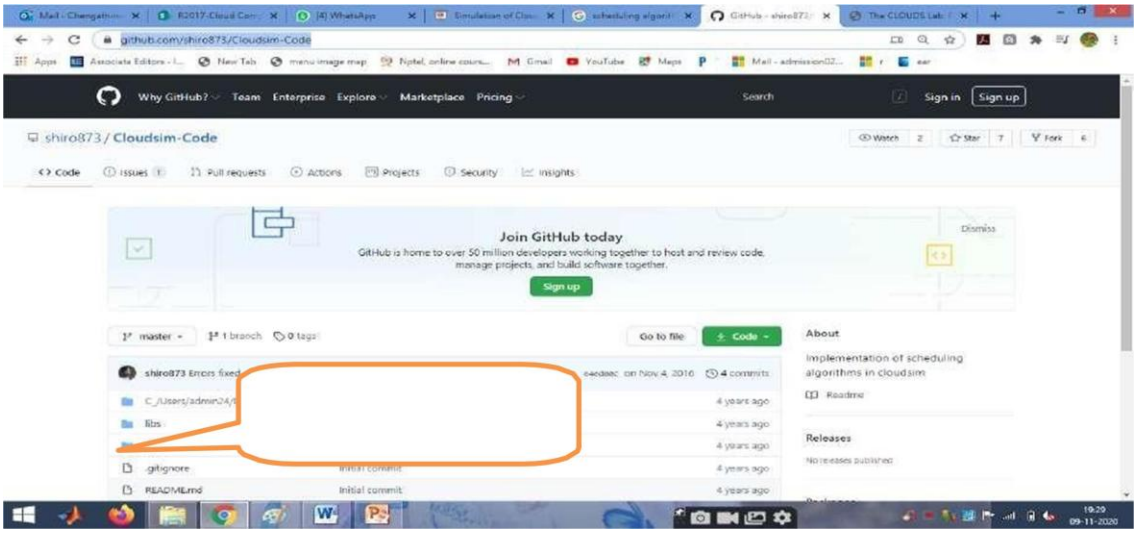
Step 1: Link to download Eclipse and download Eclipse for Windows 64bit into your Local machine

https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide- java-developers



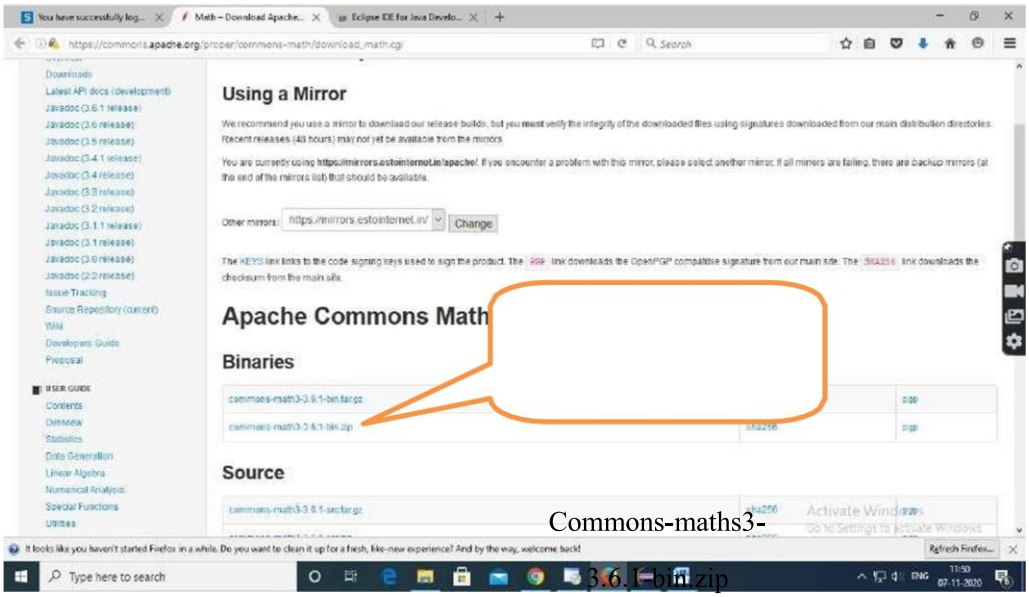
Step 2: Download scheduling source code cloudsim-code-master from git hub repository in your local machine

https://github.com/shiro873/Cloudsim-Code

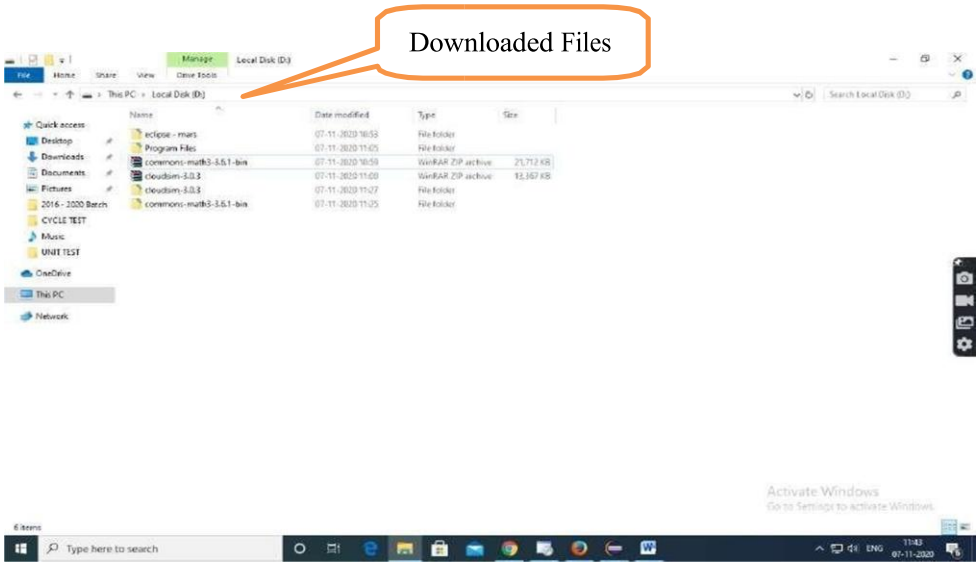


Step 3: Download commons-maths3-3.6.1 from git hub repository in your local machine

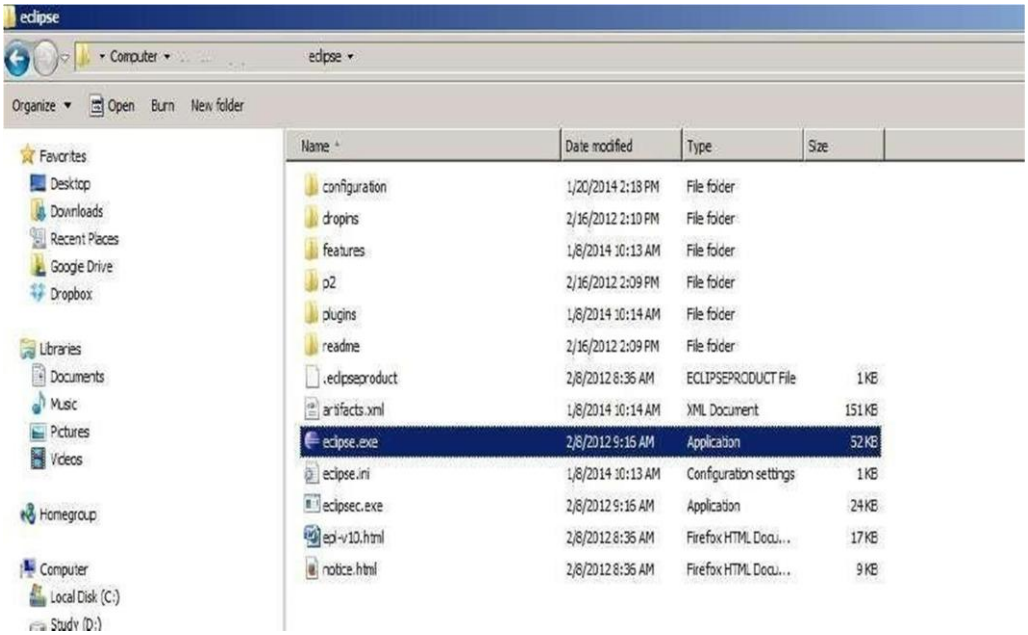
https://commons.apache.org/proper/commonsmath/download_math.cgi



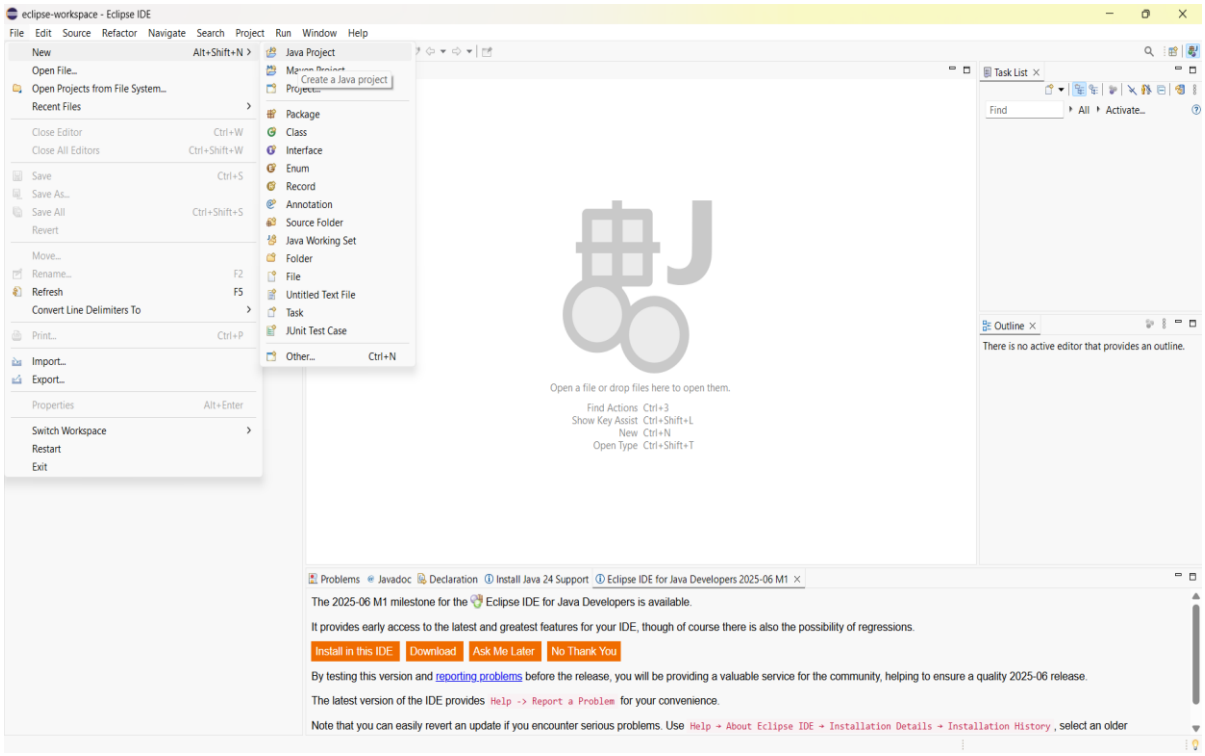
Step 4: Downloaded Eclipse, cloudsim-3.0.3 and Apache Commons Math 3.6.1 in your local machine and extract cloudsim-3.0.3 and Apache Commons Math 3.6.1



Step 5: First of all, navigate to the folder where you have unzipped the eclipse folder and open Eclipse.exe

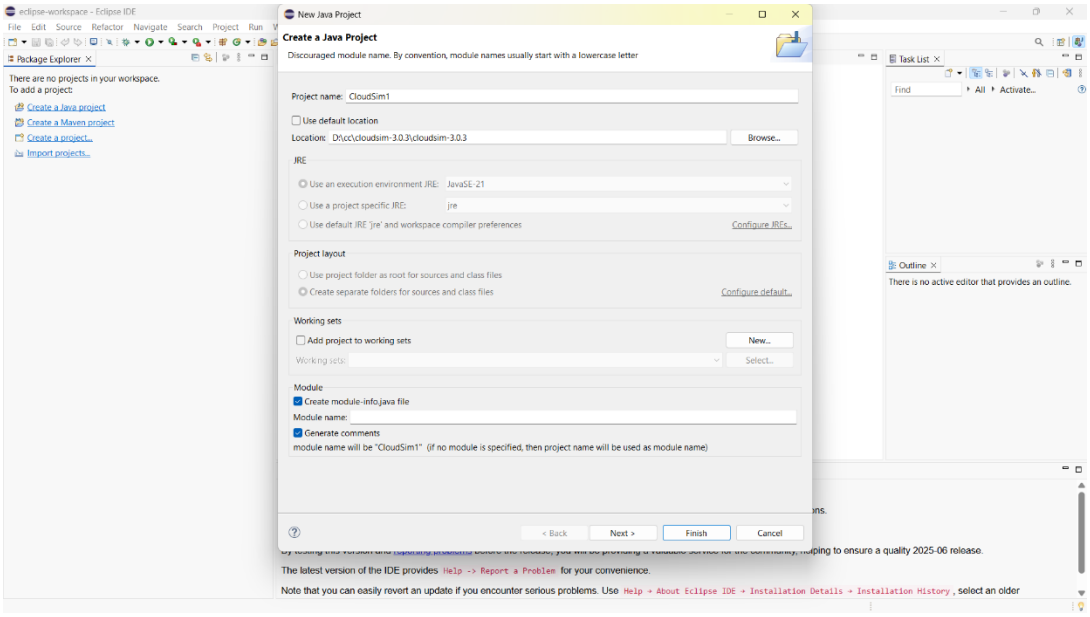


Step 6: Now within Eclipse window navigate the menu: File -> New -> Project, to open the new project wizard

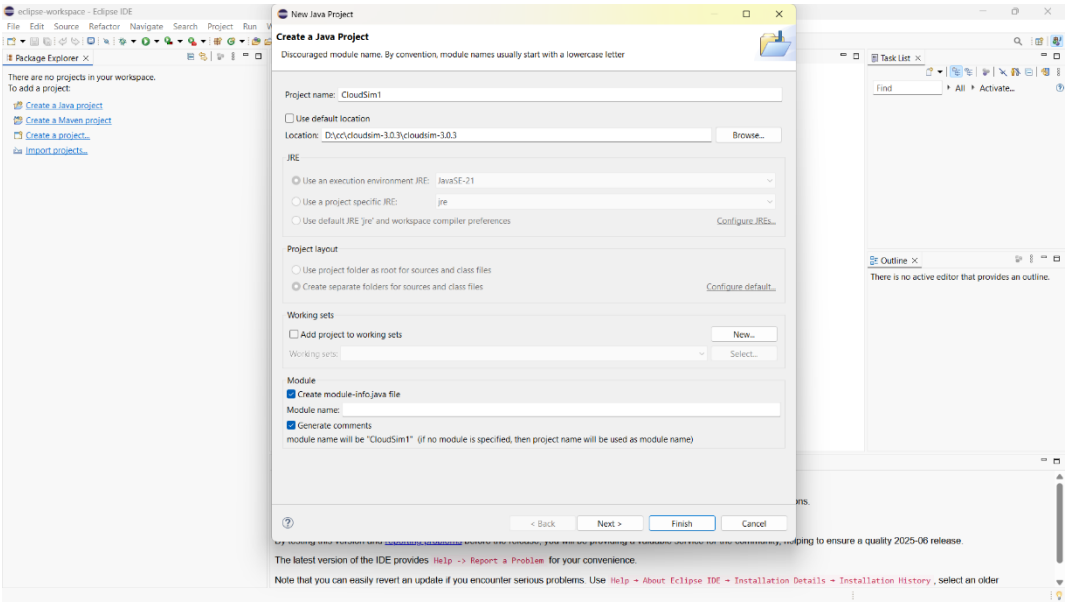


Step 8: Now a detailed new project window will open, here you will provide the project name and the path of CloudSim-master-code project source code, which will be done as follows:

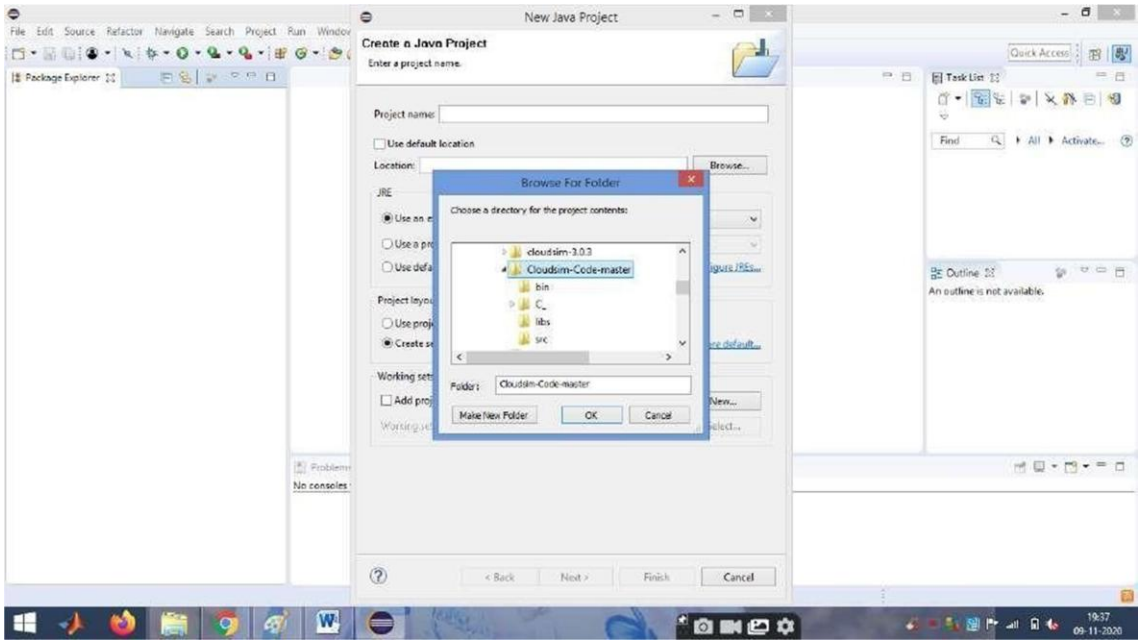
Project Name: CloudSim



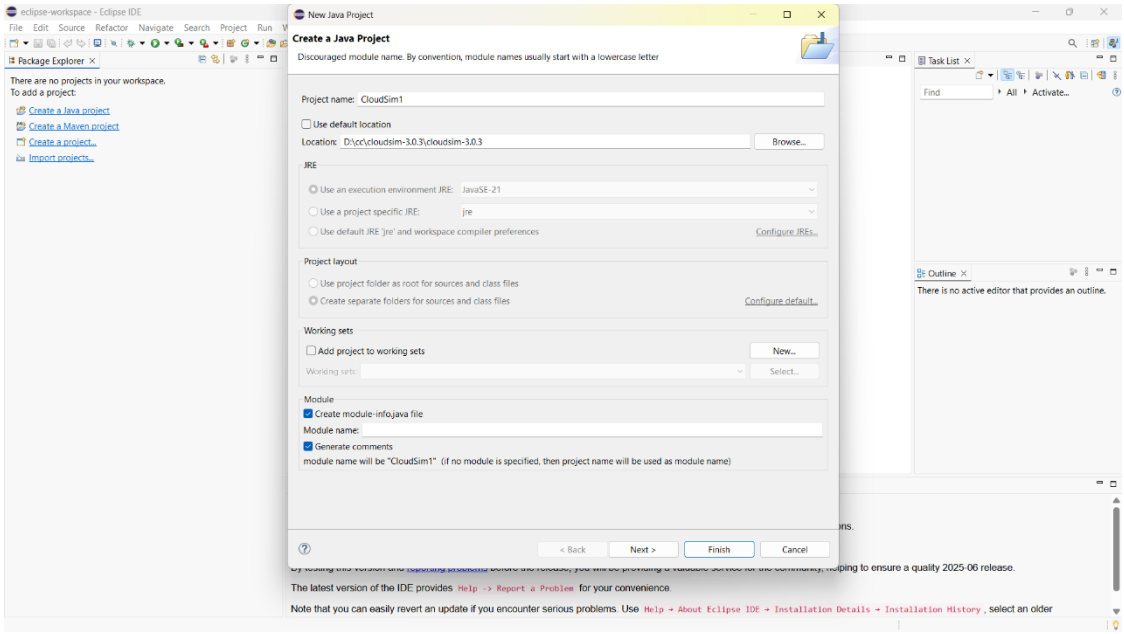
Step 9: Unselect the ‘Use default location’ option and then click on ‘Browse’ to open the path where you have unzipped the Cloudsim-code-master project and finally click Next to set project settings.



Step 10: Make sure you navigate the path till you can see the bin, docs, examplesetc folder in the navigation plane.



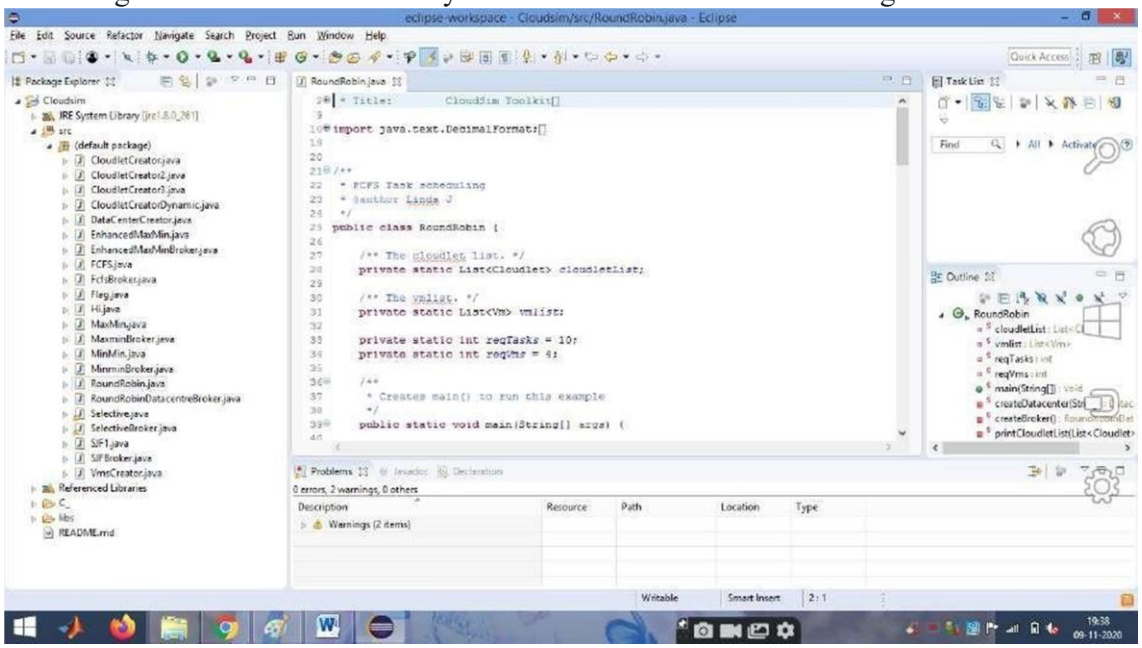
Step 11: Once done finally, click Next‘ to go to the next step i.e. setting up of project settings



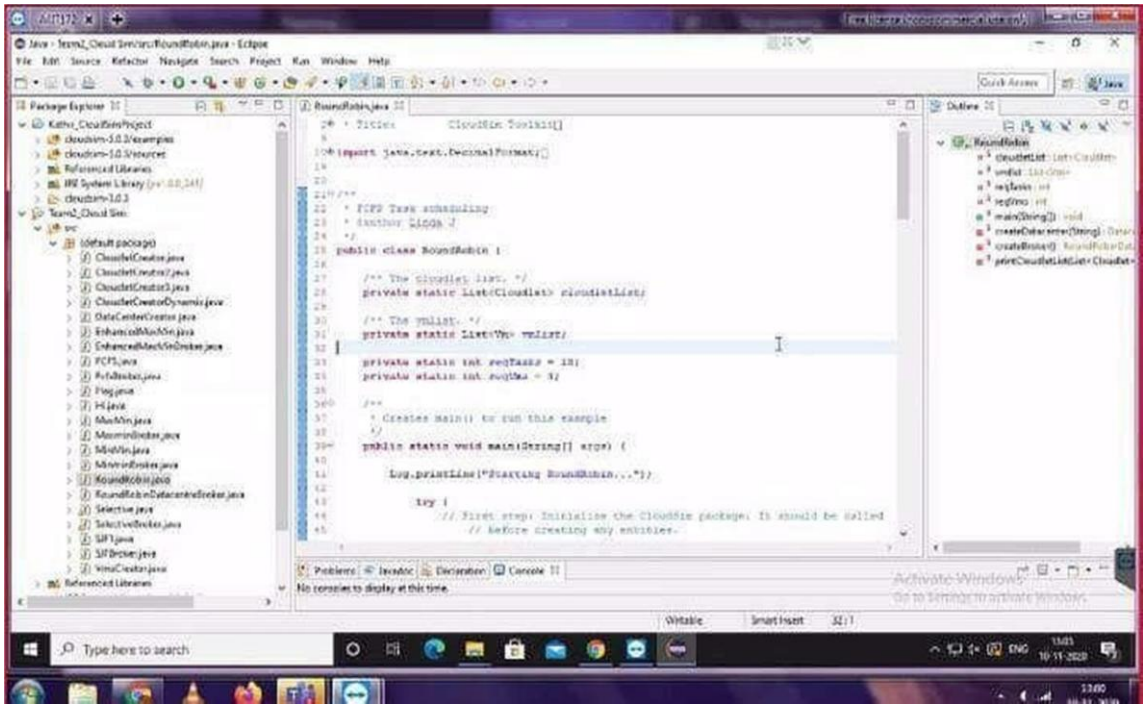
Step 12: Once the project is configured you can open the Project Explorer and start exploring the Cloudsim project. Also for the first time eclipse automatically start

building the workspace for newly configured Cloudsim project, which may take some time depending on the configuration of the computer system.

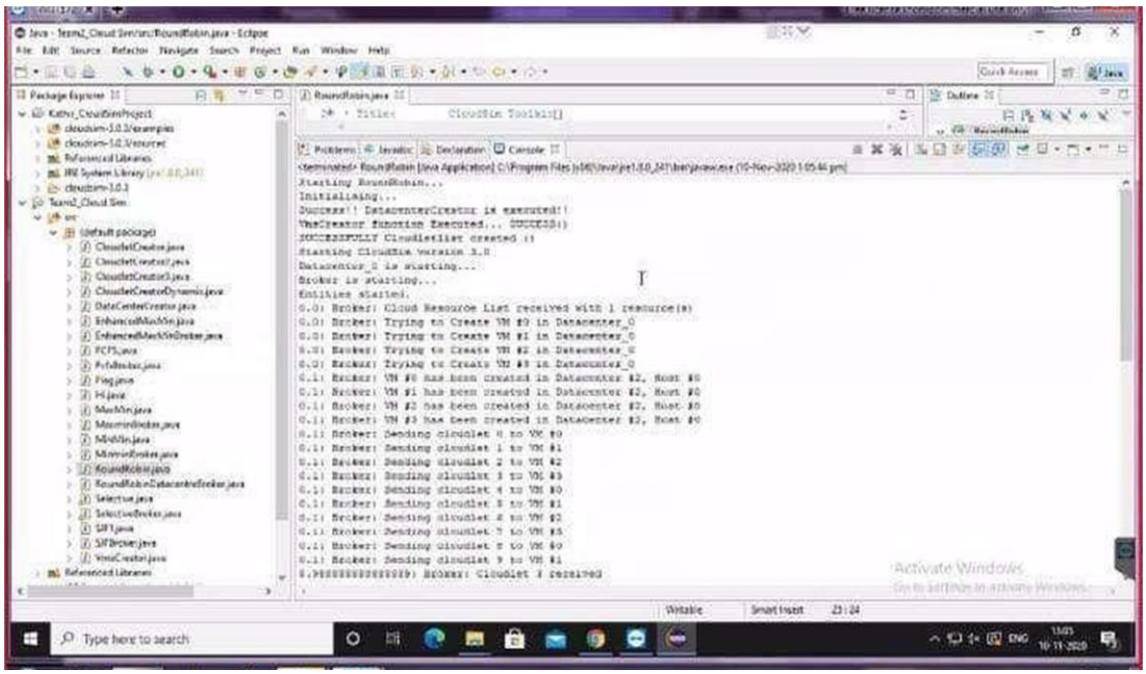
Following is the final screen which you will see after Cloudsim is configured.

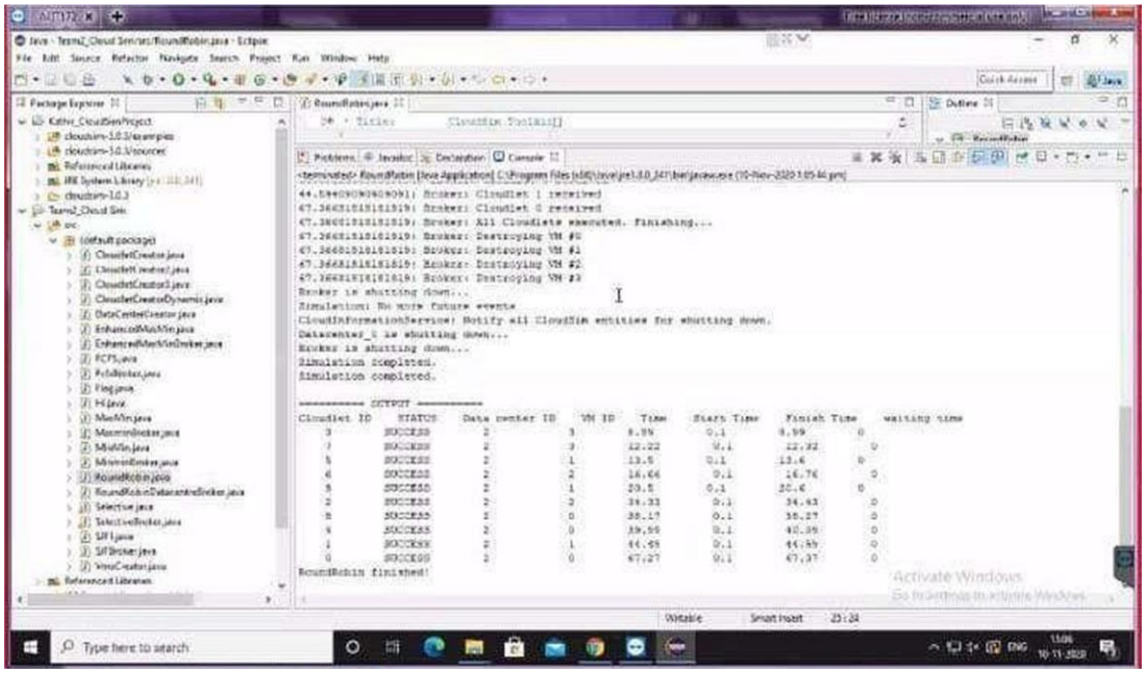


Step 13: Now just to check you within the Project Explorer, you should navigate to the src folder, then expand the package default package and double click to open the RoundRobin.java.



Step 14: Now navigate to the Eclipse menu _Run ->Run or directly use a keyboard shortcut ‘Ctrl + F11’ to execute the ‘RoundRobin.java‘. If it is successfully executed it should be displaying the following type to output in the console window of the Eclipse IDE.





Result:

Thus the scheduling algorithm is executed in cloudsim is simulated using Eclipse Environment successfully.

EX NO: 6 DATE:	**Find a procedure to transfer the files from one virtual machine to another virtual machine**

Aim:

To Find a procedure to transfer the files from one virtual machine to another virtual machine

Procedure:

Prerequisites

- Both VMs are powered on.
- Both are connected to the same network (e.g., NAT Network, Bridged, or same VPC if on cloud).
- You have SSH access between them.

Step-by-Step Using SCP

1. Find the IP Address of Destination VM

On the destination VM, run:

```
ip a
```

or

```
ifconfig
```

Look for the IP address (e.g., 192.168.56.102).

2. Ensure SSH is Installed and Running

On the destination VM:

```
sudo systemctl start ssh
sudo systemctl enable ssh
```

If SSH is not installed:

```
sudo apt install openssh-server    # For Debian/Ubuntu
```

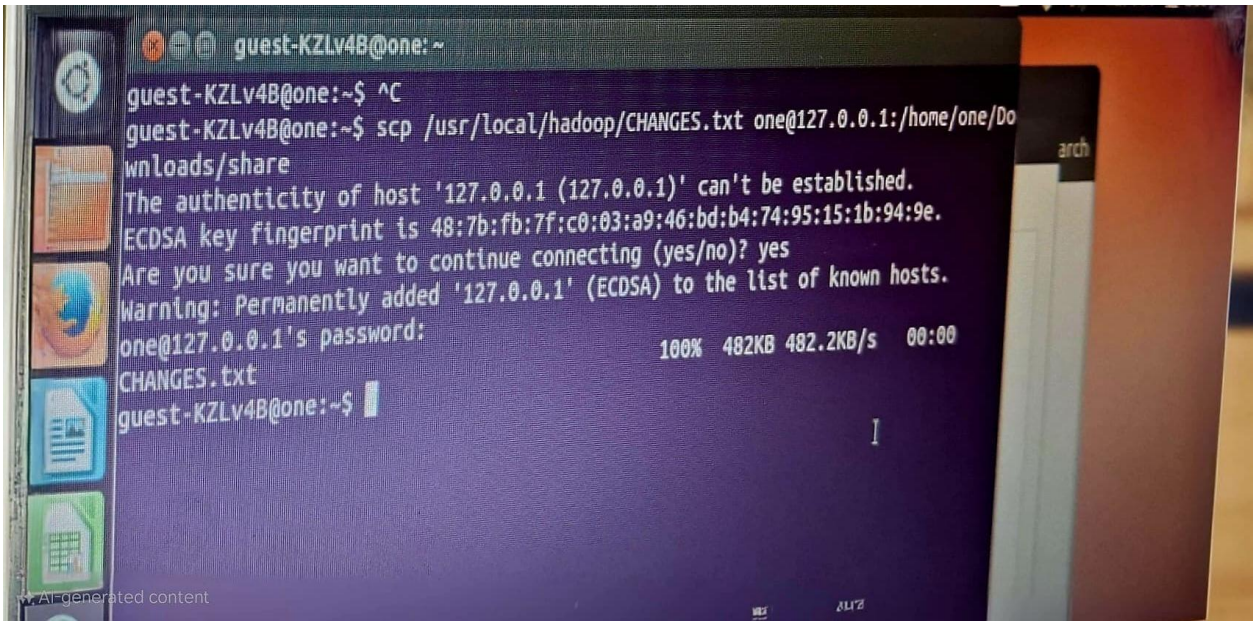
3. Use SCP to Transfer the File

On the source VM, use this command:

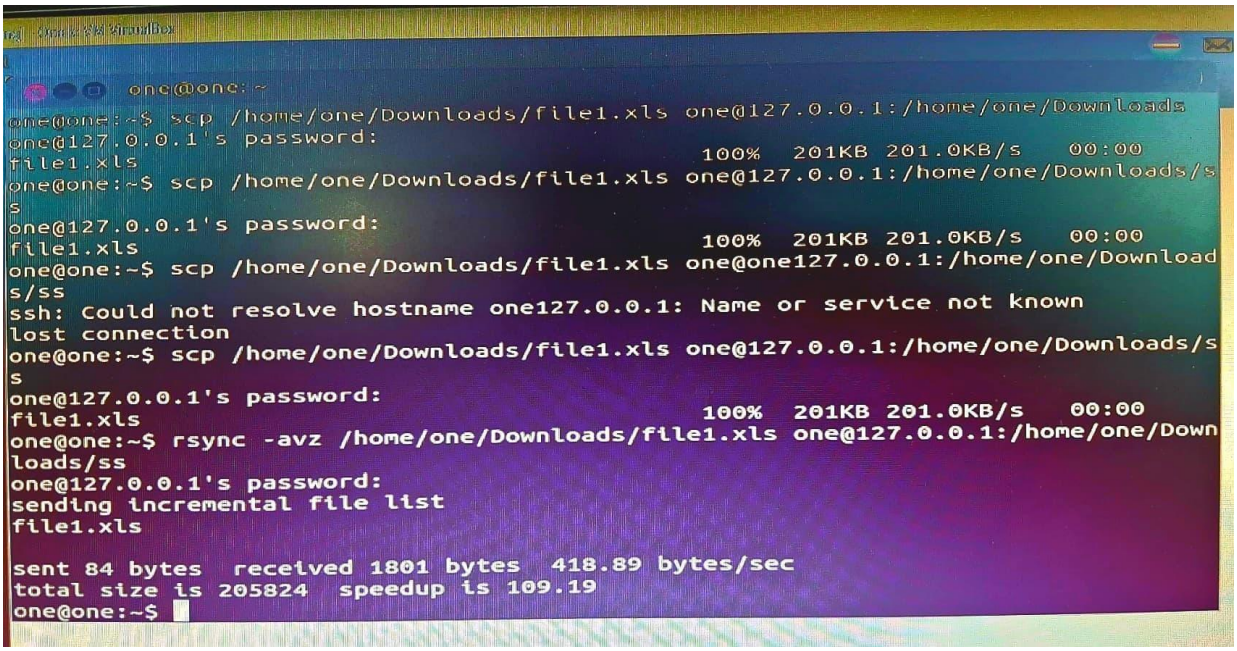
```
scp /path/to/file username@destination_vm_ip:/path/to/destination
```

Example:

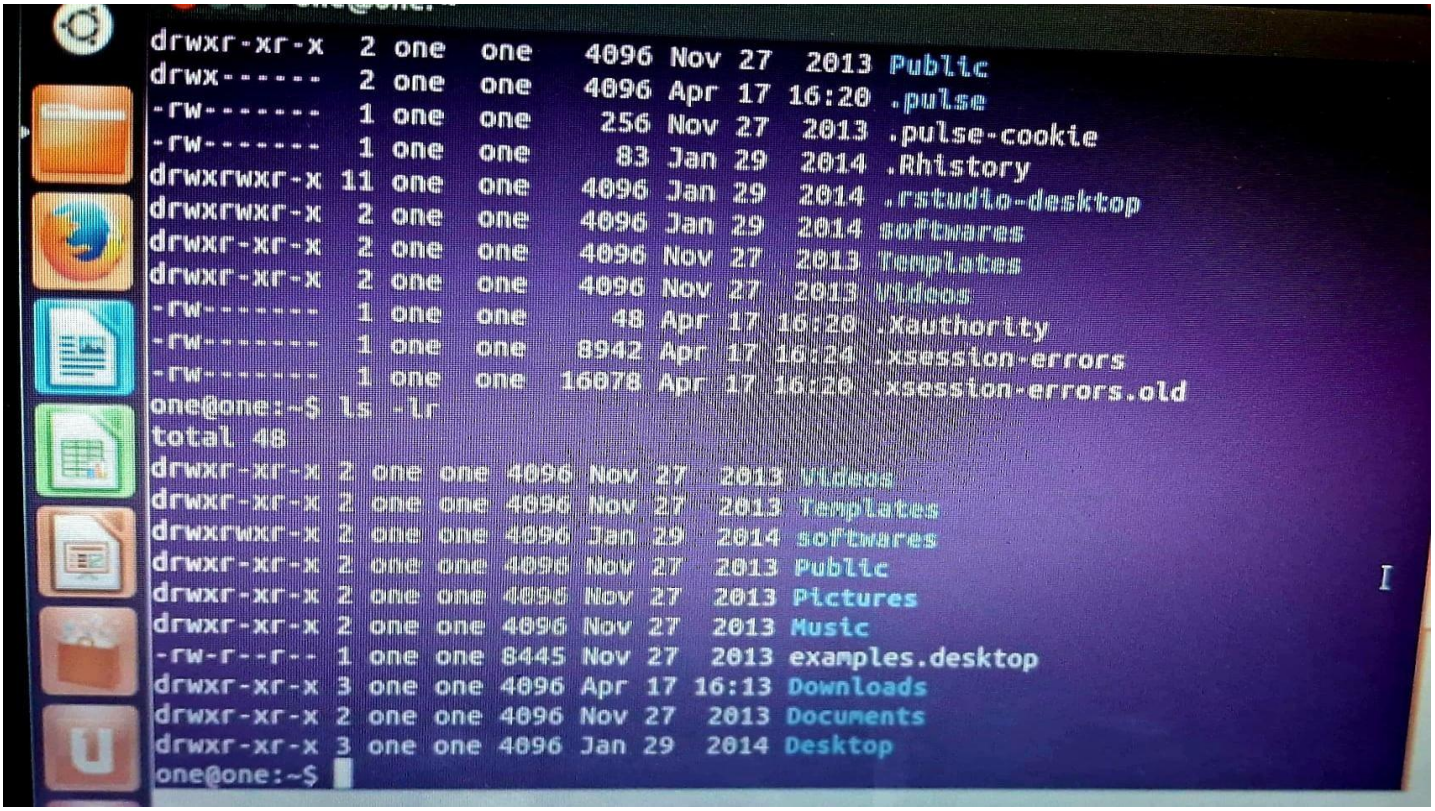
scp /home/user/data.txt user2@192.168.56.102:/home/user2/



You'll be prompted for the password of user2 on the destination VM.



If the file is not visible use ls -lr command



RESULT:

Thus a procedure to transfer the files from one virtual machine to another virtual machine has been found and implemented successfully .

EX NO: 7 DATE:	INSTALL HADOOP SINGLE NODE CLUSTER AND RUN SIMPLE APPLICATION LIKE WORD COUNT

Aim :

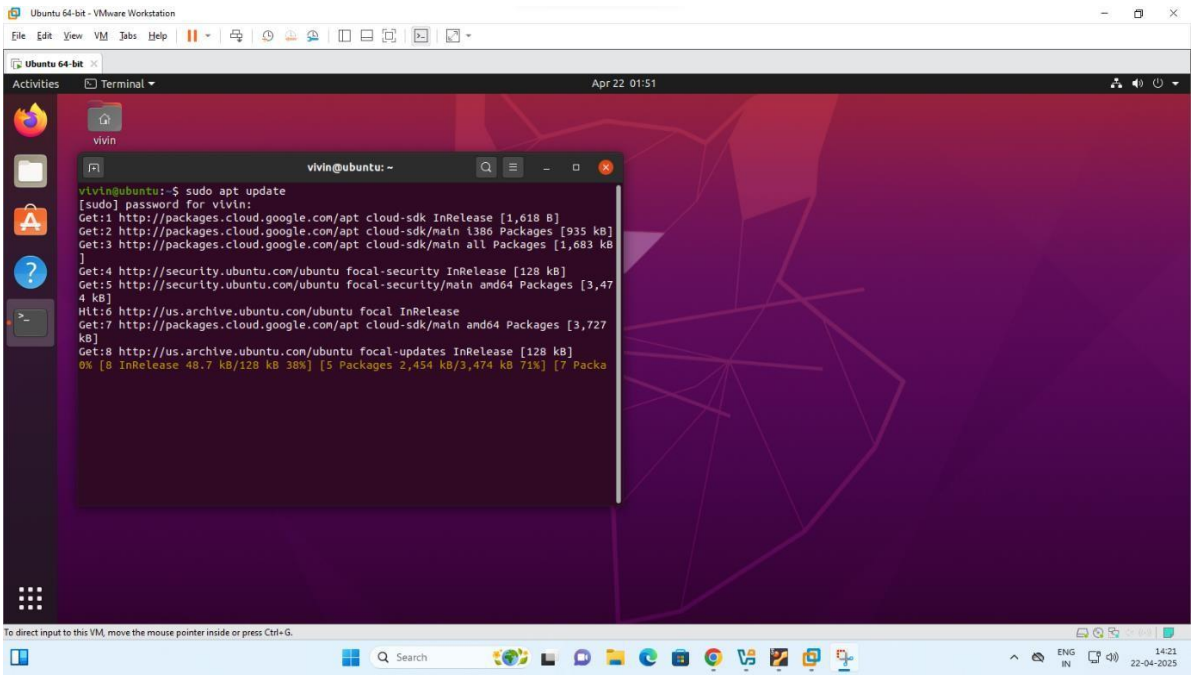
To install Hadoop single node cluster and run simple application like word count program.

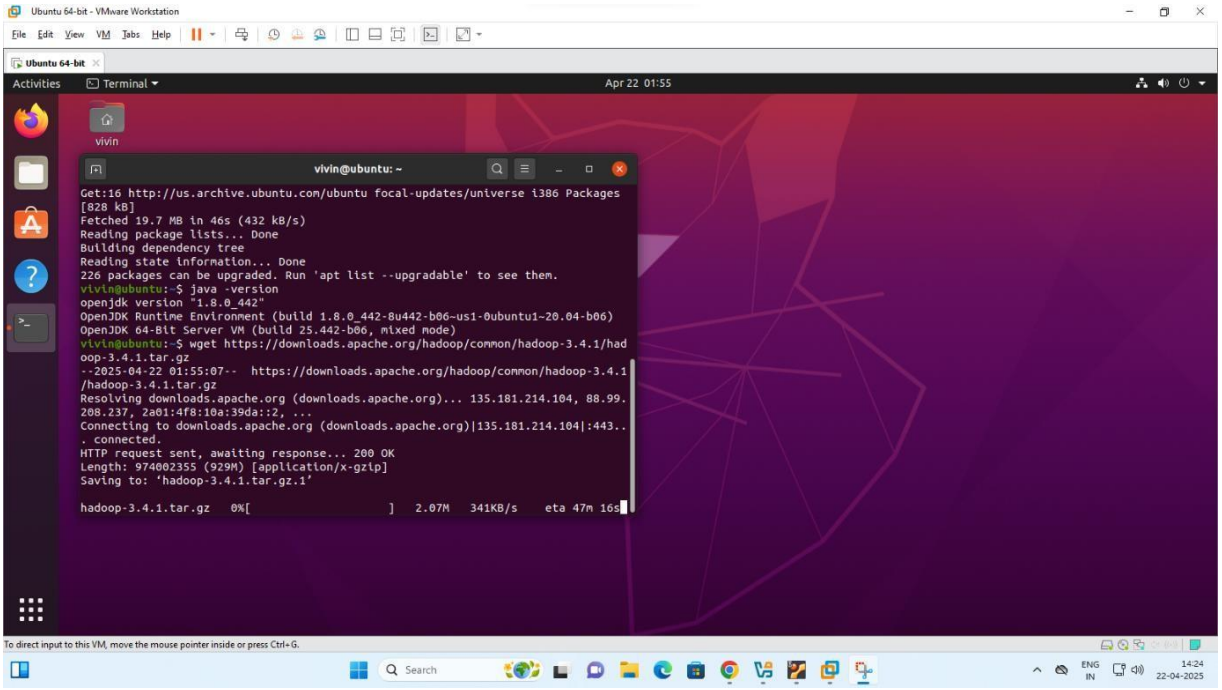
Procedure :

Step 1:

To install a Hadoop

1. To update the apt in your system using **$sudo apt update** command.
2. To install java jdk on your system using **$sudo apt install openjdk-8-jdk** command.
3. To use **java -version** command to verify the java installation.
4. To install Hadoop using **$wget** **https://downloads.apache.org/hadoop/common/hadoop-3.4.1/hadoop-3.4.1.tar.gz**





Step 2:

To create a text file and named as **wordcount.java** and to write the below program into the file

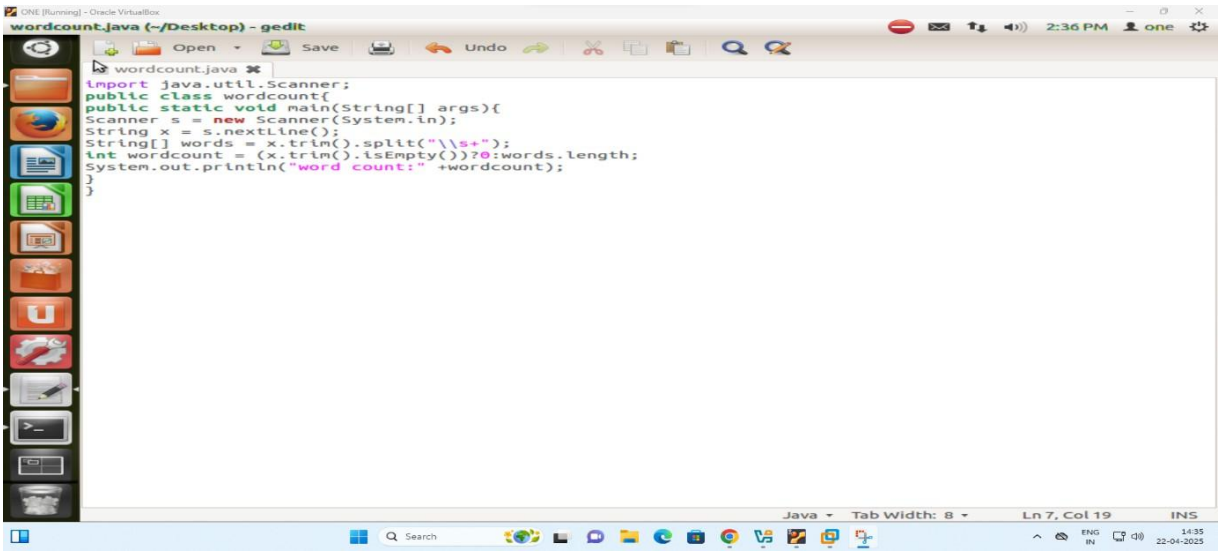
```
import java.util.Scanner;

public class WordCount {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter a sentence:");
        String input = scanner.nextLine();

        String[] words = input.trim().split("\\s+");
```

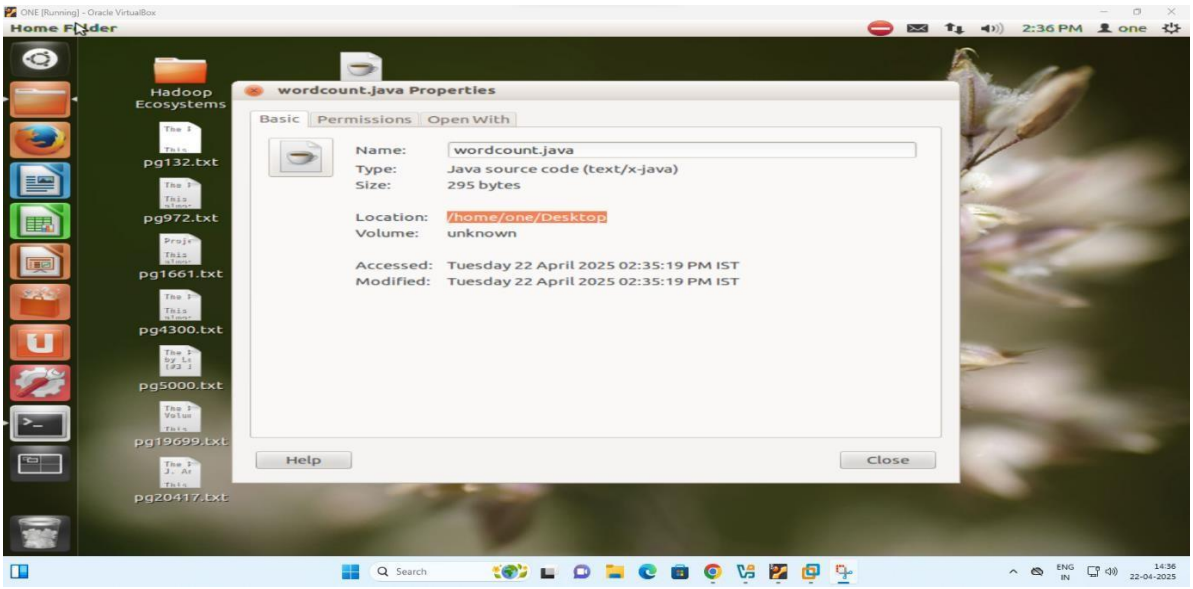
```
        int wordCount = (input.trim().isEmpty()) ? 0 : words.length;
        System.out.println("Word count: " + wordCount);
        scanner.close();
    }
}
```

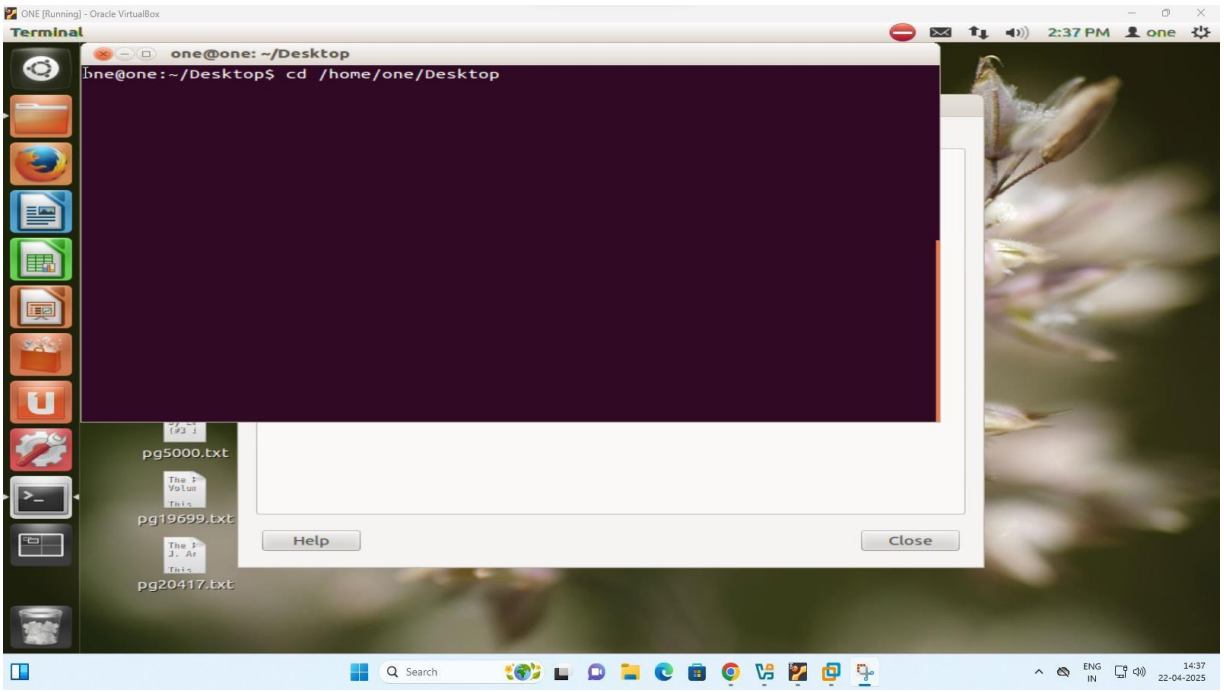


Step 3:

To compile and the run the program.

1. To copy the file path and move to location on the terminal





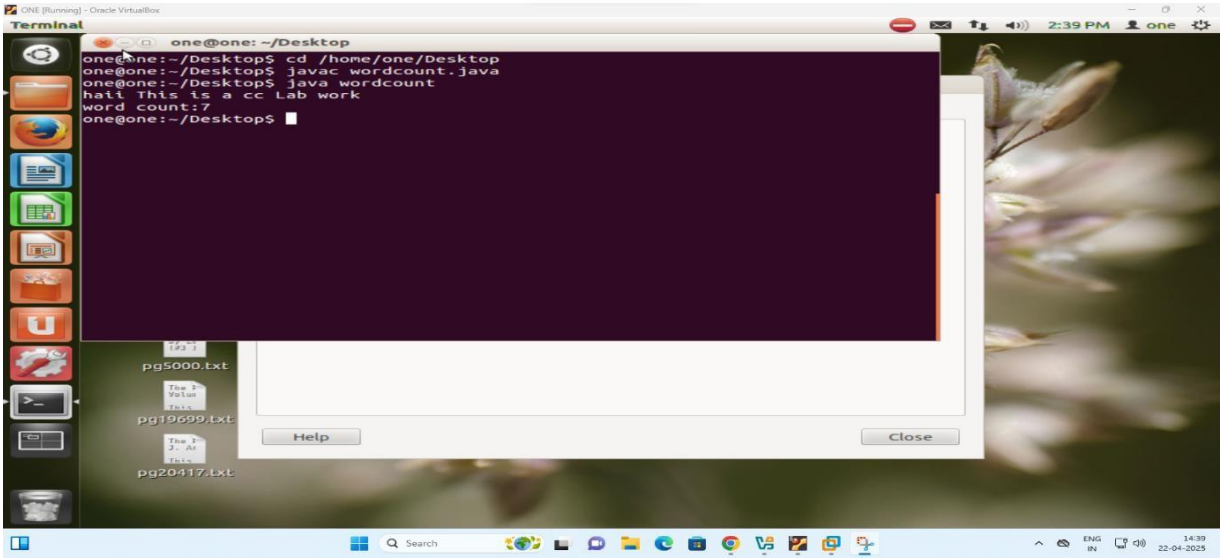
2. To compile the java program

Javac wordcount.java

`

3. To run the java program

Java wordcount



Result :

Thus, the install hadoop single node cluster and run simple application like word count program was successfully executed.

EX NO: 8 DATE:	**Creating and executing your first container using docker**

Aim:

To create and execute a container using docker.

PROCEDURE:

Install Prerequisites

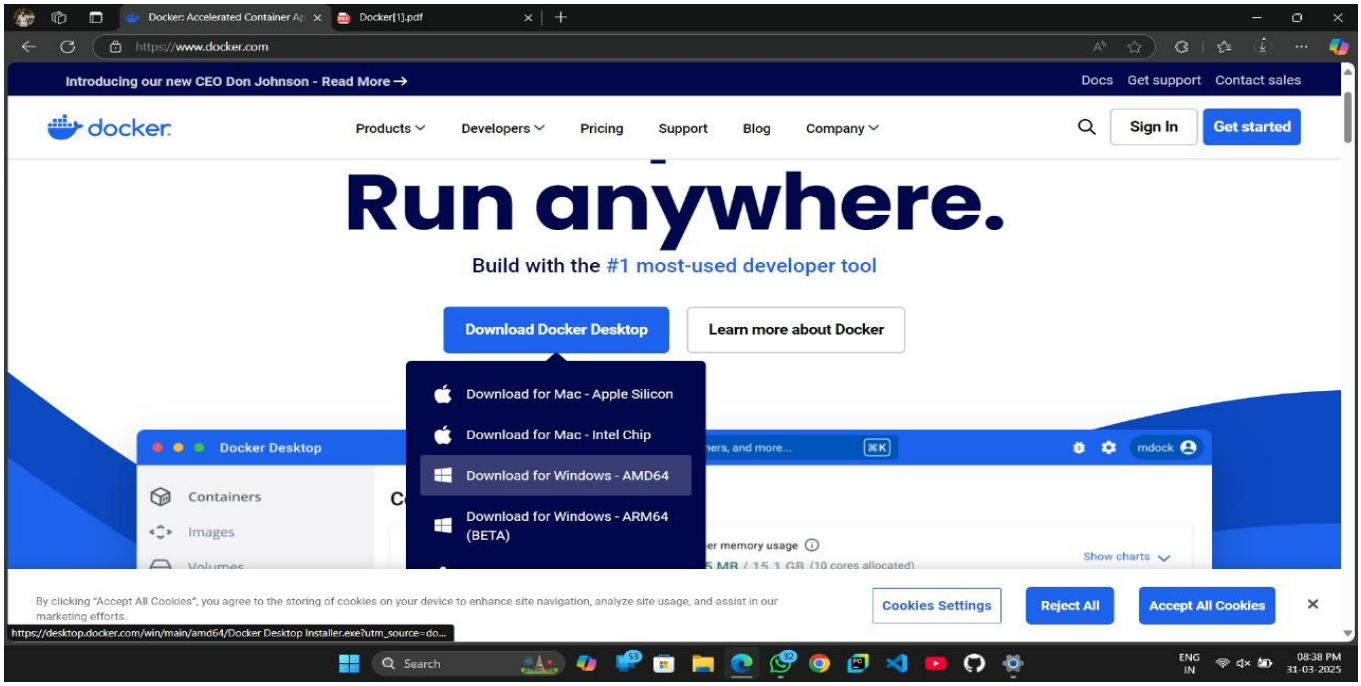
Install the following:

➢ VS Code

➢ Docker VS Code Extensions

• Open VS Code, go to Extensions (Ctrl+Shift+X)

• Search for "Docker" and install the official Docker extension



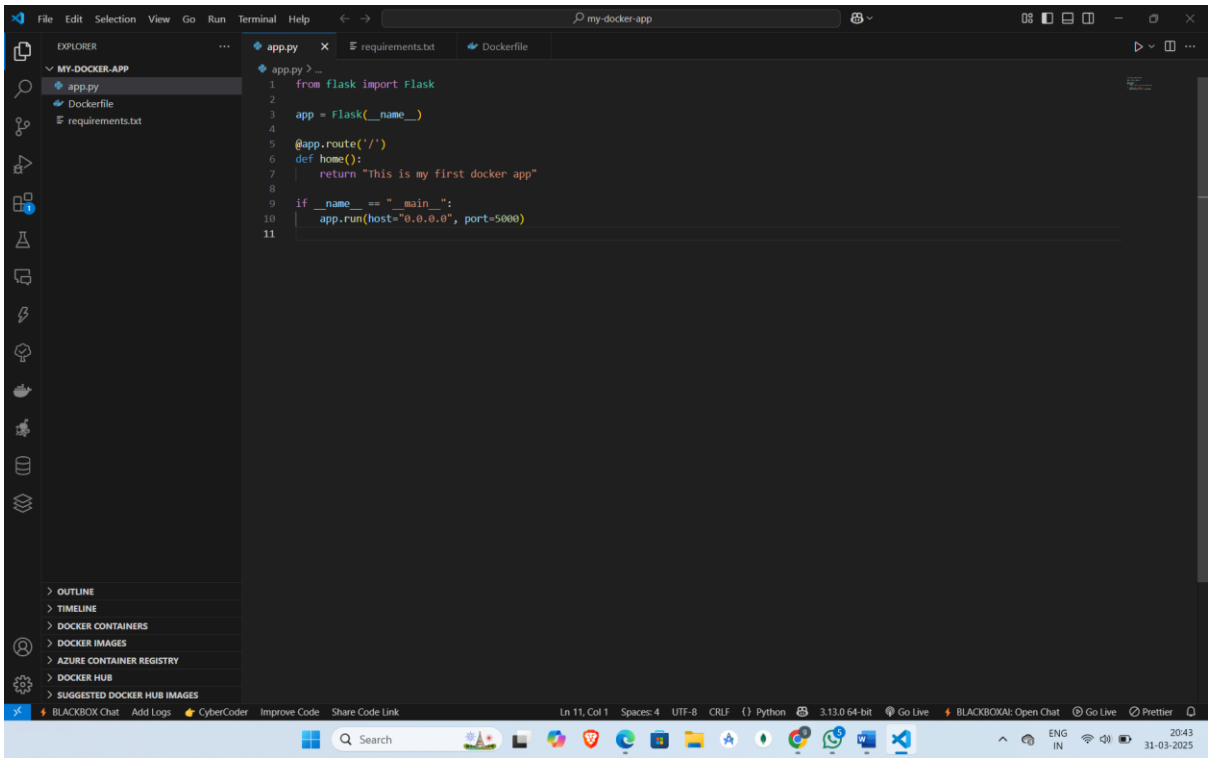
Set Up the Project in VS Code

➢ Open VS Code and create a new folder for your project:

• Click File → Open Folder • Choose or create a folder, e.g., my-docker-app

➢ Open the VS Code Terminal (Ctrl + ~) Run the following command to create the required files.

➢ touch app.py requirements.txt Dockerfile



Write the Flask Application

Inside app.py, add: python

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
    return "This is my first docker App!!"
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Inside requirements.txt, add:

```
flask
```

Inside Dockerfile, add: dockerfile

```
# Use an official Python image
FROM python:3.9
# Set the working directory
WORKDIR /app
```

Copy files into the container

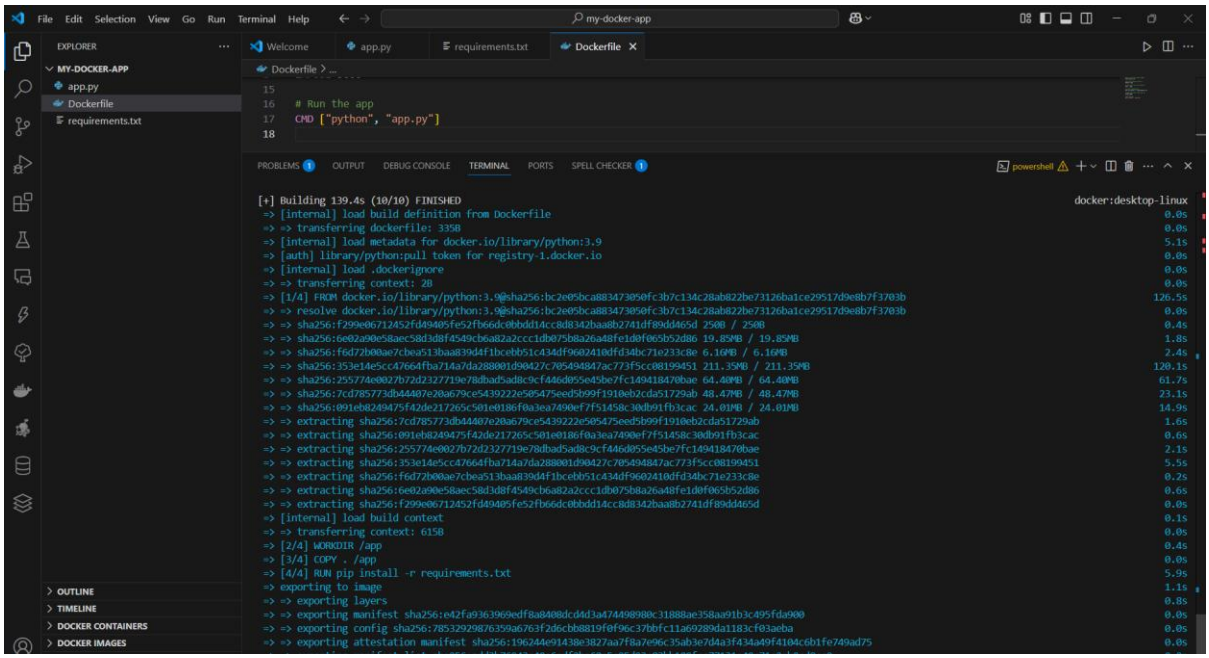
COPY . /app
Install dependencies
RUN pip install -r requirements.txt
Expose the port Flask runs on
EXPOSE 5000
Run the app
CMD ["python", "app.py"]

Build & Run the Docker Container

Open **VS Code Terminal** (Ctrl + ~)

1.Build the Docker image:

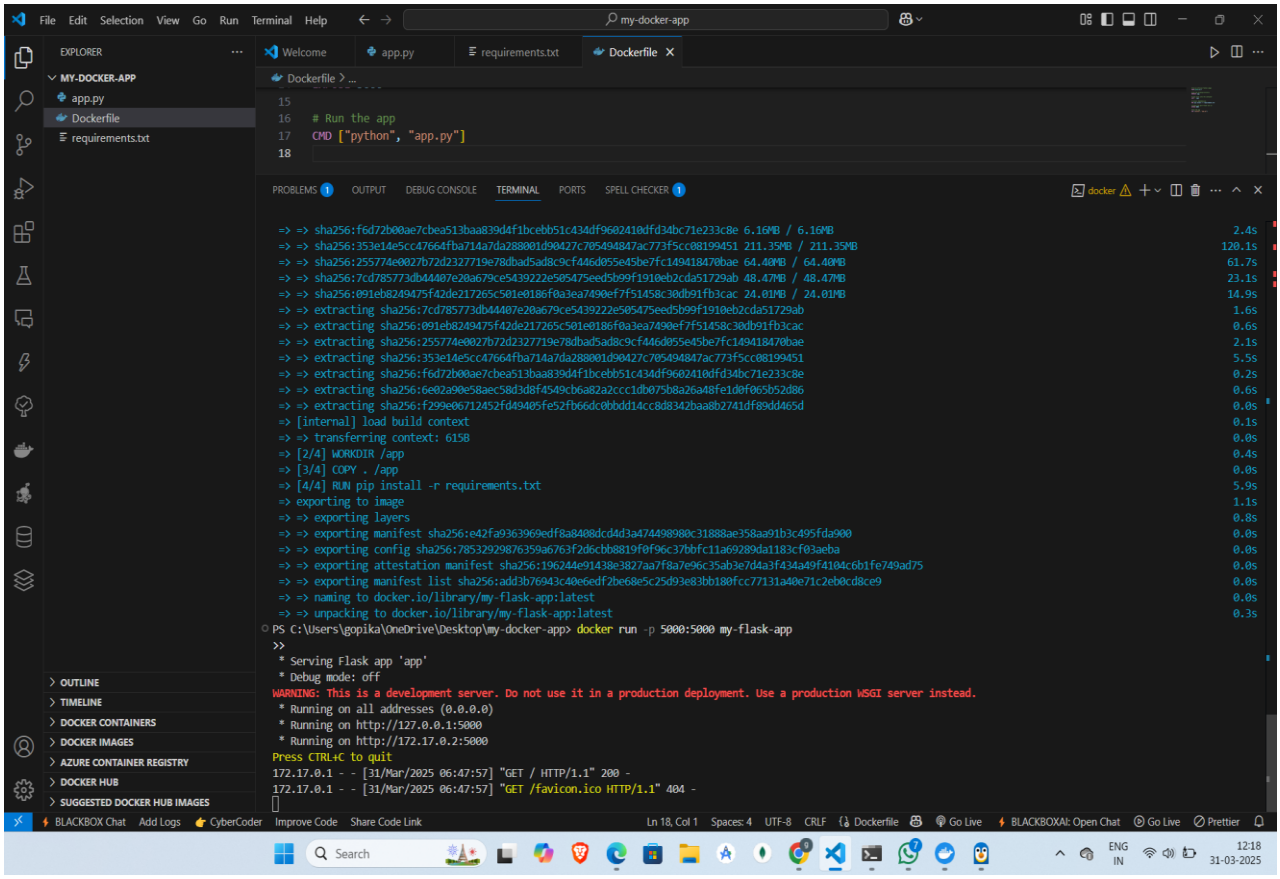
docker build -t my-flask-app

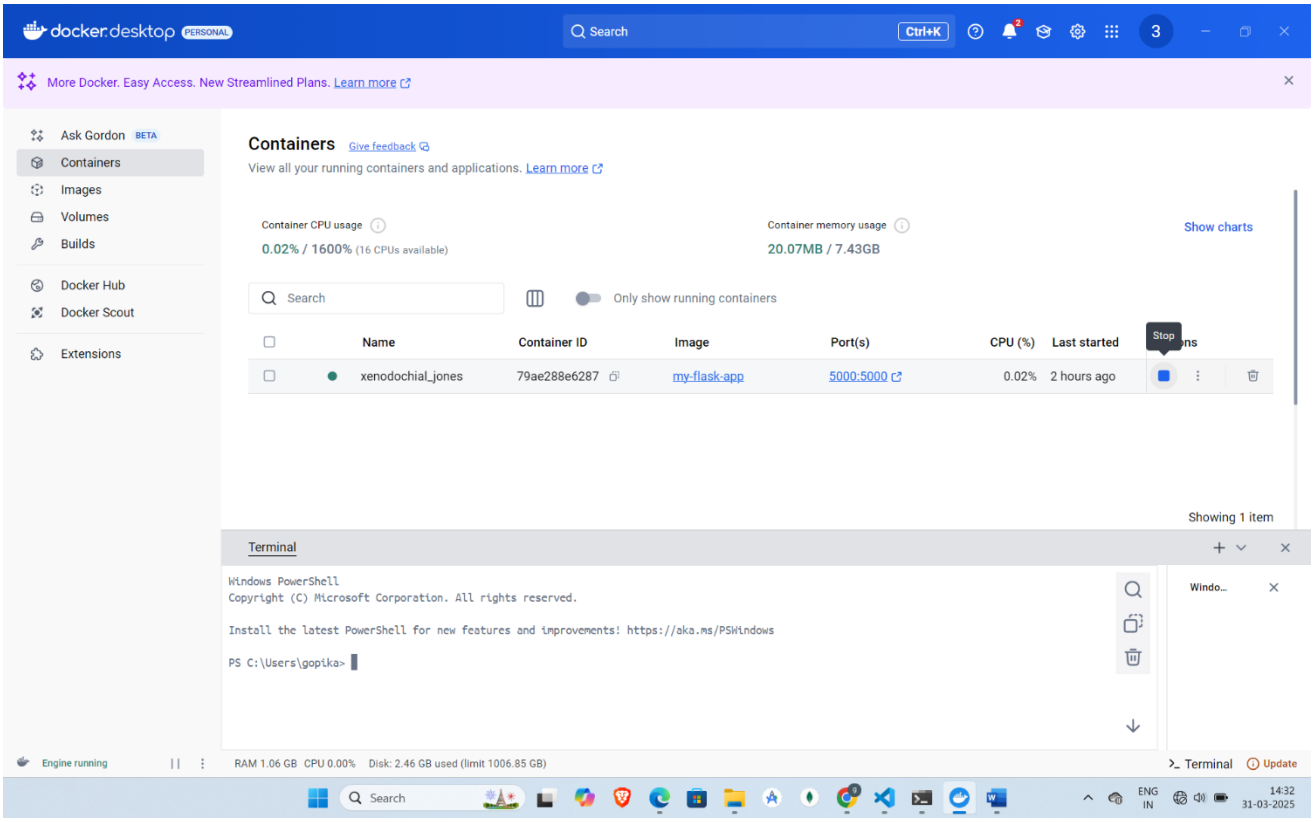


2.Run the container:

docker run -p 5000:5000 my-flask-app

Open **http://localhost:5000** in your browser.

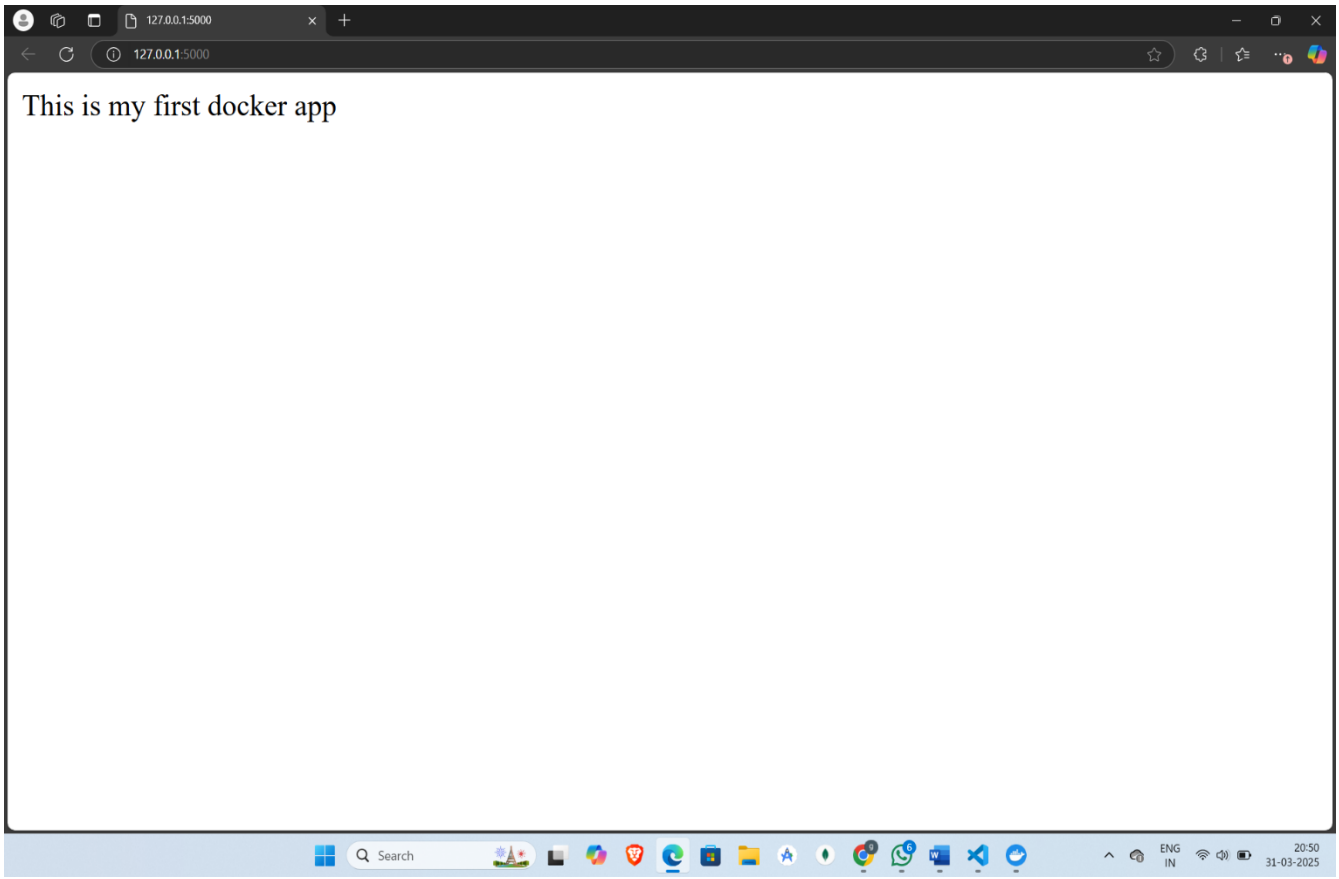

File Edit Selection View Go Run Terminal Help
my-docker-app
EXPLORER
MY-DOCKER-APP
app.py
Dockerfile
requirements.txt
Welcome
app.py
requirements.txt
Dockerfile
Run the app
CMD ["python", "app.py"]
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SPELL CHECKER 1
=> => sha256:f6d72b00ae7cbea513baa839d4f1bcebb51c434df9602410dfd34bc71e233c8e 6.16MB / 6.16MB 2.4s
=> => sha256:353e14e5cc47664fba714a7da288001d90427c705494847ac773f5cc08199451 211.35MB / 211.35MB 120.1s
=> => sha256:255774e0027b72d2327719e78dbad5ad8c9cf446d055e45be7fc149418470bae 64.40MB / 64.40MB 61.7s
=> => sha256:7cd785773db44407e20a679ce5439222e505475eed5b99f1910eb2cda51729ab 48.47MB / 48.47MB 23.1s
=> => sha256:091eb8249475f42de217265c501e0186f0a3ea7490ef7f51458c30db91fb3cac 24.01MB / 24.01MB 14.9s
=> => extracting sha256:7cd785773db44407e20a679ce5439222e505475eed5b99f1910eb2cda51729ab 1.6s
=> => extracting sha256:091eb8249475f42de217265c501e0186f0a3ea7490ef7f51458c30db91fb3cac 0.6s
=> => extracting sha256:255774e0027b72d2327719e78dbad5ad8c9cf446d055e45be7fc149418470bae 2.1s
=> => extracting sha256:353e14e5cc47664fba714a7da288001d90427c705494847ac773f5cc08199451 5.5s
=> => extracting sha256:f6d72b00ae7cbea513baa839d4f1bcebb51c434df9602410dfd34bc71e233c8e 0.2s
=> => extracting sha256:6e02a90e58aec58d3d8f4549cb6a82a2ccc1db075b8a26a48fe1d0f065b52d86 0.6s
=> => extracting sha256:f299e06712452fd49405fe52fb66dc0bbdd14cc8d8342baa8b2741df89dd465d 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 6158 0.0s
=> [2/4] WORKDIR /app 0.4s
=> [3/4] COPY . /app 0.0s
=> [4/4] RUN pip install -r requirements.txt 5.9s
=> exporting to image 1.1s
=> => exporting layers 0.8s
=> => exporting manifest sha256:e42fa9363969edf8a8408dcd4d3a474498980c31888ae358aa91b3c495fda900 0.0s
=> => exporting config sha256:7853292987635 9a6763f2d6cbb8819f0f96c37bbfc11a69289da1183cf03aeba 0.0s
=> => exporting attestation manifest sha256:196244e91438e3827aa7f8a7e96c35ab3e7d4a3f434a49f4104c6b1fe749ad75 0.0s
=> => exporting manifest list sha256:add3b76943c40e6edf2be68e5c25d93e83bb180fcc77131a40e71c2eb0cd8ce9 0.0s
=> => naming to docker.io/library/my-flask-app:latest 0.0s
=> => unpacking to docker.io/library/my-flask-app:latest 0.3s
PS C:\Users\gopika\OneDrive\Desktop\my-docker-app> docker run -p 5000:5000 my-flask-app
>>
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [31/Mar/2025 06:47:57] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [31/Mar/2025 06:47:57] "GET /favicon.ico HTTP/1.1" 404 -
OUTLINE
TIMELINE
DOCKER CONTAINERS
DOCKER IMAGES
AZURE CONTAINER REGISTRY
DOCKER HUB
SUGGESTED DOCKER HUB IMAGES
BLACKBOX Chat Add Logs CyberCoder Improve Code Share Code Link
Ln 18, Col 1 Spaces: 4 UTF-8 CRLF Dockerfile Go Live BLACKBOXAI: Open Chat Go Live Prettier
Search
ENG IN
12:18 31-03-2025


docker desktop PERSONAL
Search Ctrl+K
More Docker. Easy Access. New Streamlined Plans. Learn more
Ask Gordon BETA
Containers
Images
Volumes
Builds
Docker Hub
Docker Scout
Extensions
Containers
Give feedback
View all your running containers and applications. Learn more
Container CPU usage
0.02% / 1600% (16 CPUs available)
Container memory usage
20.07MB / 7.43GB
Show charts
Search
Only show running containers
Name
Container ID
Image
Port(s)
CPU (%)
Last started
Stop
xenodochial_jones
79ae288e6287
my-flask-app
5000:5000
0.02%
2 hours ago
Showing 1 item
Terminal
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\gopika>
Engine running
RAM 1.06 GB CPU 0.00% Disk: 2.46 GB used (limit 1006.85 GB)
Terminal
Update
Search
ENG IN
14:32 31-03-2025

Use VS Code's Docker Extension

With the **Docker extension** installed:

- Open **Docker Panel** (Ctrl+Shift+P → Docker: Show Containers)
- See running containers, images, logs, and even stop/start containers visually!



Debugging in VS Code

Add a .vscode/launch.json file

Use this configuration:

```
{
   "version": "0.2.0",
   "configurations": [
      {
         "name": "Python: Remote Attach",
         "type": "python",
         "request": "attach",
```

```
        "connect": {
            "host": "localhost",
            "port": 5678
        },
        "pathMappings": [
            {
                "localRoot": "${workspaceFolder}",
                "remoteRoot": "/app"
            }
        ]
    }
  ]
}
```

Modify app.py to allow debugging:

```
import debugpy
debugpy.listen(("0.0.0.0", 5678))
print("Waiting for debugger to attach...")
debugpy.wait_for_client()
```

Restart the container and attach the VS Code debugger!

Stop & Clean Up

- **Stop container**

docker stop <container_id>

- **Remove all stopped containers:**

docker container prune

- **Remove the image:**

docker rmi my-flask-app

RESULT:

Thus a container using docker has been created and executed successfully.

Ex no:9 Date:	9.Run a Container from Docker Hub.

Aim:

To run a containerized application using an image from Docker Hub.

Software Requirement:

- Docker (Install from: https://docs.docker.com/get-docker/)
- Terminal/Command Line

Theory:

Docker Hub is a cloud-based registry that allows you to pull (download) and push (upload) container images. Many open-source and official container images are hosted here for quick deployment.

Running a container from Docker Hub involves:

- Pulling an image from the Docker Hub repository.
- Running the image as a container using Docker Engine.

Procedure:

1. Install Docker (if not already installed):

Download and install Docker Desktop for your OS:
https://docs.docker.com/get-docker/

2.Verify installation:

```
docker --version
```

3. Pull an image from Docker Hub:

Example: Pull the official Nginx image.

```
docker pull nginx
```

4. Run the container using the pulled image:

```
docker run -d -p 8080:80 nginx
```

This command:

- Runs the container in detached mode (-d)
- Maps container port 80 to local port 8080
- Uses the nginx image from Docker Hub

4. Verify the container is running:

docker ps

5. Access the application:

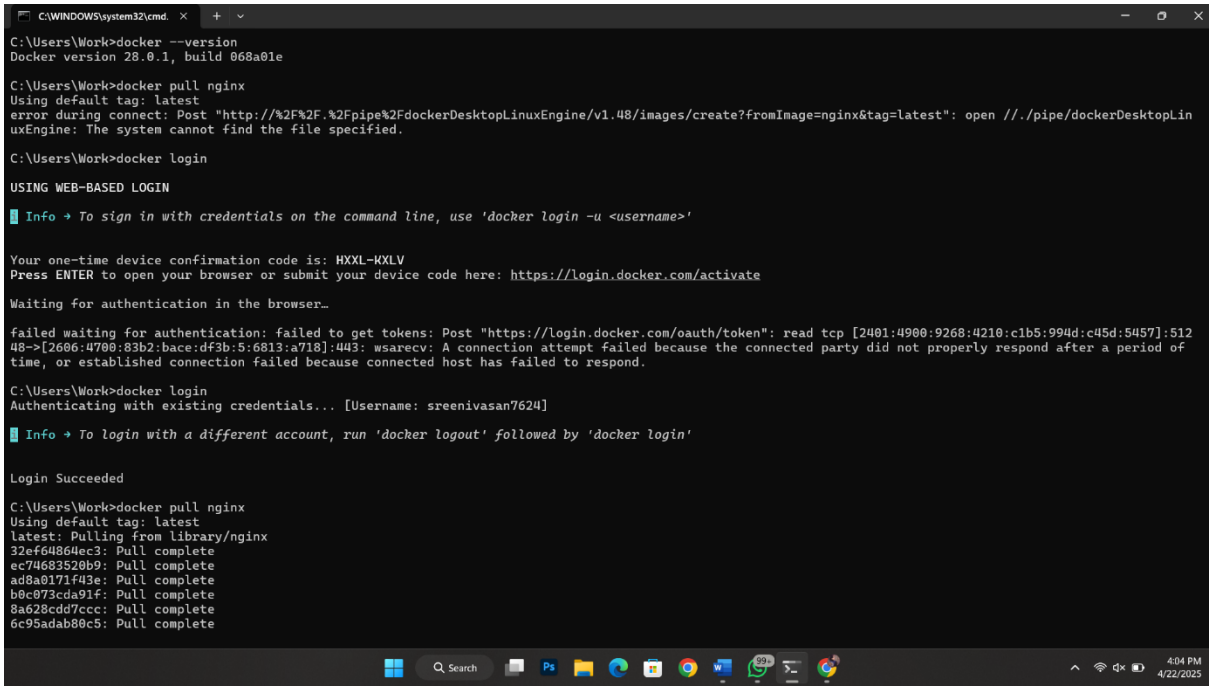
Open your web browser and go to:http://localhost:8080

You should see the default Nginx welcome page.

6. Stop and remove the container (optional):

docker stop <container_id>

docker rm <container_id>



Result:

Thus, a Docker container was successfully run using an image from Docker Hub, and its output was verified via the local web browser.