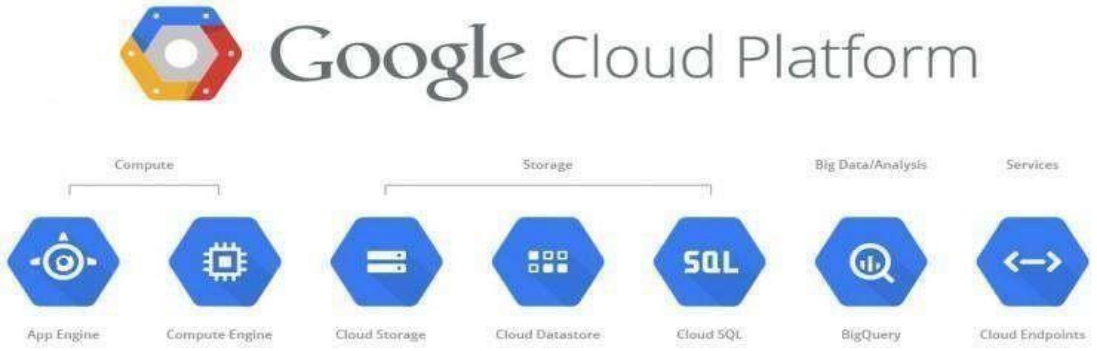
Ex No. 1 & 2	**Install Google App Engine. Create hello world app and other simple web applications using python/java. Use GAE launcher to launch the web applications**

Introduction

- **Google Cloud Platform (GCP)**
  - **Google Cloud Platform** (**GCP**), offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube.
  - Alongside a set of management tools, it provides a series of modular cloud services including computing, data storage, data analytics and machine learning.
  - Google Cloud Platform provides infrastructure as a service, platform as a service, and serverless computing environments.



- **Platform as a Service (PaaS)** ○ Cloud computing service which provides a computing platform and a solution stack as a service.
  - Consumer creates the software using tools and/or libraries from the provider. ○ Provider provides the networks, servers, storage, etc.

- **Google App Engine:**
  - Google App Engine was first released as a beta version in April 2008.
  - It is a is a Platform as a Service (PaaS) cloud computing platform for developing and hosting web applications in Google-managed data centers.

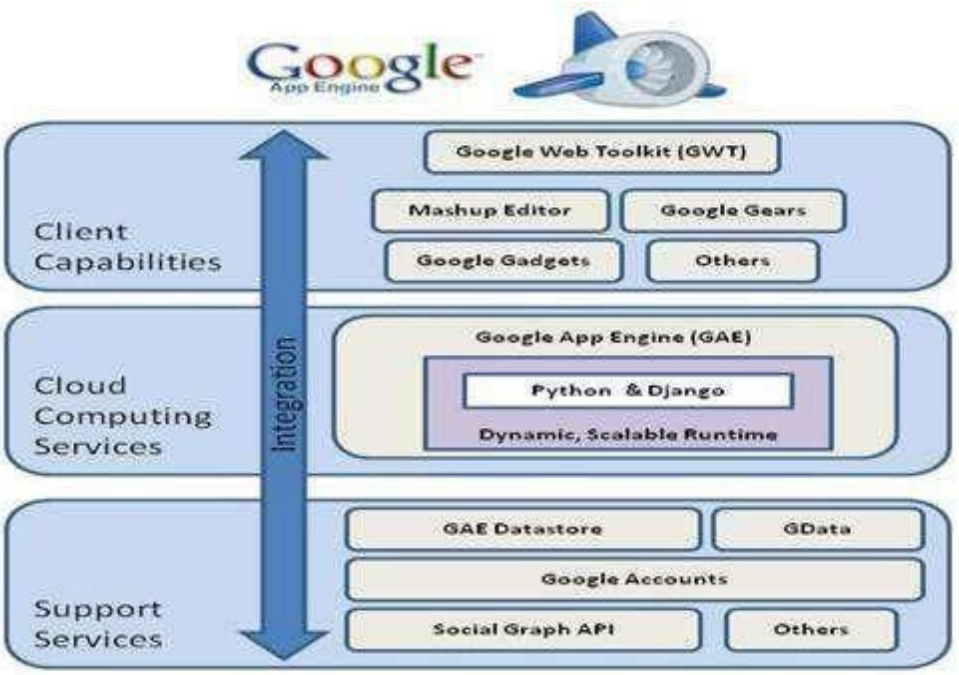
- Google‘s App Engine opens Google‘s production to any person in the world at no charge.
- Google App Engine is software that facilitates the user to run his web applications on Google infrastructure.
- It is more reliable because failure of any server will not affect either the performance of the end user or the service of the Google.
- It virtualizes applications across multiple servers and data centers.

✦ Other cloud-based platforms include offerings such as Amazon Web Services and Microsoft's Azure Services Platform. **O Introduction of Google App Engine**

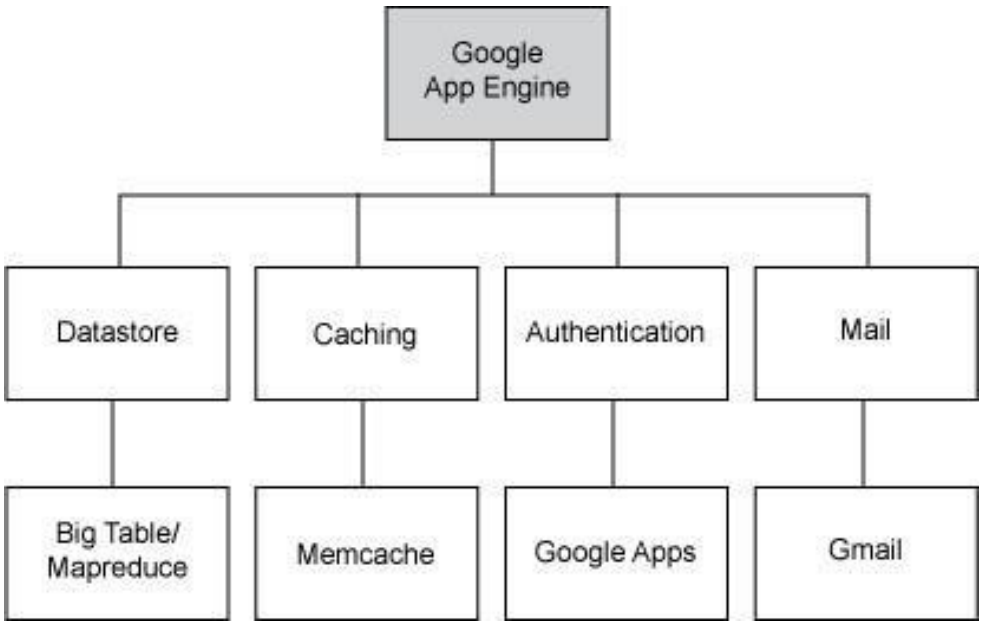
- Google App Engine lets you run your web applications on Google's infrastructure. App Engine applications are easy to build, easy to maintain,and easy to scale as your traffic and data storage needs grow. With App Engine, there are no servers to maintain: You just upload your application, and it's ready to serve your users.
- You can serve your app from your own domain name (such as https://www.example.com/) using Google Apps. Or, you can serve your app using a free name on the appspot.com domain. You can share your application with the world, or limit access to members of your organization.
- Google App Engine supports apps written in several programming languages. With App Engine's Java runtime environment, you can build your app using standard Java technologies, including the JVM, Java servlets, and the Java programming language—or any other language using a JVM-based interpreter or compiler, such as JavaScript or Ruby. App Engine also features a dedicated Python runtime environment, which includes a fast Python interpreter and the Python standard library. The Java and Python runtime environments are built to ensure that your application runs quickly, securely, and without interference from other apps on the system.
- With App Engine, you only pay for what you use. There are no set-up costs and no recurring fees. The resources your application uses, such as storage and bandwidth, are measured by the gigabyte, and billed at competitive rates. You control the maximum amounts of resources your app can consume, so it always stays within your budget. App Engine costs nothing to get started. All applications can use up to 500 MB of storage and enough CPU and bandwidth to support an efficient app serving around 5 million page views a month,

absolutely free. When you enable billing for your application, your free limits are raised, and you only pay for resources you use above the free levels.

- **Architecture of Google App Engine**



- **Features of Google App Engine**



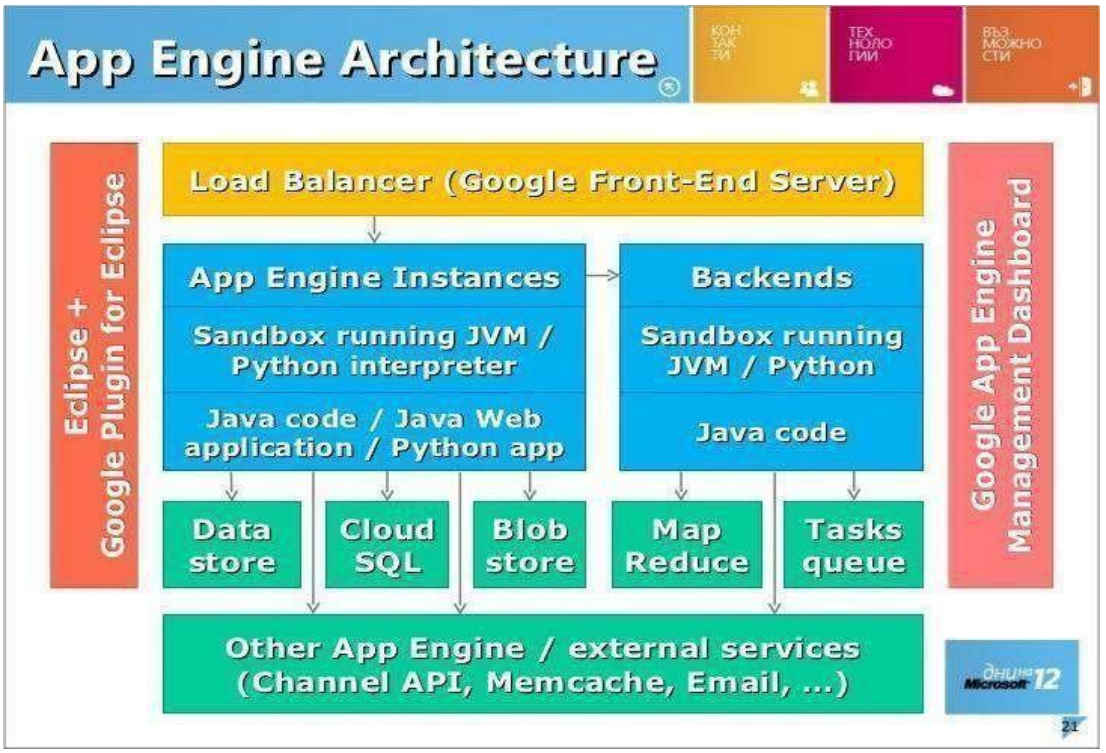
- **GAE Application Environment:**

  - Google App Engine makes it easy to build an application that runs reliably, even under heavy load and with large amounts of data. App Engine includes the following features:
  - Persistent storage with queries, sorting and transactions
  - Automatic scaling and load balancing
  - APIs for authenticating users and sending email using Google Accounts
  - Task queues for performing work outside of the scope of a web request
  - Scheduled tasks for triggering events at specified times and regular intervals
  - Dynamic web serving, with full support for common web technologies

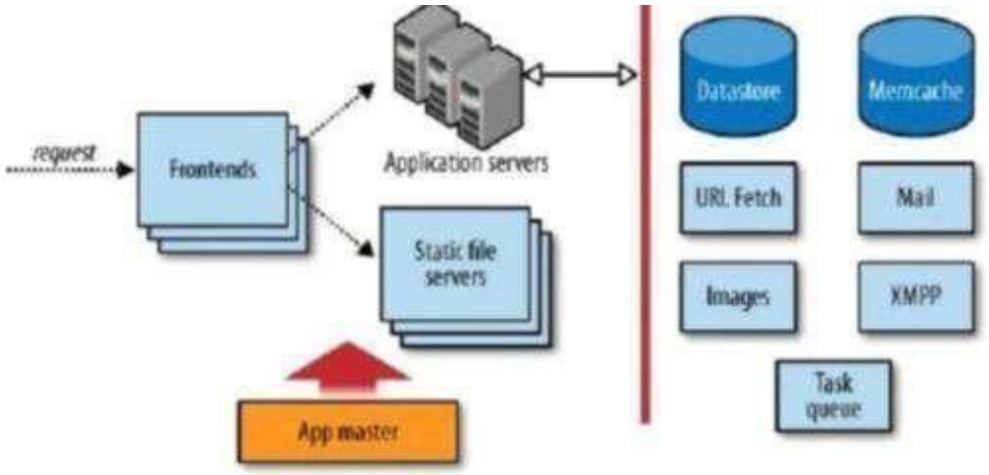
- **Java Runtime Environment**

  - You can develop your application for the Java runtime environment using common Java web development tools and API standards. Your app interacts with the environment using the Java Servlets standard, and can use common web application technologies such as Java Server Pages
  - The Java runtime environment uses Java 6. The App Engine Java SDK supports developing apps using either Java 5 or 6. The environment includes the Java SE Runtime Environment (JRE) 6 platform and libraries. The restrictions of the sandbox environment are implemented in the JVM. An app can use any JVM byte code or library feature, as long as it does not exceed the sandbox restrictions. For instance, byte code that attempts to open a socket or write to a file will throw a runtime exception.
  - Your app accesses most App Engine services using Java standard APIs. For the App Engine data store, the Java SDK includes implementations of the Java Data Objects (JDO) and Java Persistence API (JPA) interfaces. Your app can use the JavaMail API to send email messages with the App Engine Mail service. The java.net HTTP APIs accesses the App Engine URL fetch service.
  - App Engine also includes low-level APIs for its services to implement additional adapters, or to use directly from the application. See the documentation for the data store, memcache, URL fetch, mail, images and Google Accounts APIs. Typically, Java developers use the Java programming language and APIs to implement web applications for the JVM. With the use

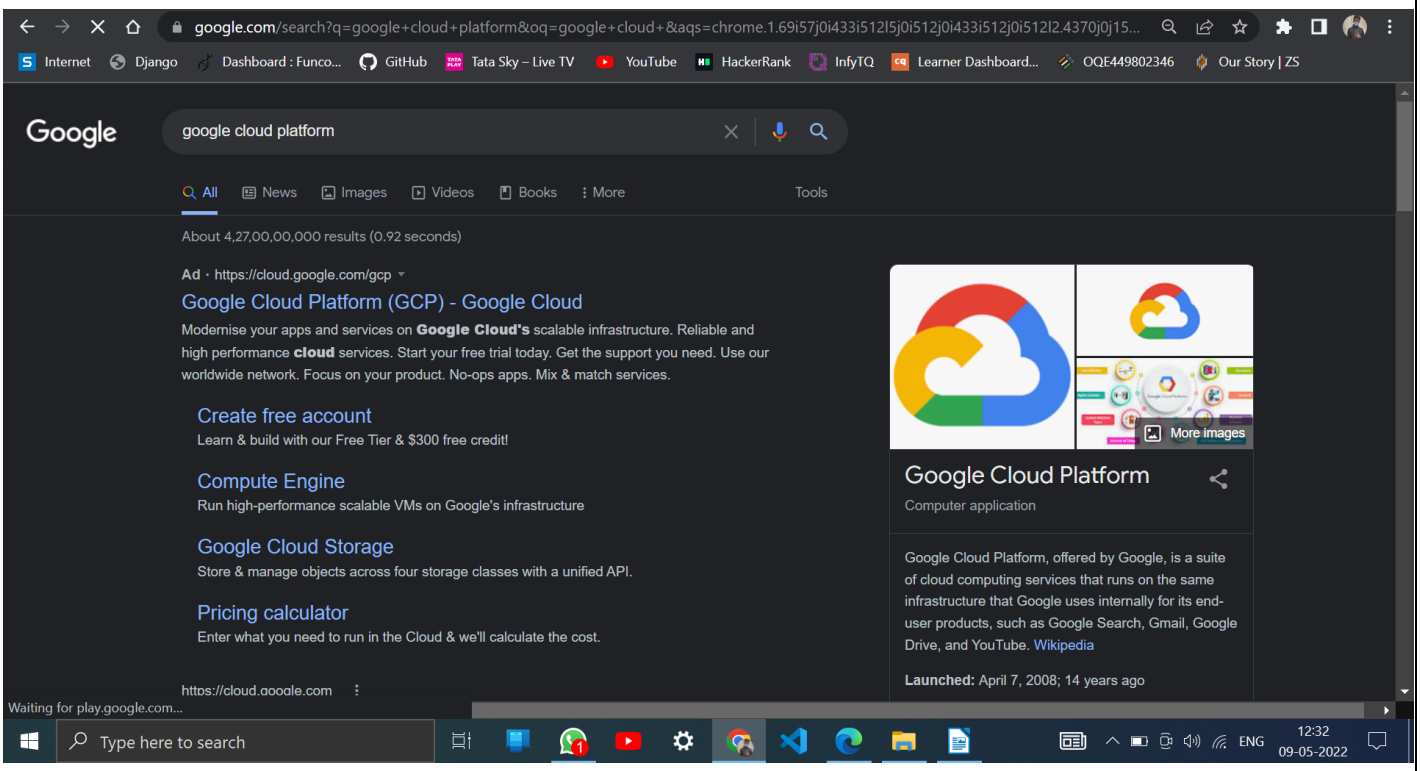
of JVM-compatible compilers or interpreters, you can also use other languages to develop web applications, such as JavaScript, Ruby.



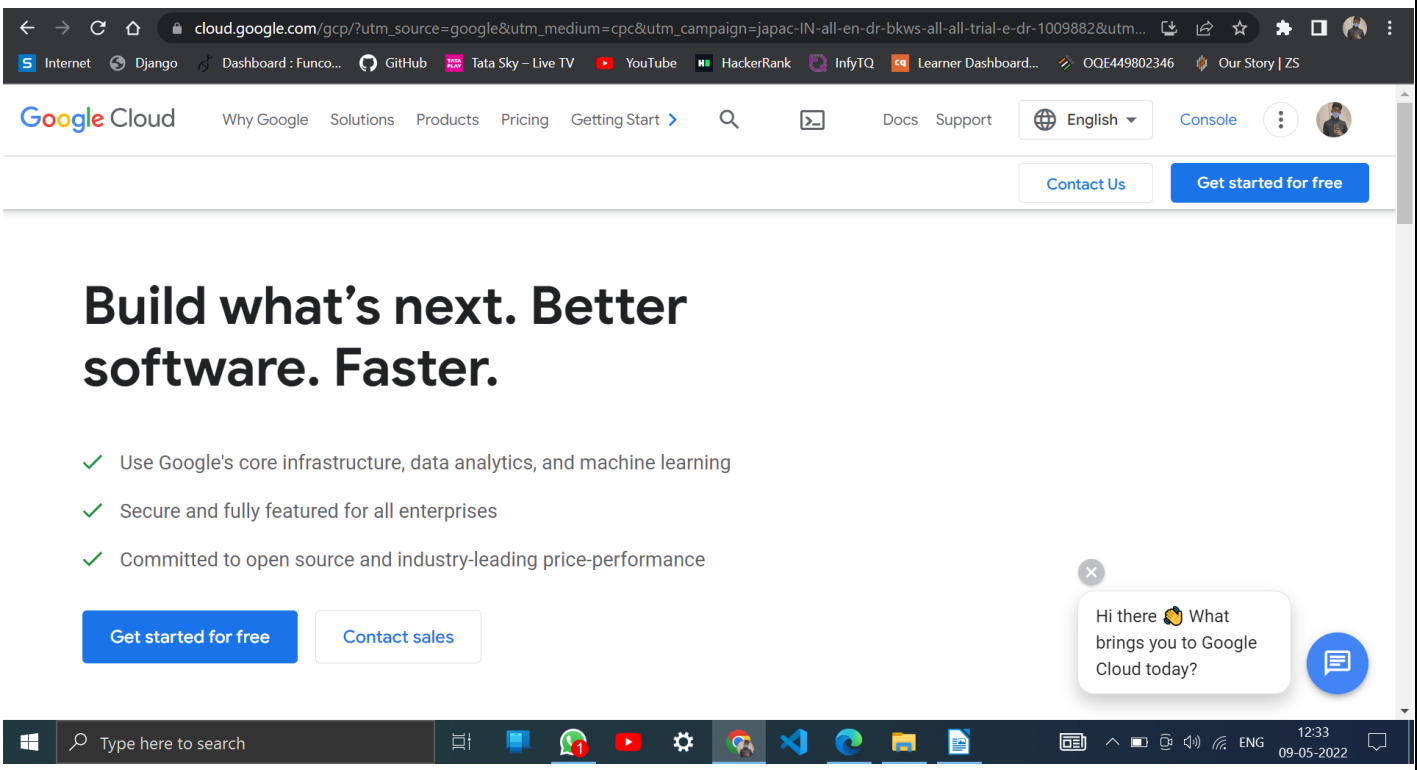
- **Workflow of Google App Engine**



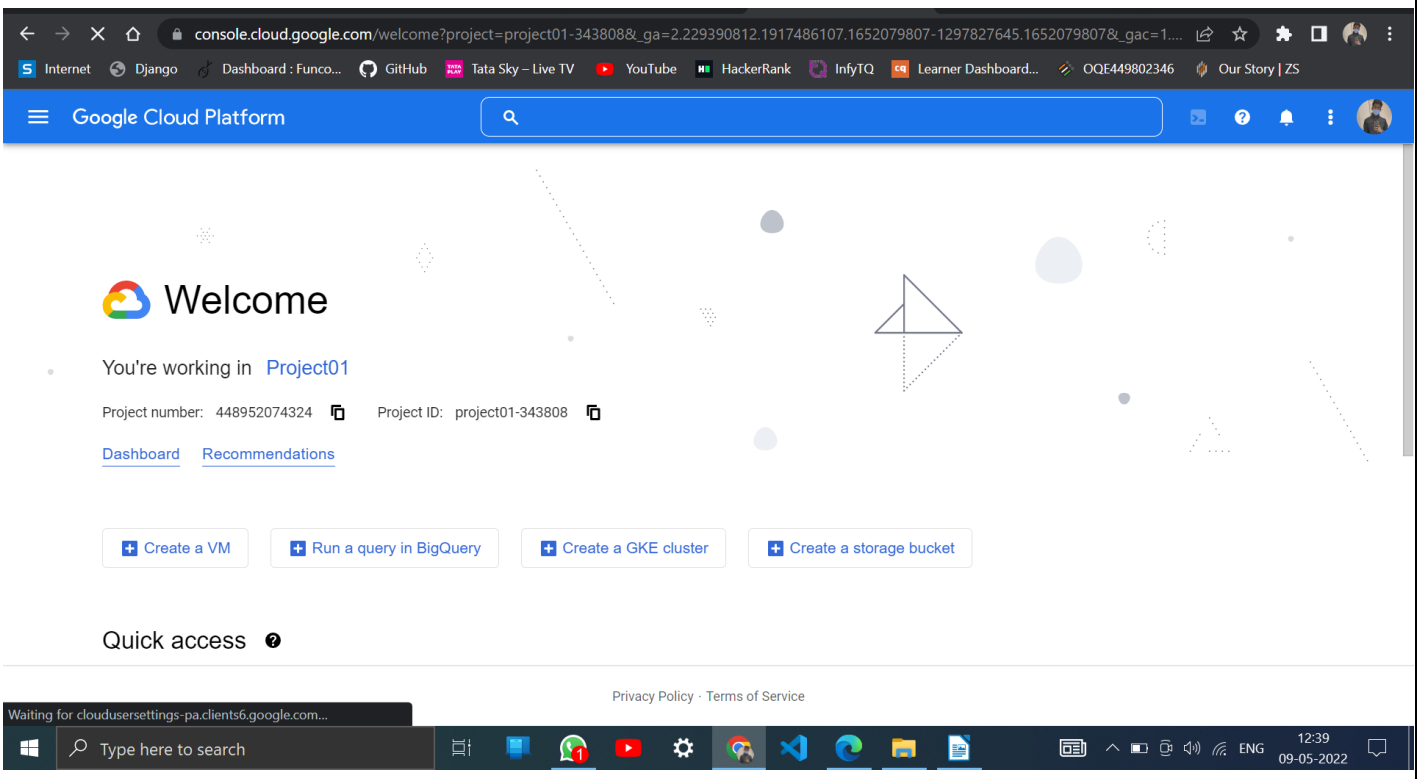
Step1 : Login to www.cloud.google.com



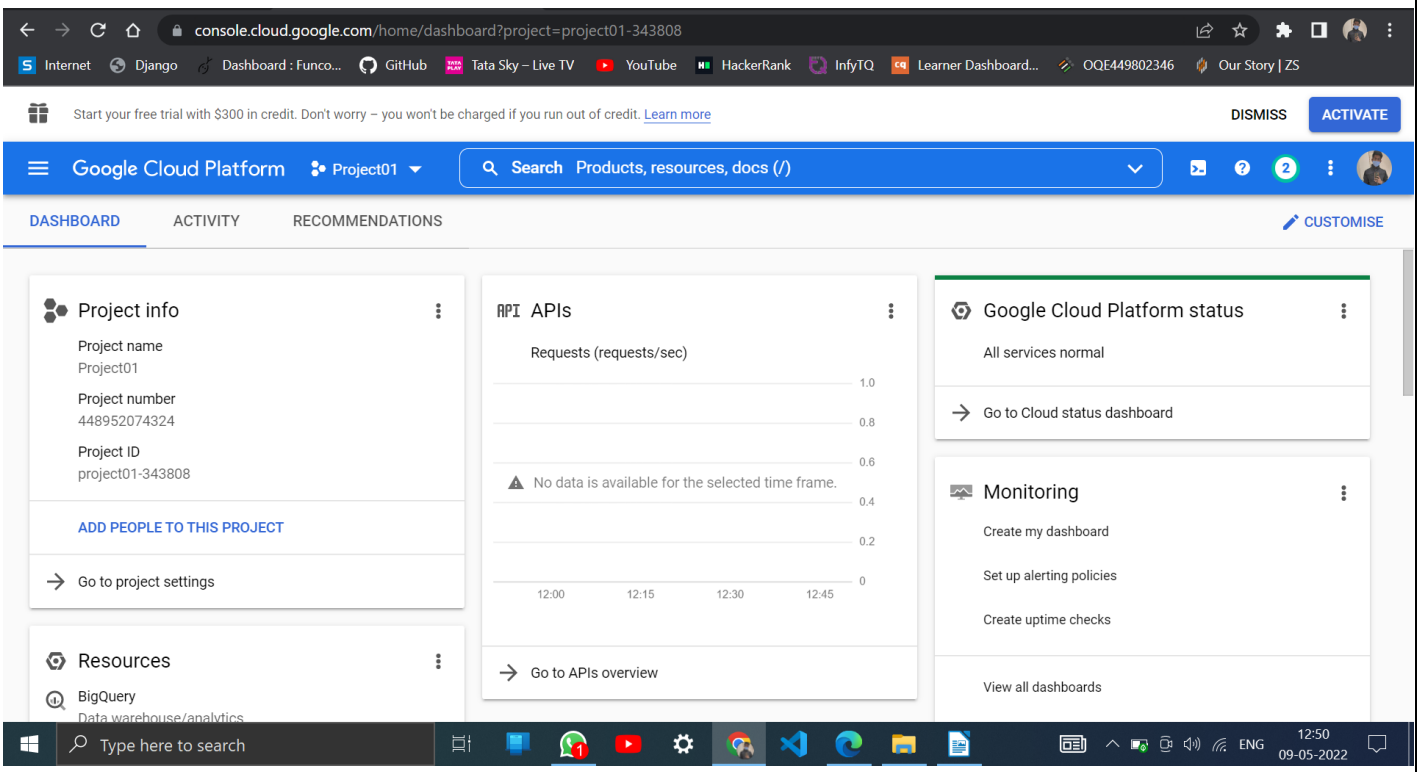
Step2 : Goto Console



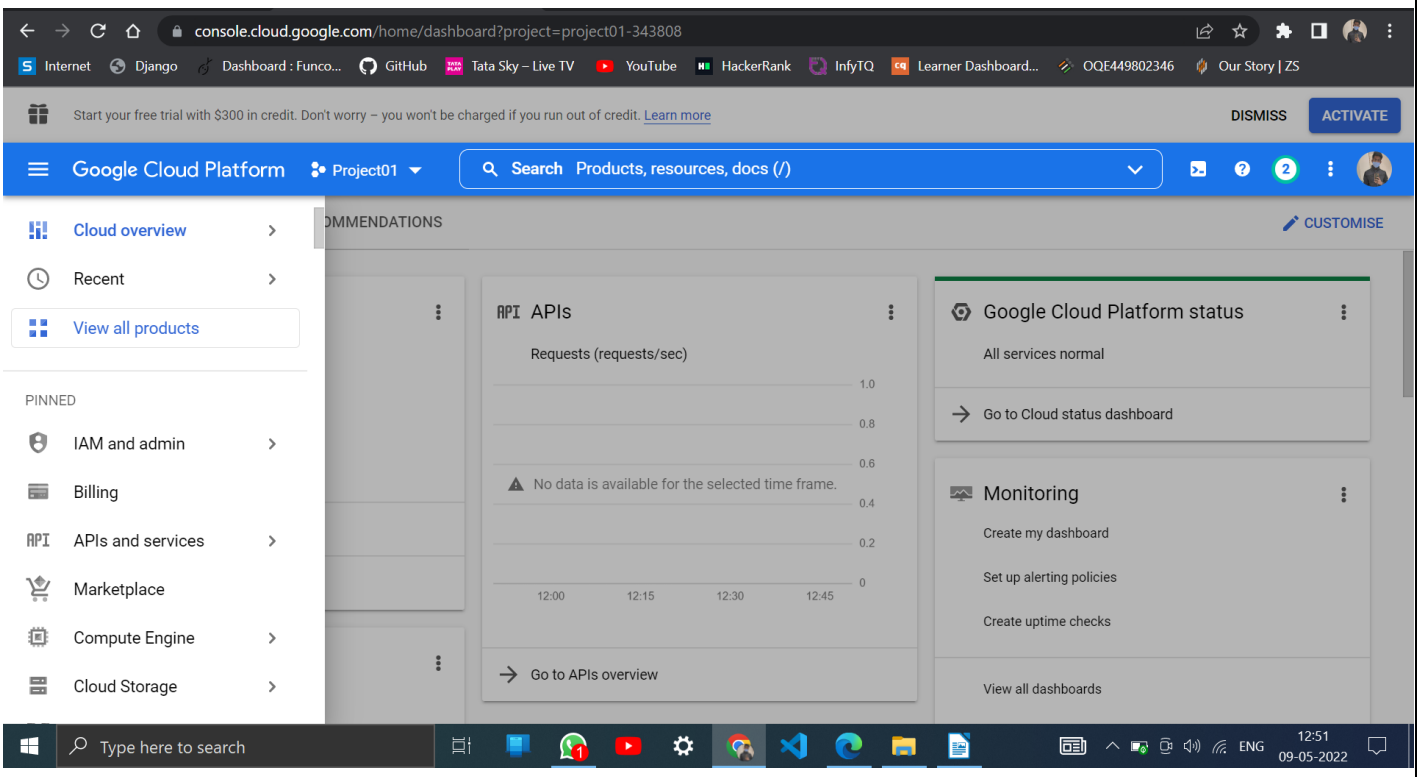
Step 3 : Google Cloud Platform is shown



Step 4 : Click Dashboard in the Google Cloud Plaform

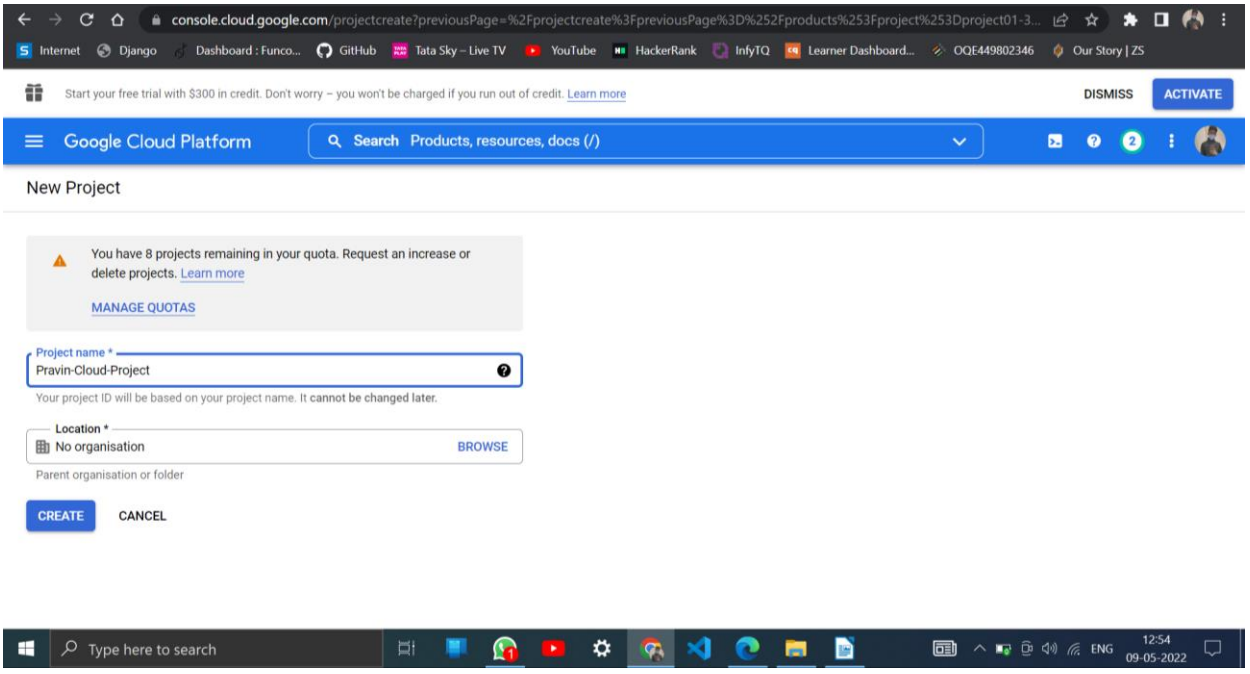


Step 5 : Dashboard in the Google Cloud Plaform

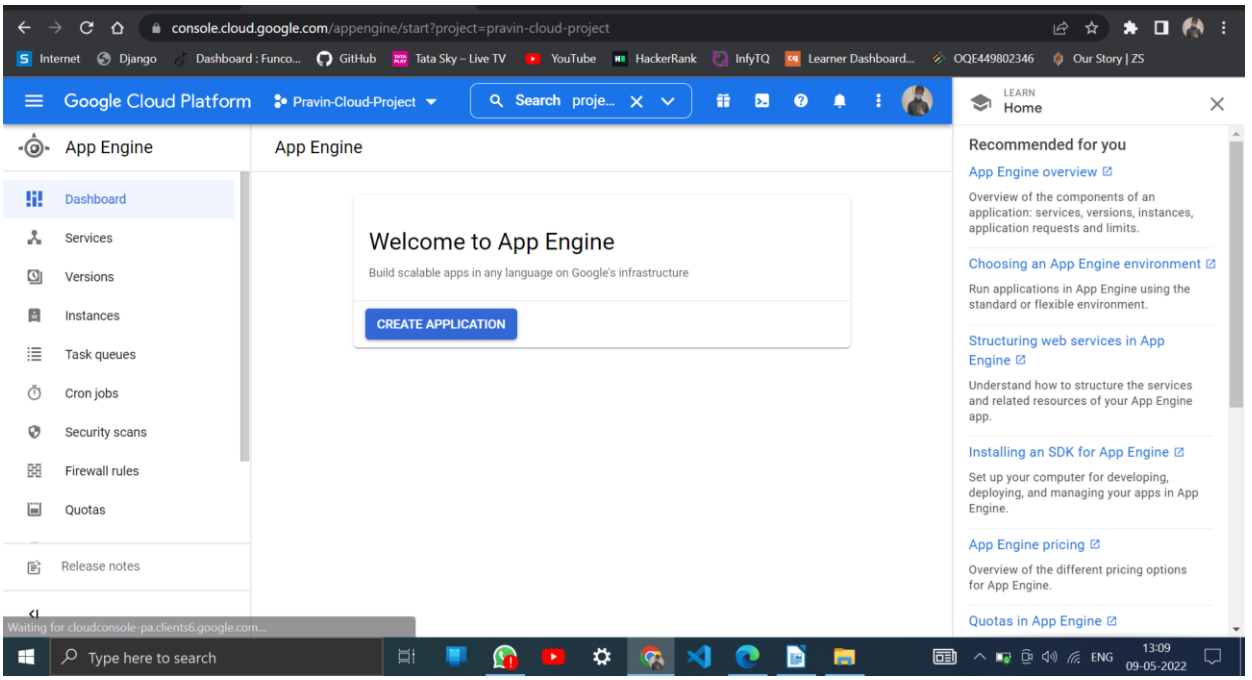


Step 6 : Click New Project and give unique Project Name.

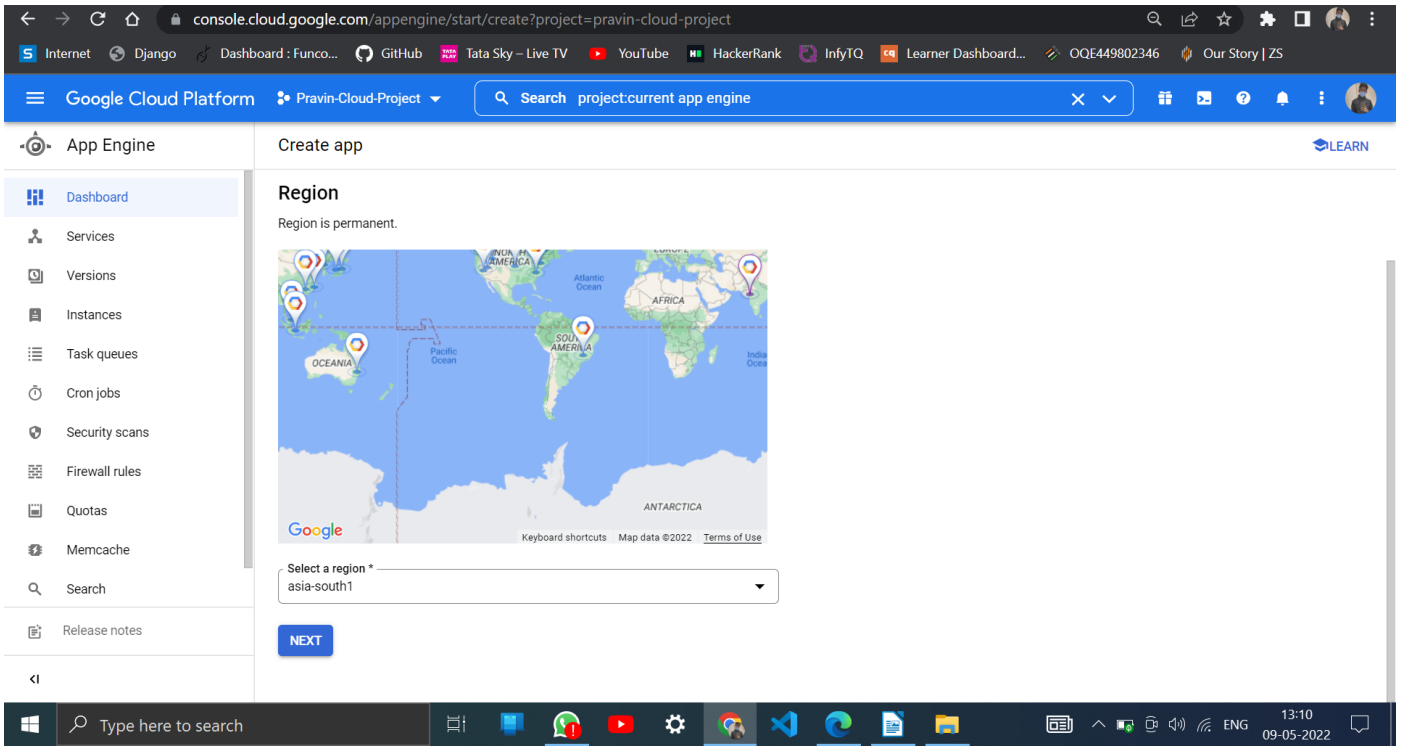
Example : pravin-cloud-project



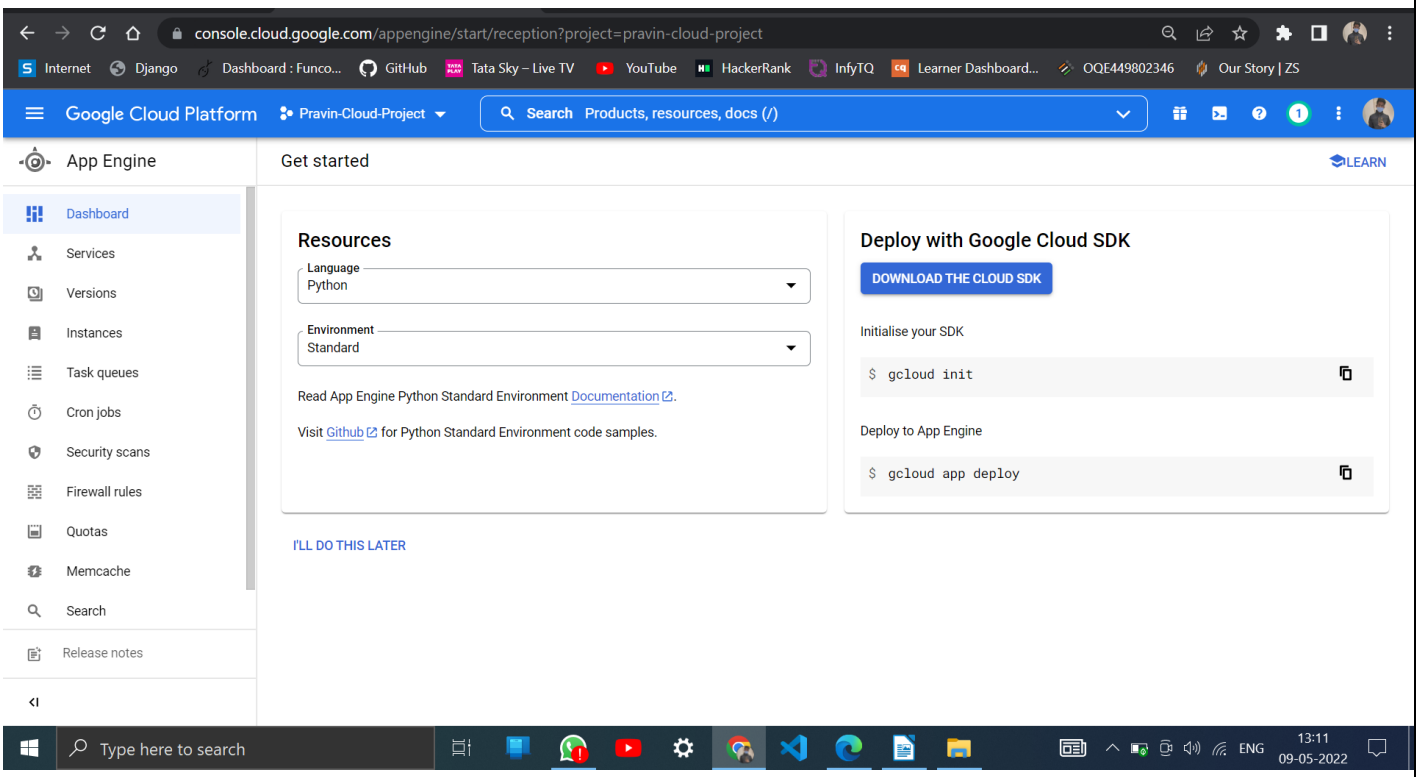
Step 7 : Google App Engine is initated



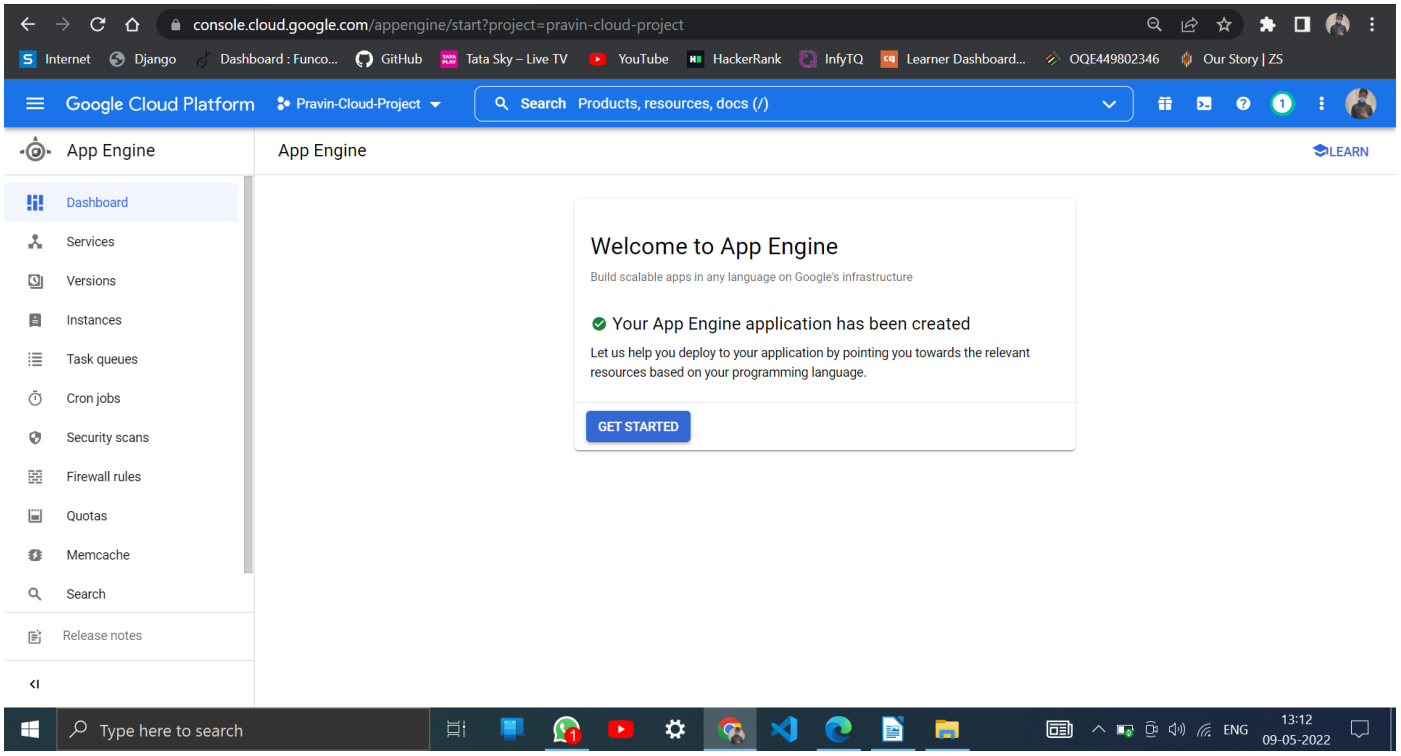
Step 8 : Click create Application



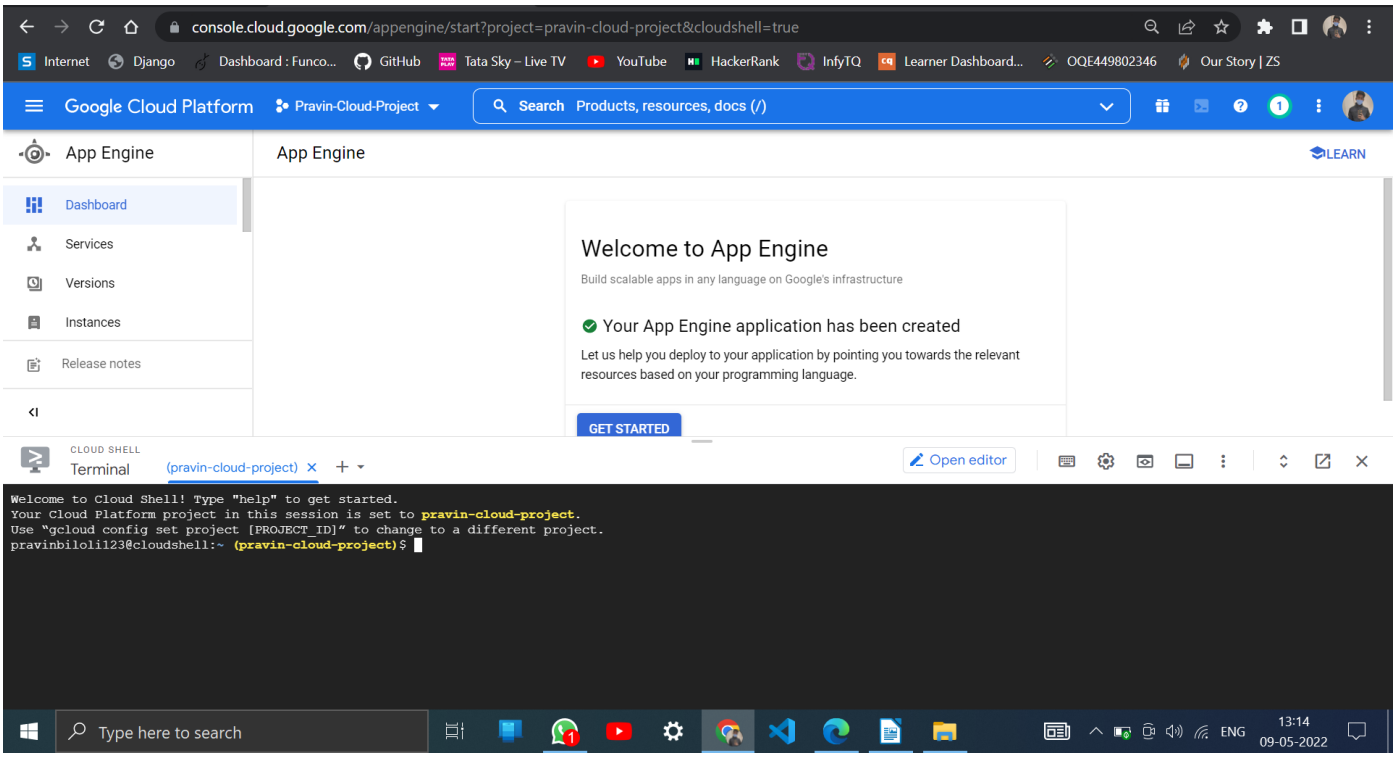
Step 9 : Create app and Select Language Python



Step 10 : Python app is created in Google App Engine

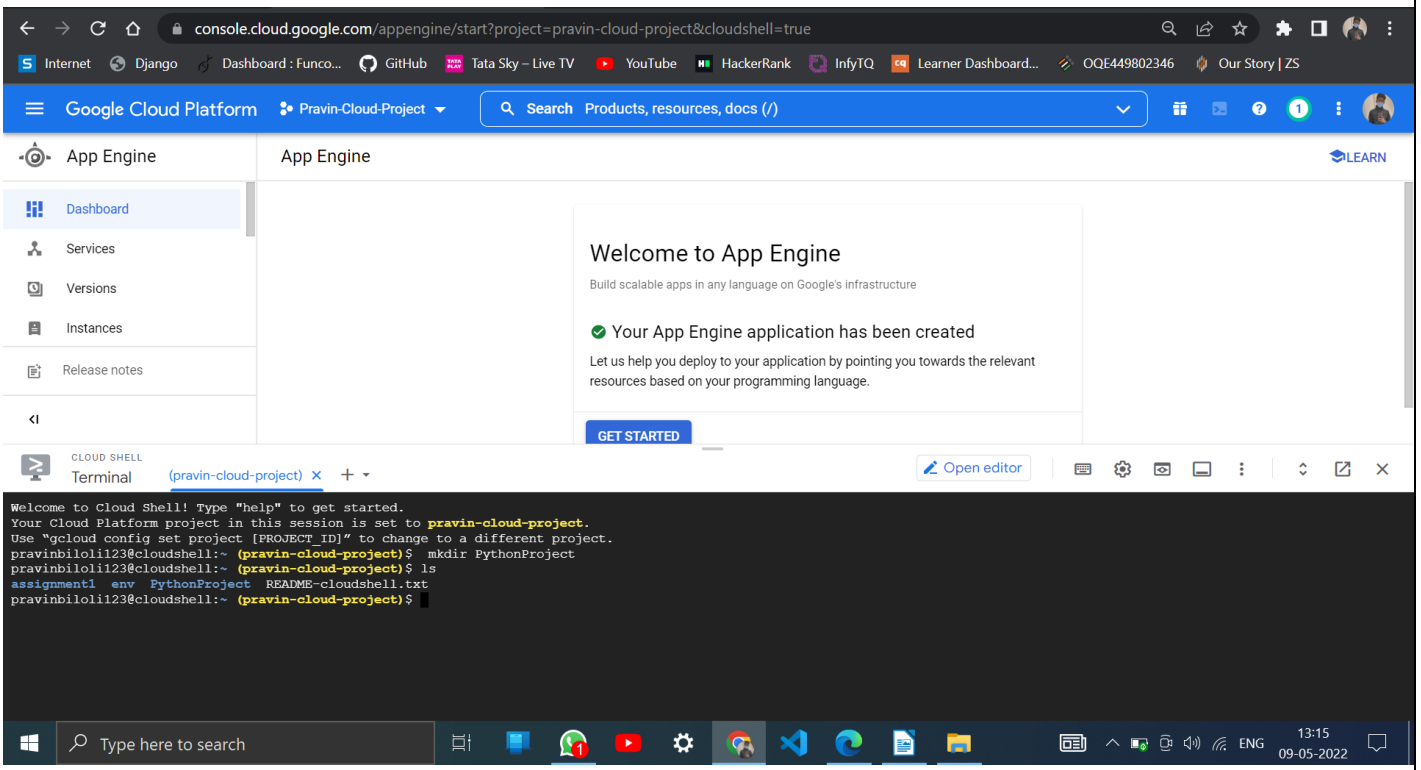


Step 12 : Click Cloud Shell in the Pravin-Cloud-Project

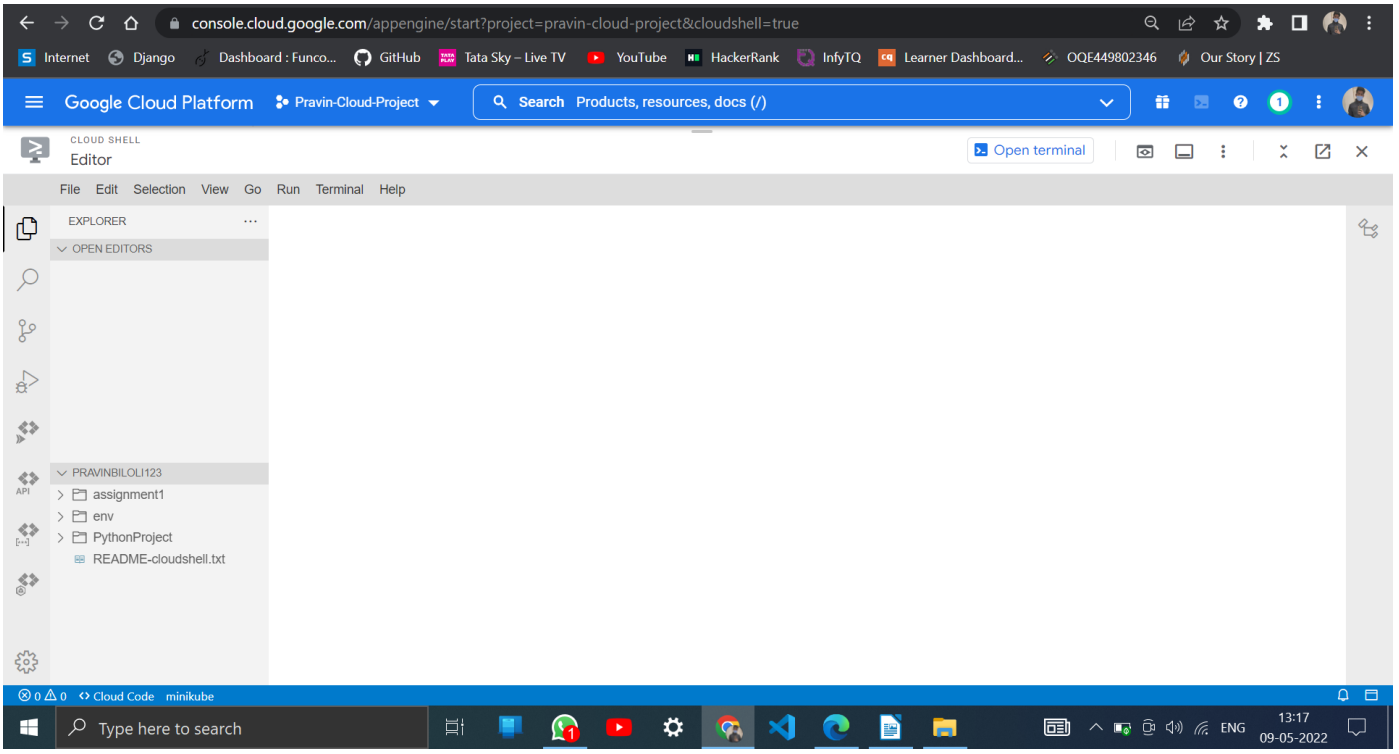


Step 13 : Create a Directory PythonProject using mkdir command

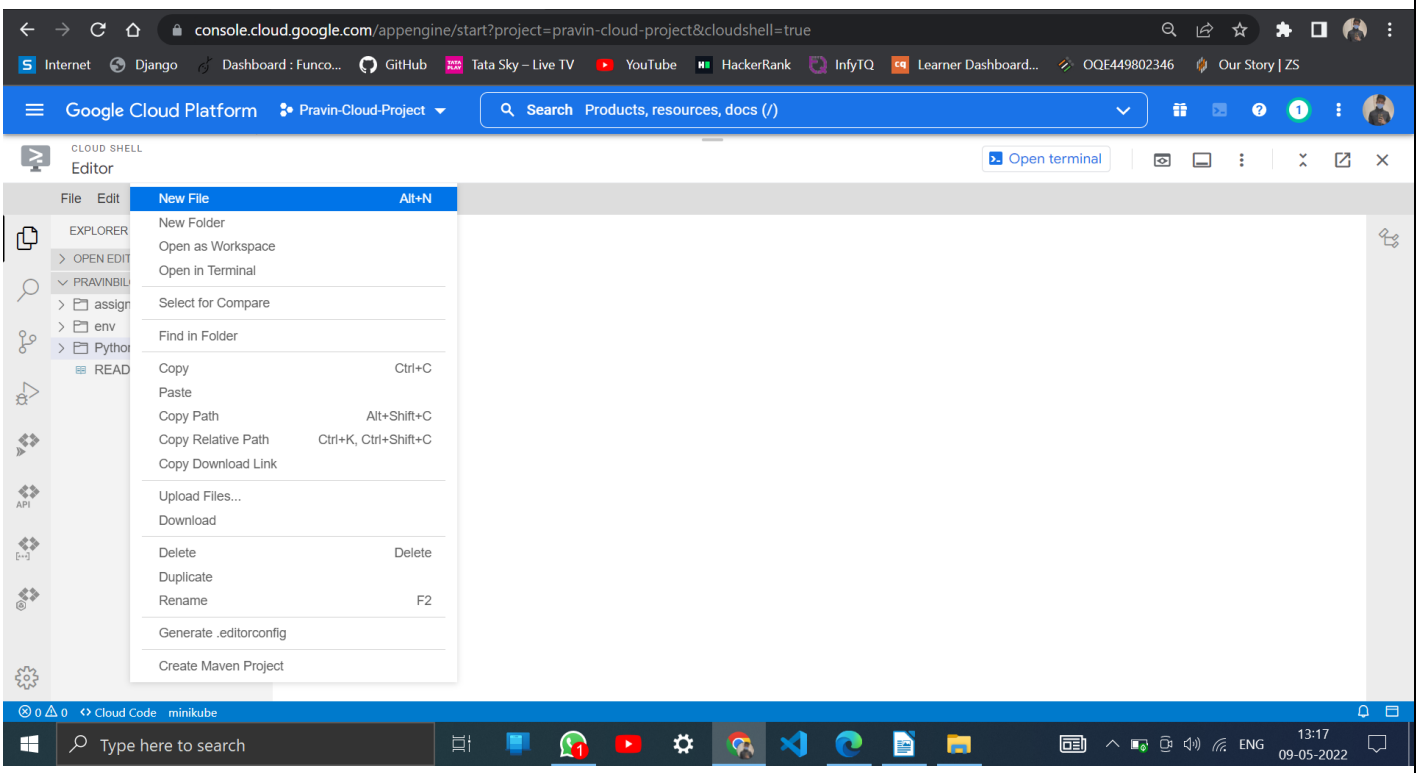
Syntax : mkdir PythonProject



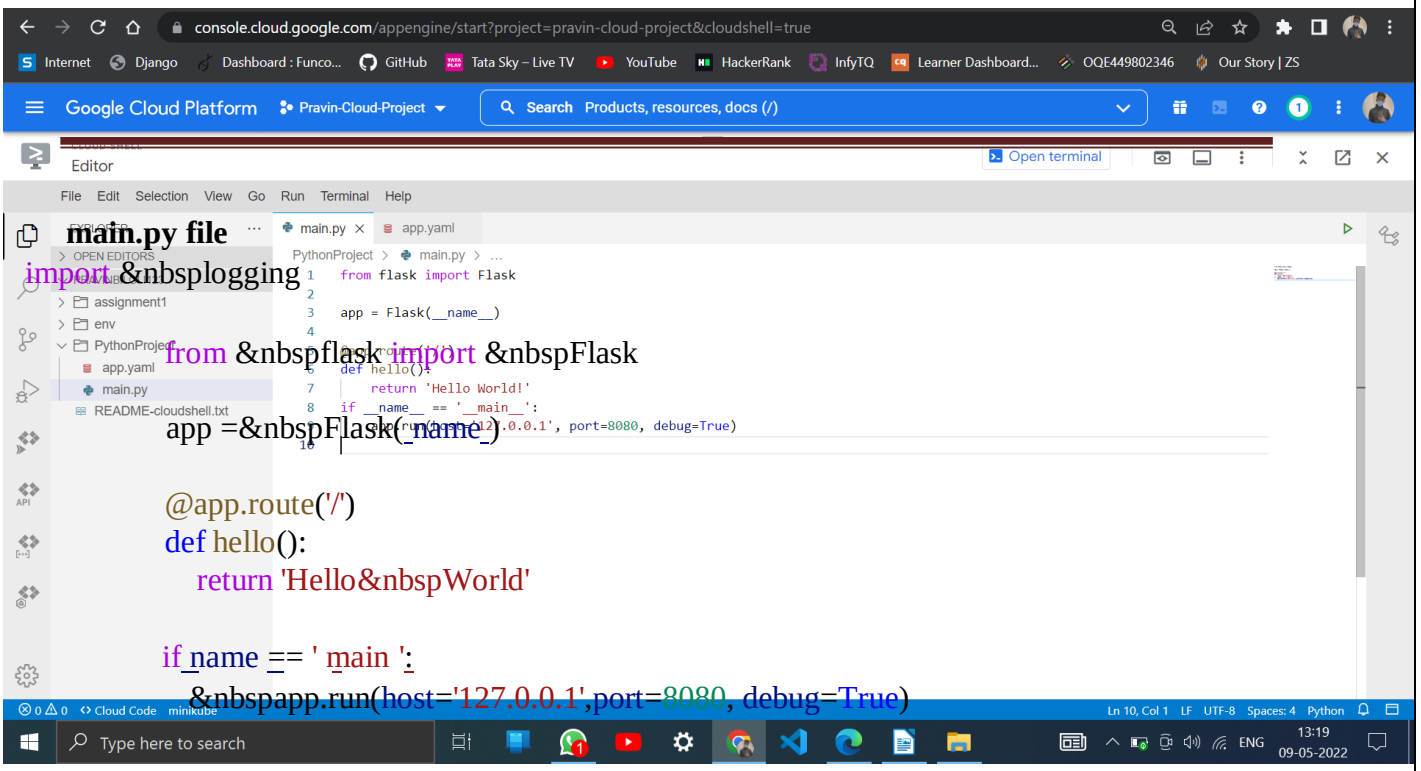
Step 14 : Click Editor to create Python application



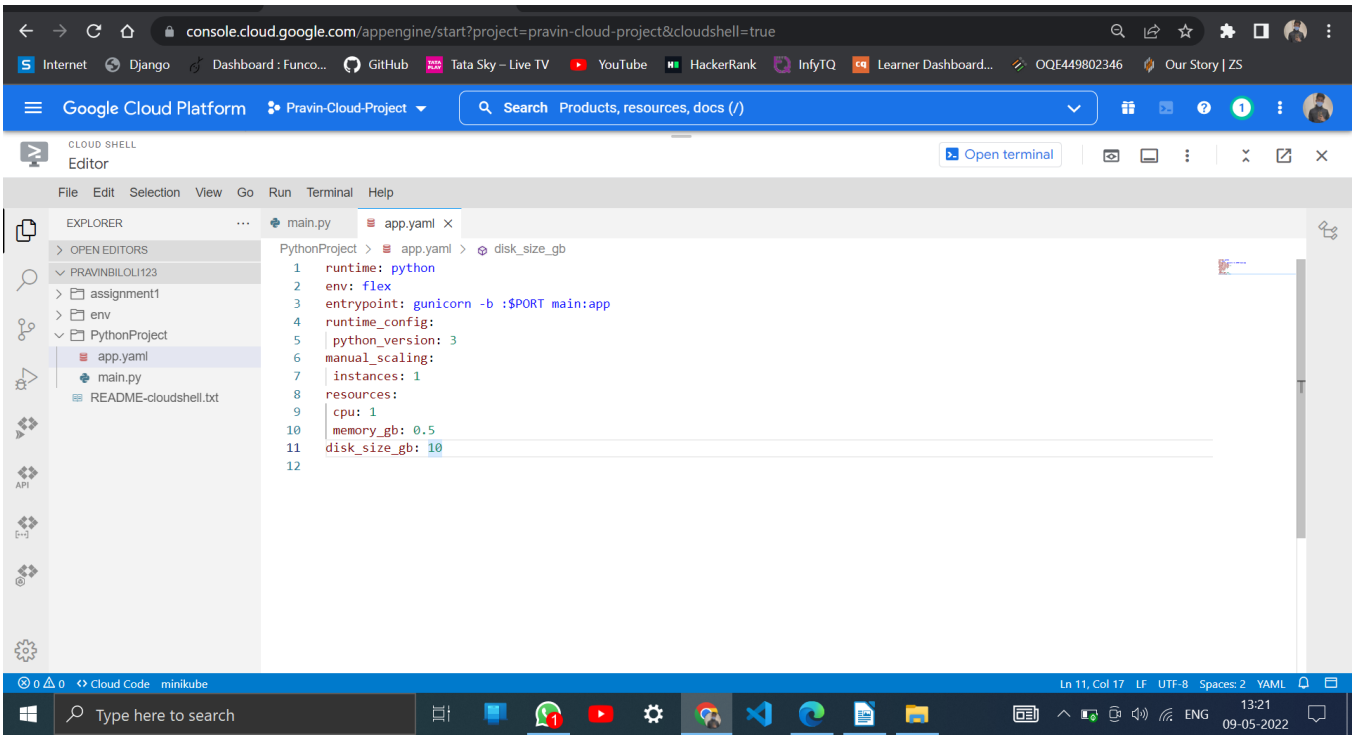
Step 15 : Click New File in the PythonProject Folder (Python file)



Step 16 : Create main.py file

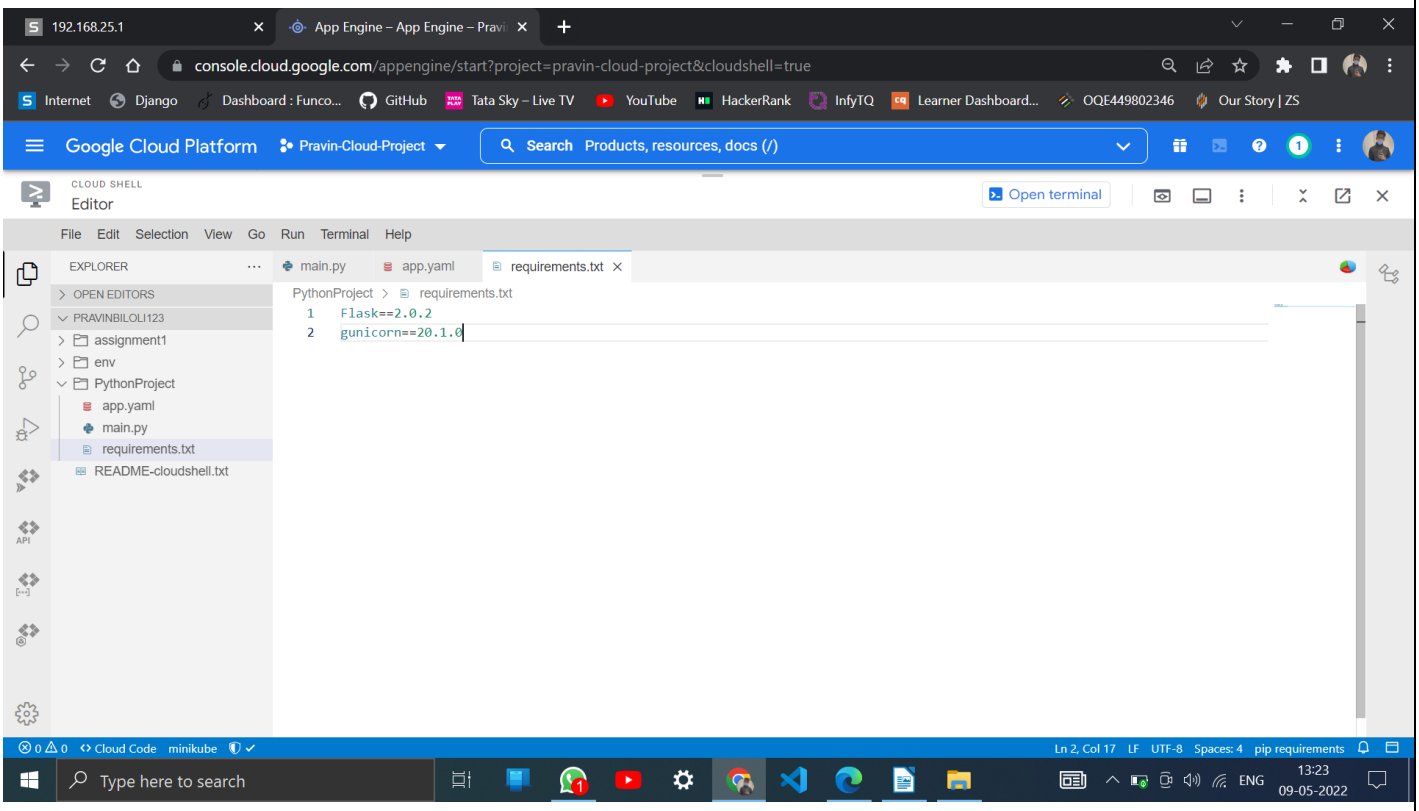


Step 17 : Create app.yaml file app.yaml




```
runtime: python
env: flex
entrypoint: gunicorn -b :$PORT&nbspmain:app

runtime_config:
    python_version: 3
```

Step 18 : Create requirements.txt file

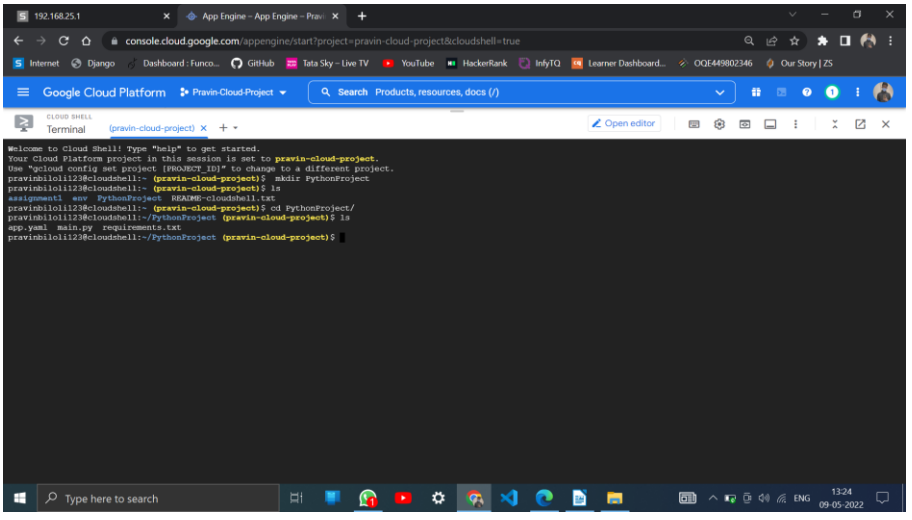


requirements.txt

Flask==0.11.1

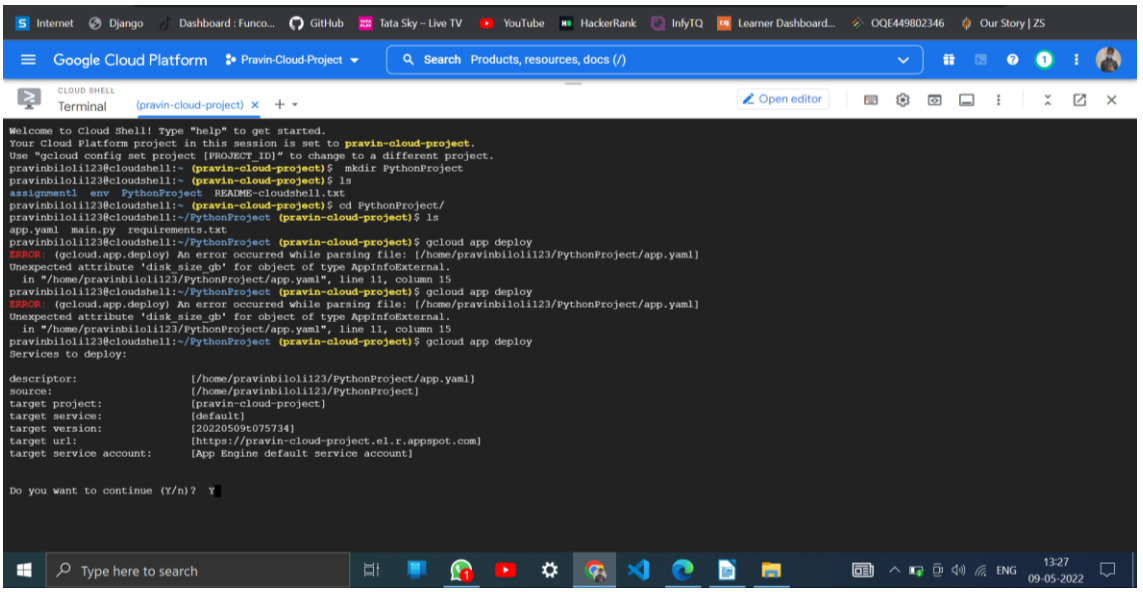
gunicorn==19.6.0

Step 19 : Move to Cloud Shell Environment to run the application

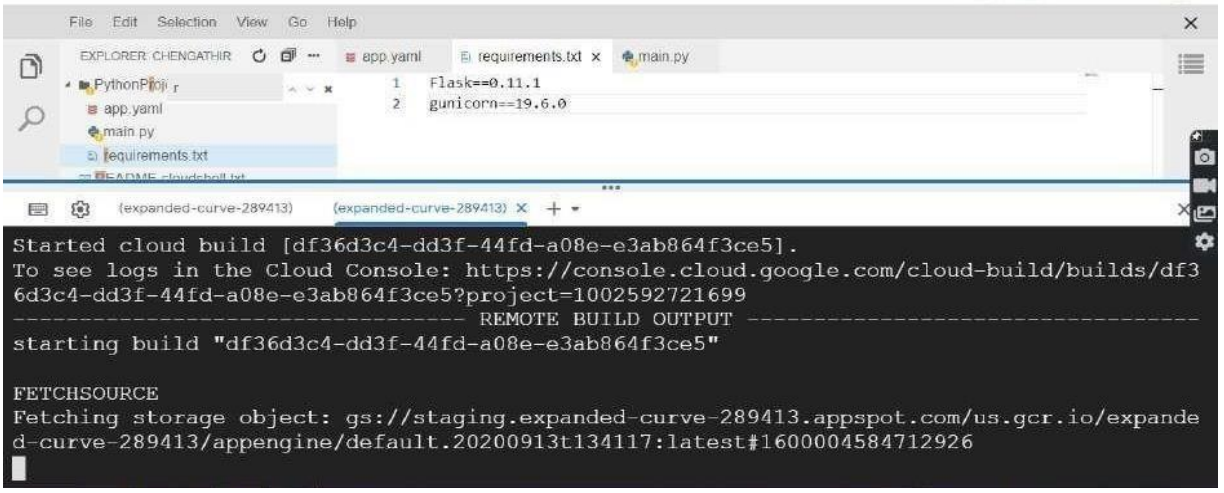


Step 20 : Move to Cloud Shell Environment to run the

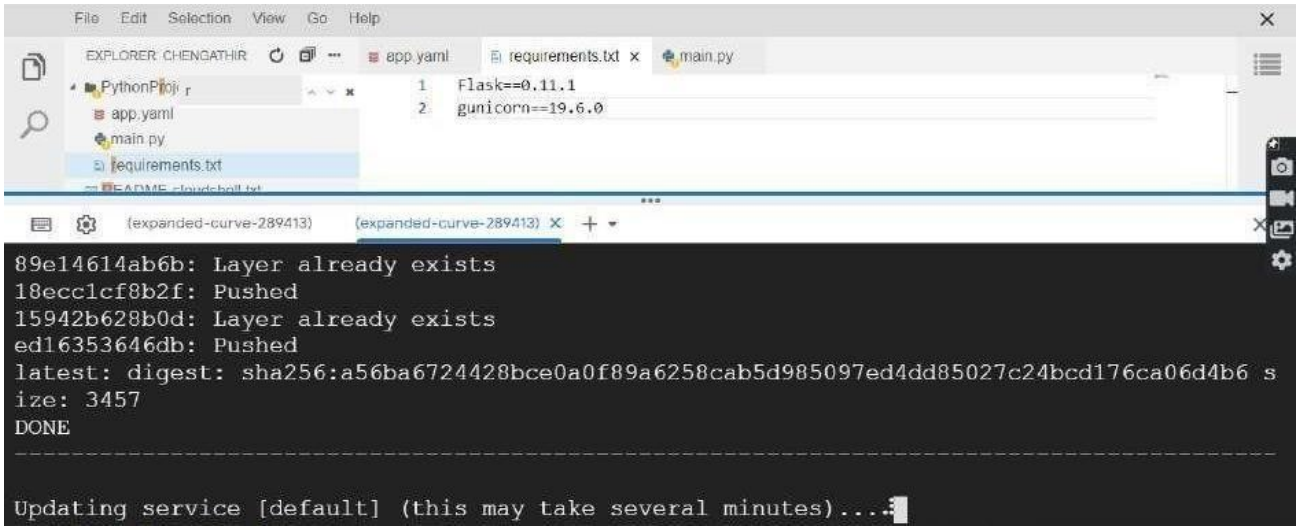
applicationSyntax: gcloud app deploy



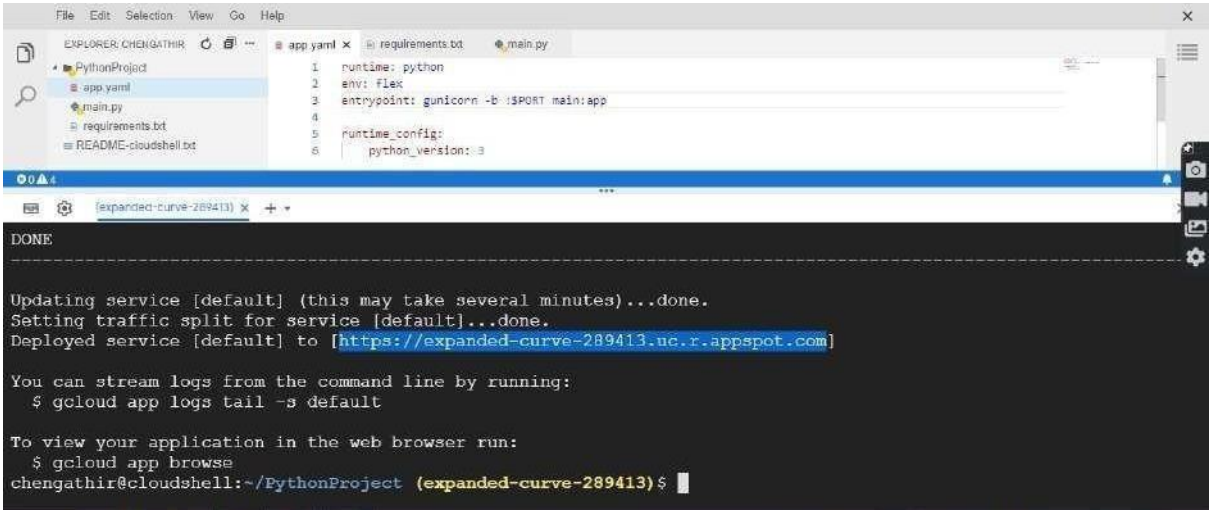
Continue the application. It enable service on the given project



It started building the object and fetching the storage object for the created application



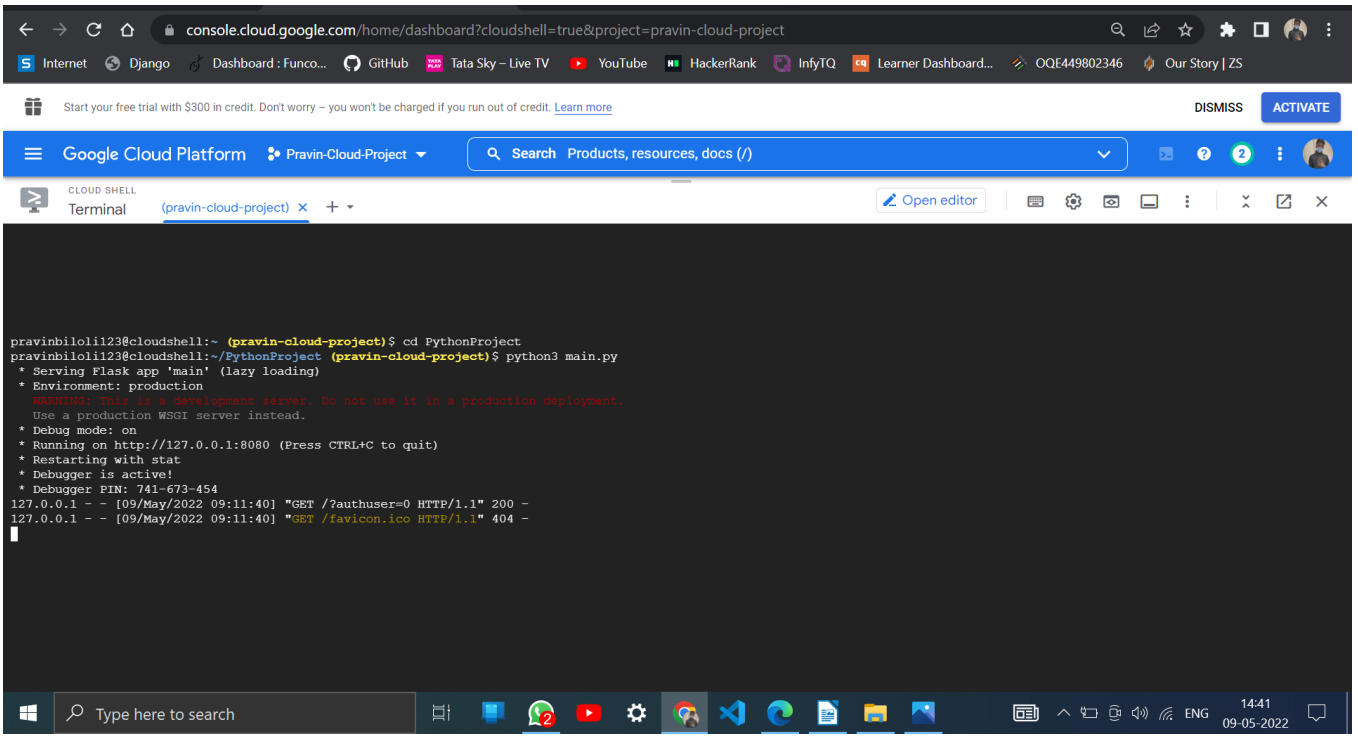
It is updating the service



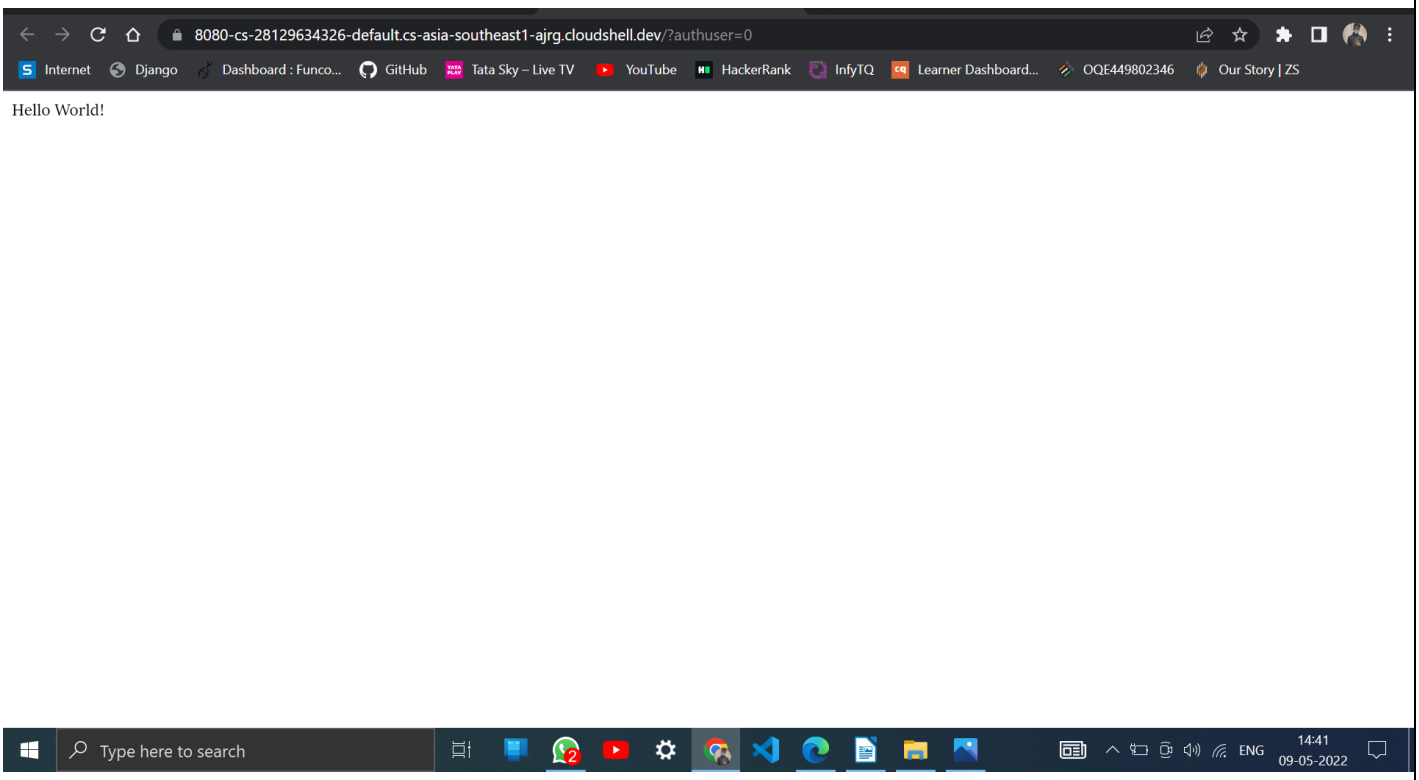
The application is successfully deployed and URL is

https://expanded-curve-289413.uc.r.appspot.com

Step 21 : Run your program in the broswer


console.cloud.google.com/home/dashboard?cloudshell=true&project=pravin-cloud-project
Internet
Django
Dashboard : Funco...
GitHub
Tata Sky – Live TV
YouTube
HackerRank
InfyTQ
Learner Dashboard...
OQE449802346
Our Story | ZS
Start your free trial with $300 in credit. Don't worry – you won't be charged if you run out of credit. Learn more
DISMISS
ACTIVATE
Google Cloud Platform
Pravin-Cloud-Project
Search Products, resources, docs (/)
CLOUD SHELL
Terminal
(pravin-cloud-project)
Open editor
pravinbiloli123@cloudshell:~ (pravin-cloud-project)$ cd PythonProject
pravinbiloli123@cloudshell:~/PythonProject (pravin-cloud-project)$ python3 main.py
* Serving Flask app 'main' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:8080 (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 741-673-454
127.0.0.1 - - [09/May/2022 09:11:40] "GET /?authuser=0 HTTP/1.1" 200 -
127.0.0.1 - - [09/May/2022 09:11:40] "GET /favicon.ico HTTP/1.1" 404 -
Type here to search
ENG
14:41
09-05-2022

Step 22 : Hello World Program is sucessfully run in the browser



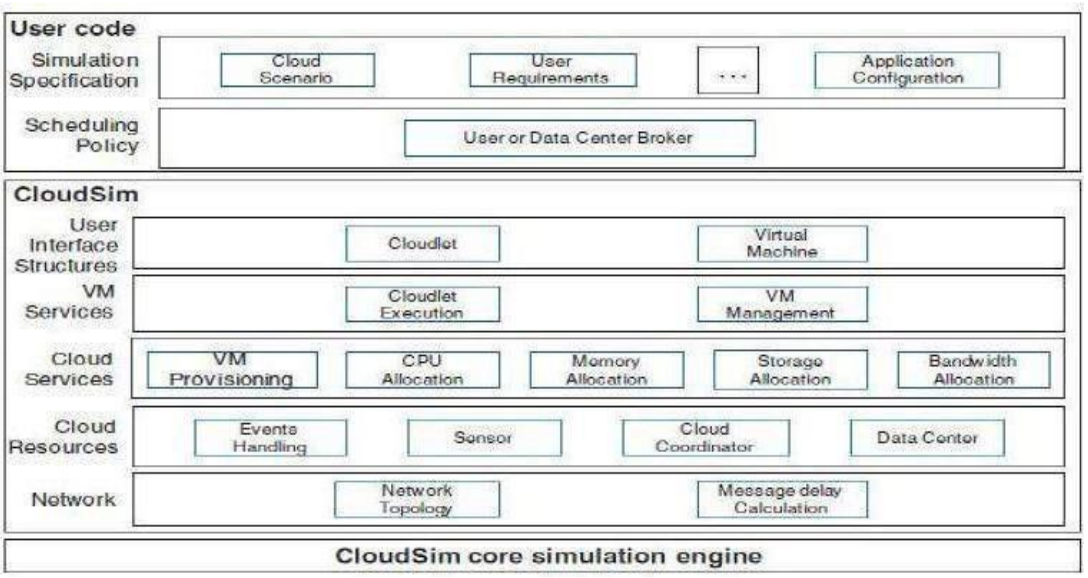
Result:

Thus the Google App Engine is installed successfully and a web application to display hello world using python is developed and deployed in the GAE and used GAE Launcher to launch the web applications.

Ex No. 3 a	Simulate a cloud scenario using CloudSim

Introduction:

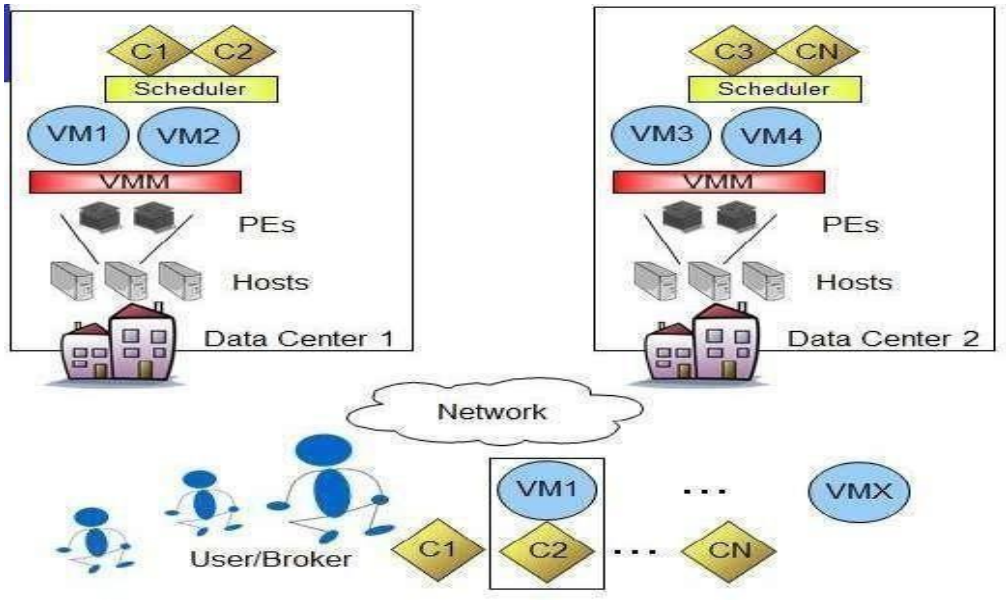
- **CloudSim**
  - A Framework for modeling and simulation of Cloud Computing Infrastructures and services
  - Originally built at the Cloud Computing Distributed Systems (CLOUDS) Laboratory, The University of Melbourne, Australia
  - It is completely written in JAVA
- **Main Features of CloudSiM** ○ Modeling and simulation
  - ○ Data centre network topologies and message-passing applications ○ Dynamic insertion of simulation elements ○ Stop and resume of simulation
  - ○ Policies for allocation of hosts and virtual machines
- **Cloudsim – Essentials**
  - JDK 1.6 or above http://tinyurl.com/JNU-JAVA
  - Eclipse 4.2 or above http://tinyurl.com/JNU-Eclipse
  - Alternatively NetBeanshttps://netbeans.org/downloads
  - Up & Running with cloudsim guide: https://goo.gl/TPL7Zh
- **Cloudsim-Directory structure**
  - cloudsim/ -- top level CloudSim directory
  - docs/ -- CloudSim API Documentation
  - examples/ -- CloudSim examples
  - jars/ -- CloudSim jar archives
  - sources/ -- CloudSim source code
- **Cloudsim - Layered Architecture**



✞ **Cloudsim - Component model classes** ○
CloudInformationService.java ○
Datacenter.java,Host.java,Pe.java ○
Vm.java,Cloudlet.java ○ DatacenterBroker.java
- ○ Storage.java,HarddriveStorage.java, SanStorage.java

✞ **Cloudsim - Major blocks/Modules** ○
org.cloudbus.cloudsim ○
org.cloudbus.cloudsim.core ○
org.cloudbus.cloudsim.core.predicates ○
org.cloudbus.cloudsim.distributions ○
org.cloudbus.cloudsim.lists ○
org.cloudbus.cloudsim.network ○
org.cloudbus.cloudsim.network.datacenter ○
org.cloudbus.cloudsim.power ○
org.cloudbus.cloudsim.power.lists ○
org.cloudbus.cloudsim.power.models ○
org.cloudbus.cloudsim.provisioners
- ○ org.cloudbus.cloudsim.util

✞ **Cloudsim - key components** ○ Datacenter ○
DataCenterCharacteristics ○ Host ○
DatacenterBroker ○ RamProvisioner ○
BwProvisioner ○ Storage
- ○ Vm
- ○ VMAllocationpolicy ○ VmScheduler ○ Cloudlet ○ CloudletScheduler ○ CloudInformationService ○ CloudSim ○ CloudSimTags ○ SimEvent ○ SimEntity ○ CloudsimShutdown ○ FutureQueue ○ DefferedQueue ○ Predicate and associative classes.

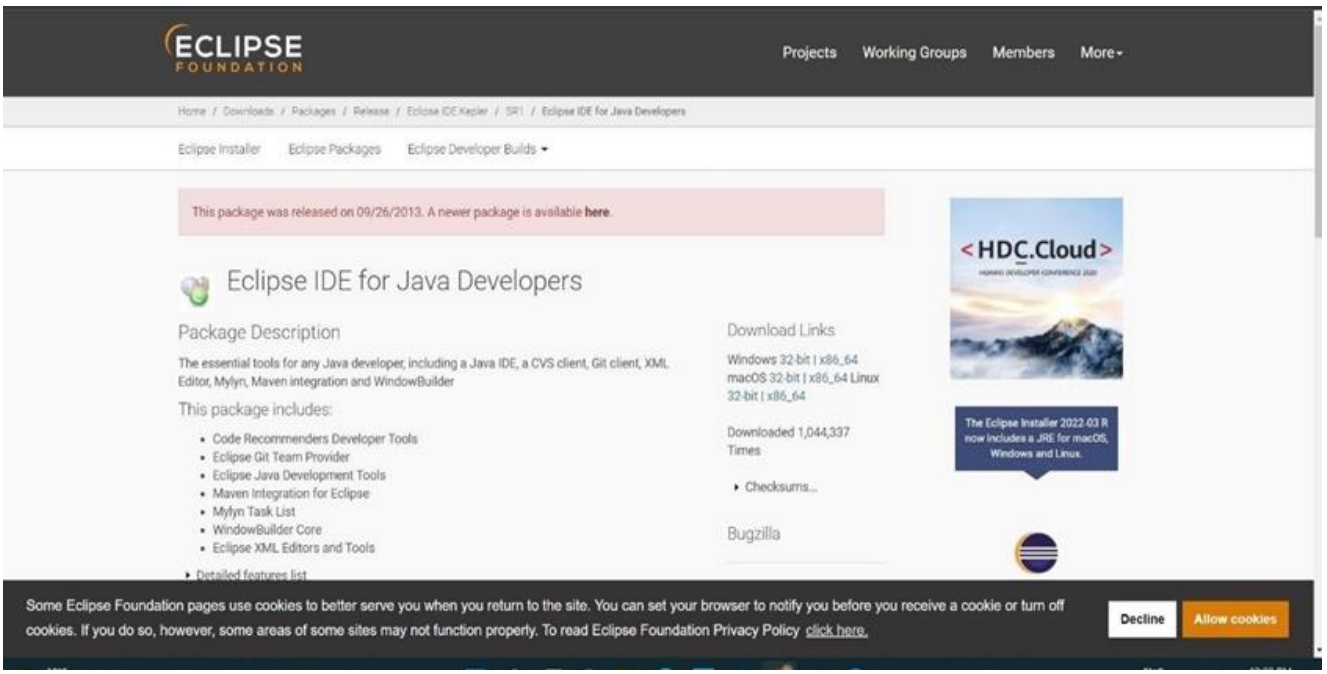


CloudSim Elements/Components

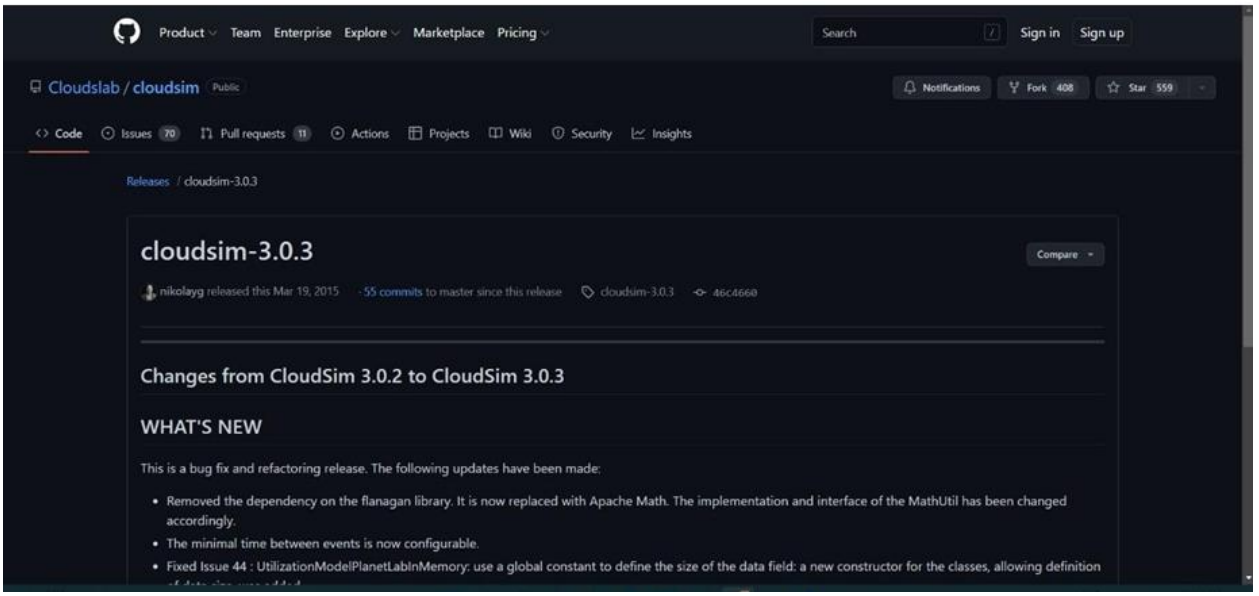
Procedure to import Eclipse, Cloudsim in your system

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

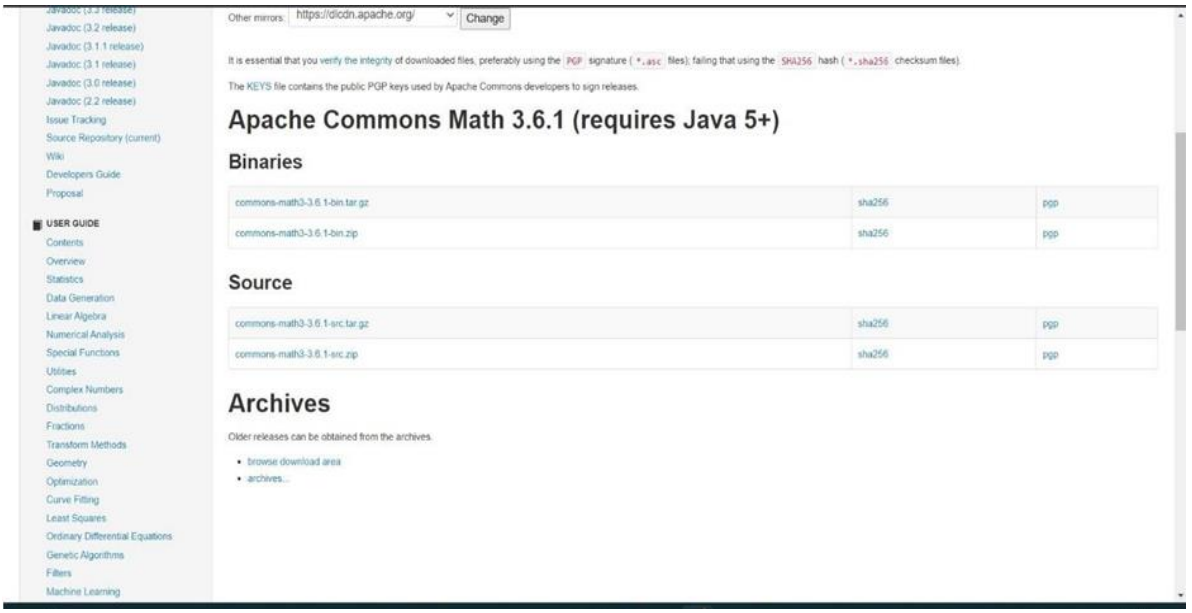
https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers



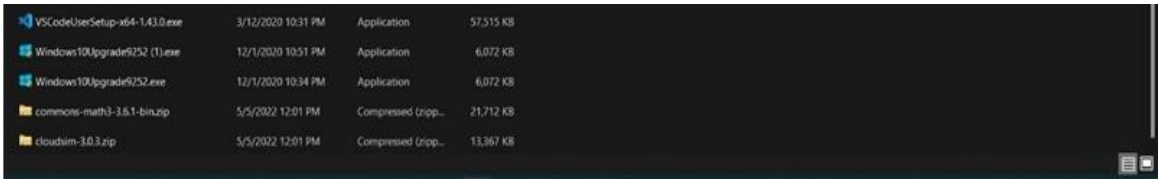
Step 2: Download cloudsim-3.0.3 from git hub repository in your local machine
https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3



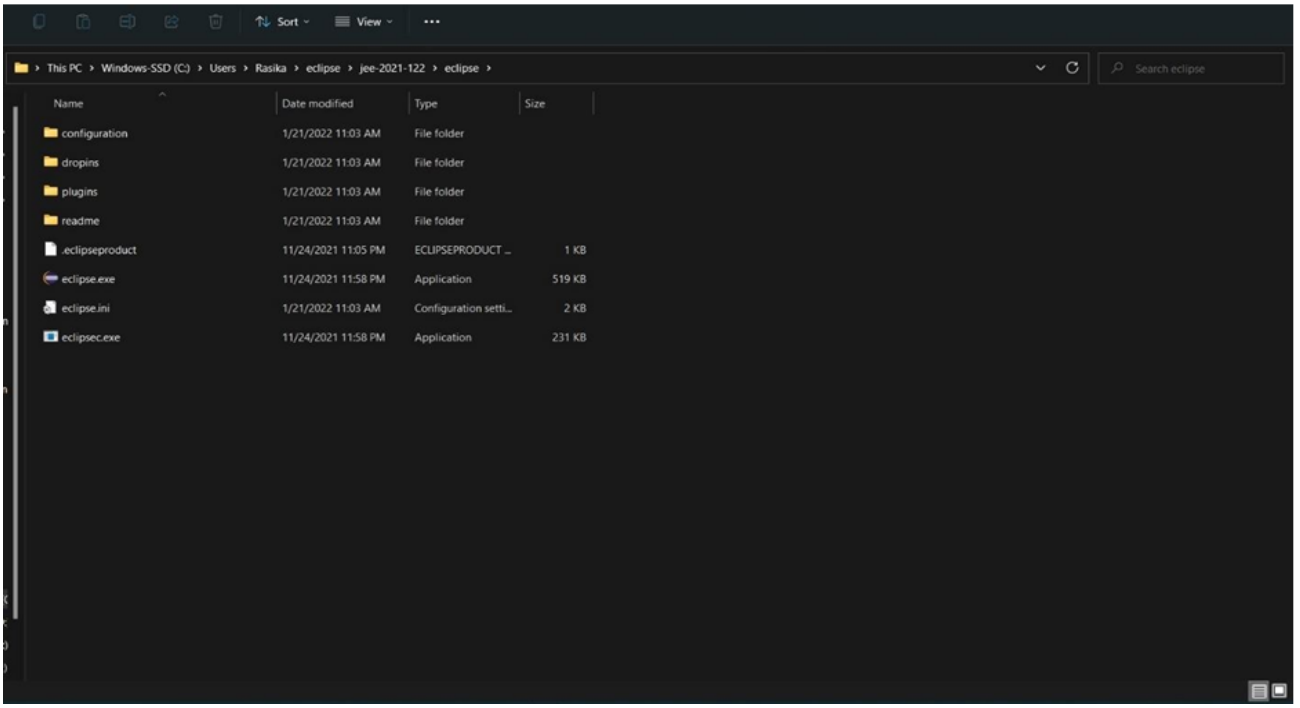
Step 3: Download commons-maths3-3.6.1 from git hub repository in your local machinehttps://commons.apache.org/proper/commons-math/download_math.cgi



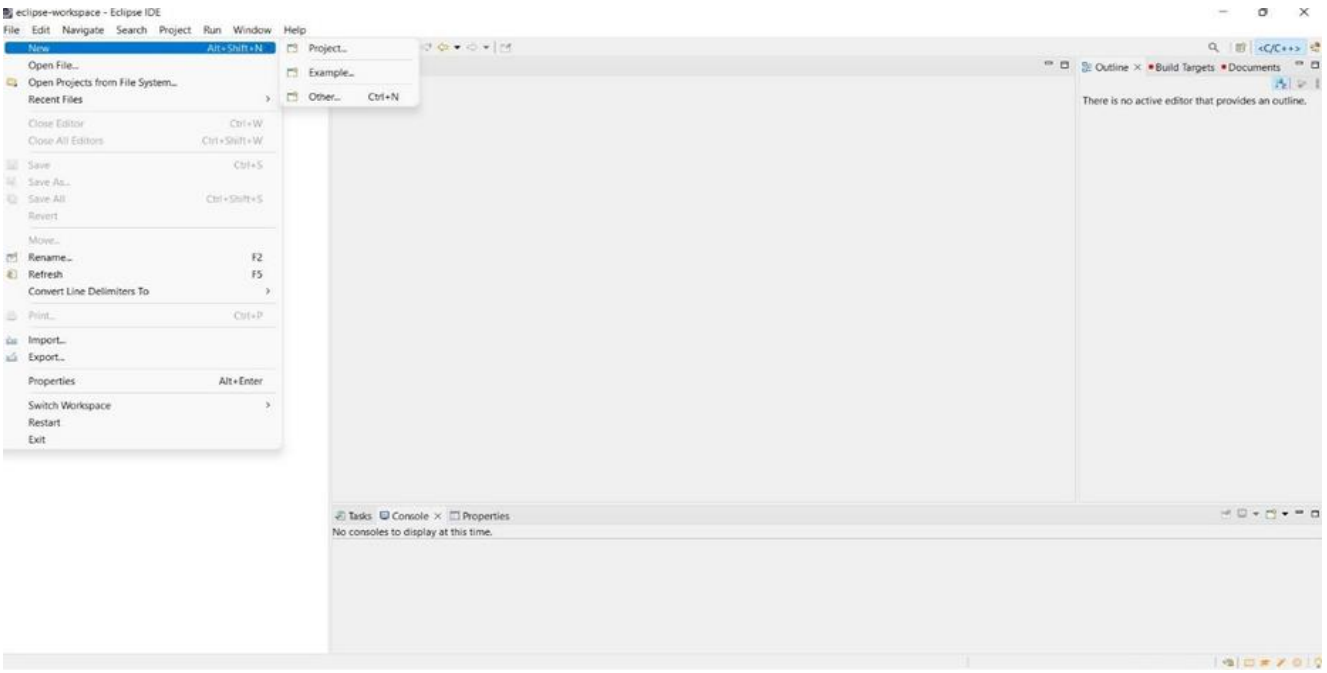
Step 4: Downloaded Eclipse, cloudsim-code-master and Apache Commons Math 3.6.1 in your local machine and extract cloudsim-3.0.3 and Apache Commons Math 3.6.1



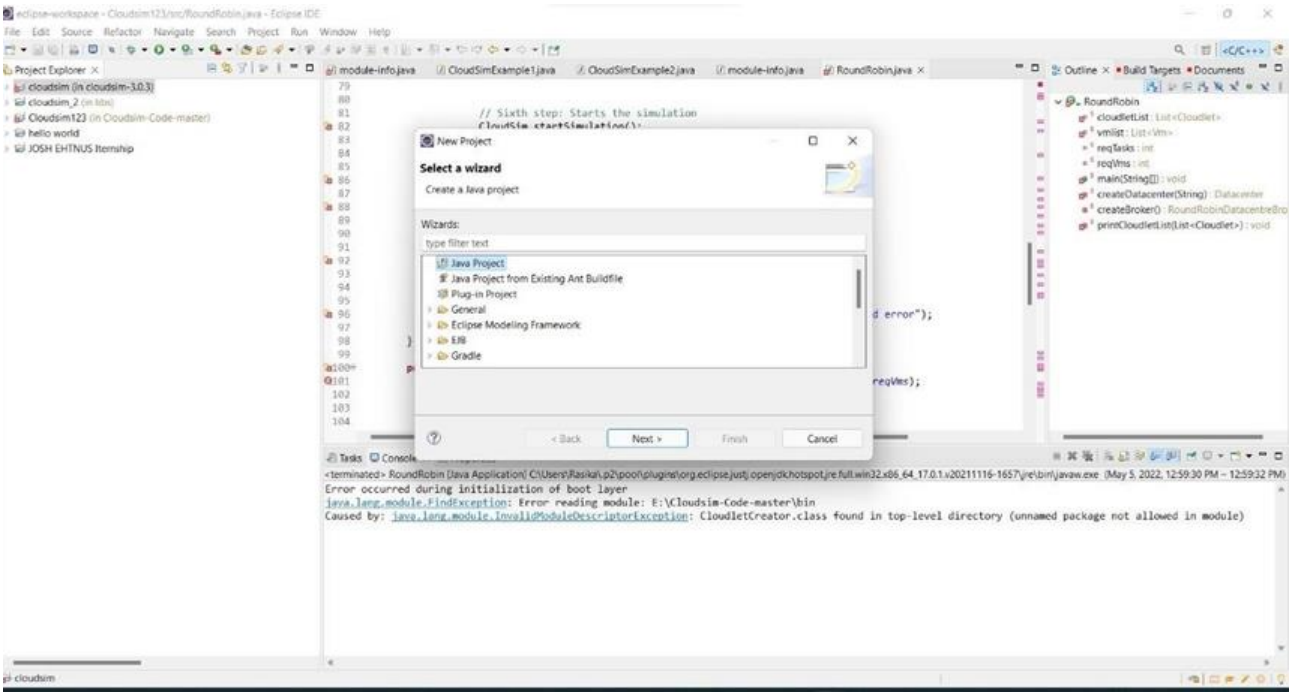
Step 5: First of all, navigate to the folder where you have unzipped the eclipse folder and open Eclipse.exe



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

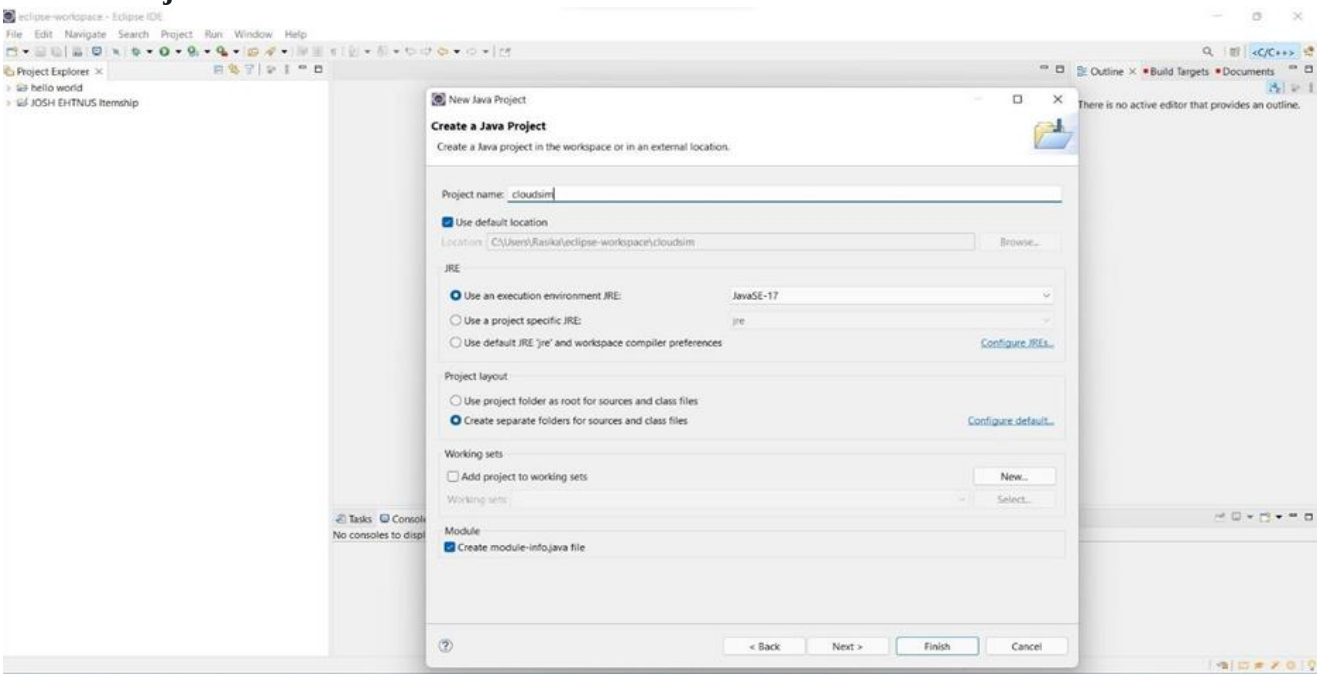


Step 7: A _*New Project*_ wizard should open. There are a number of options displayed and you have to find & select the _*Java Project*_ option, once done click *'Next*_

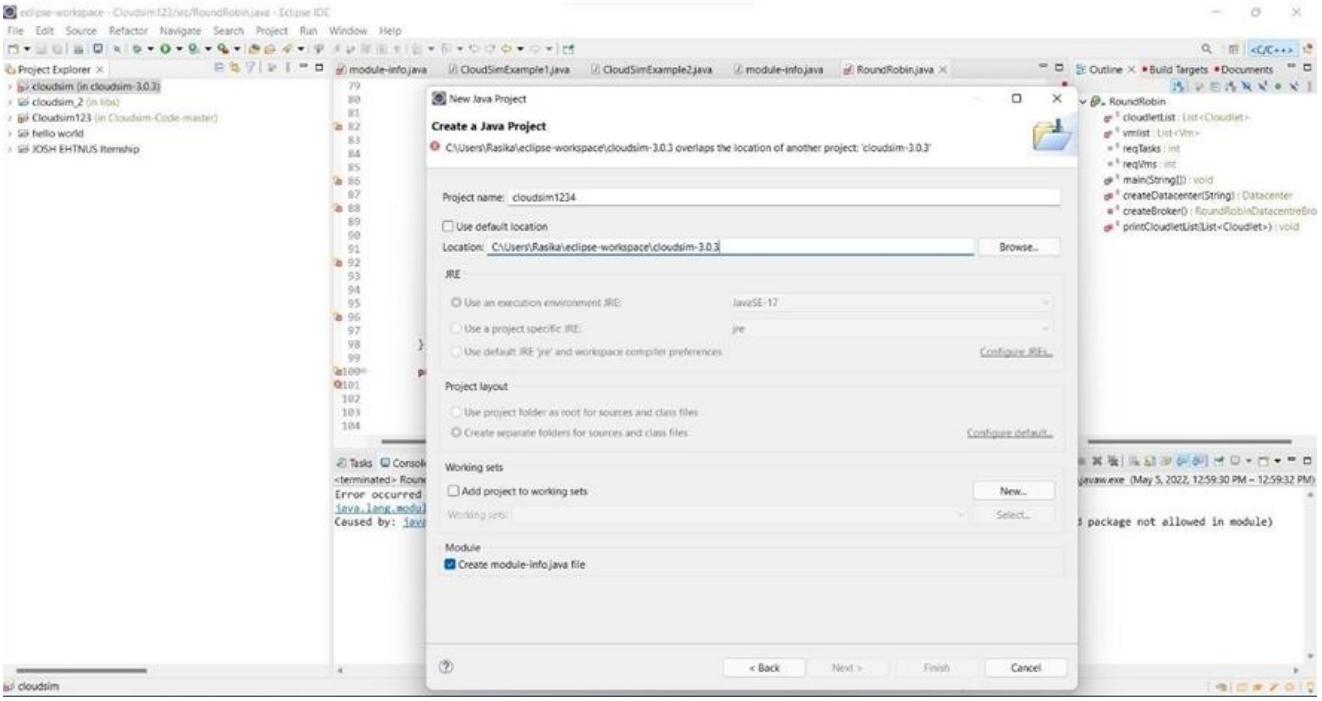


Step 8: Now a detailed new project window will open, here you will provide the project name and the path of CloudSim project source code, which will be done as follows:

Project Name: CloudSim.

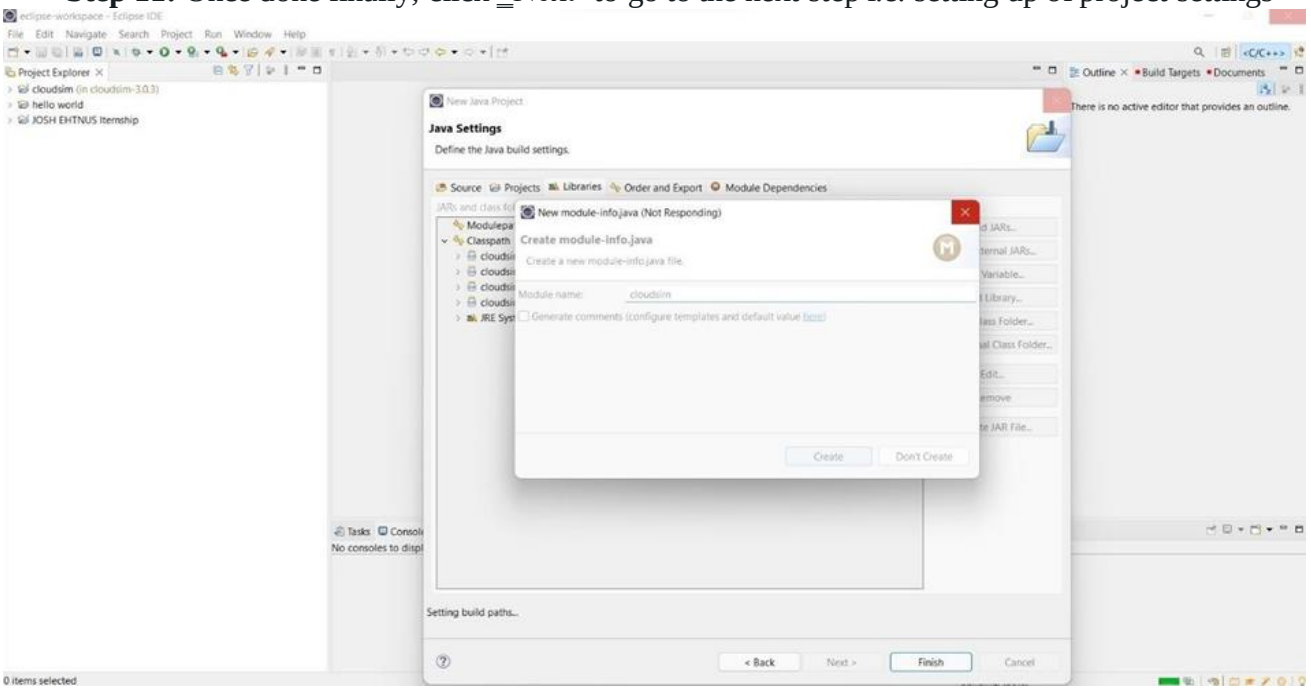


Step 9: Unselect the *'Use default location'* option and then click on *'Browse'* to open the path where you have unzipped the Cloudsim project and finally click Next to set project settings.

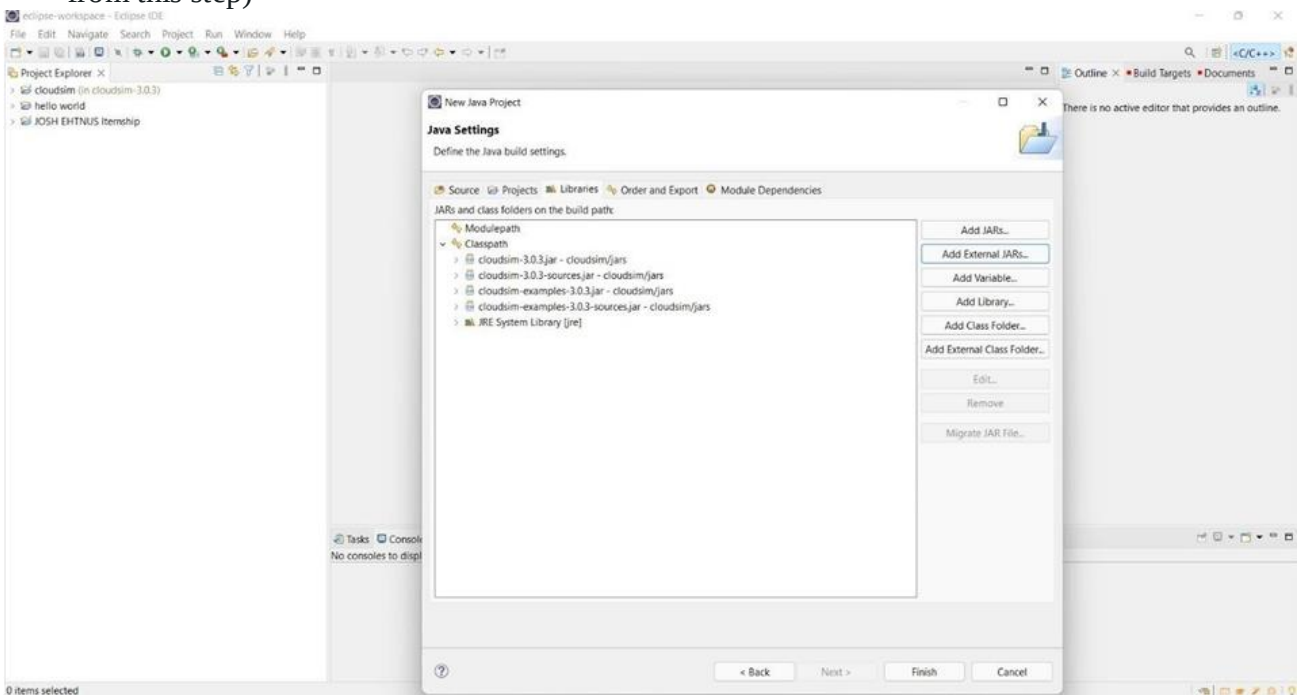


Step 10: Make sure you navigate the path till you can see the bin, docs, examplesetc folder in the navigation plane.

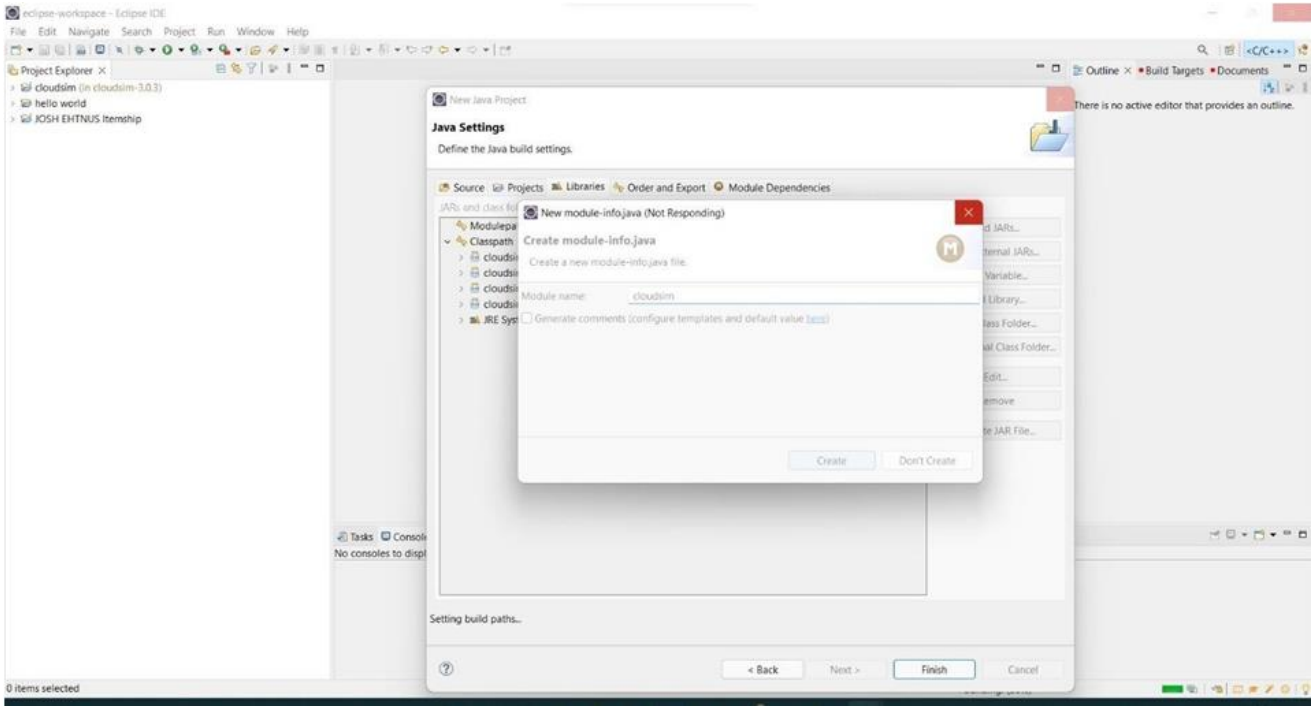
Step 11: Once done finally, click _Next‘ to go to the next step i.e. setting up of project settings



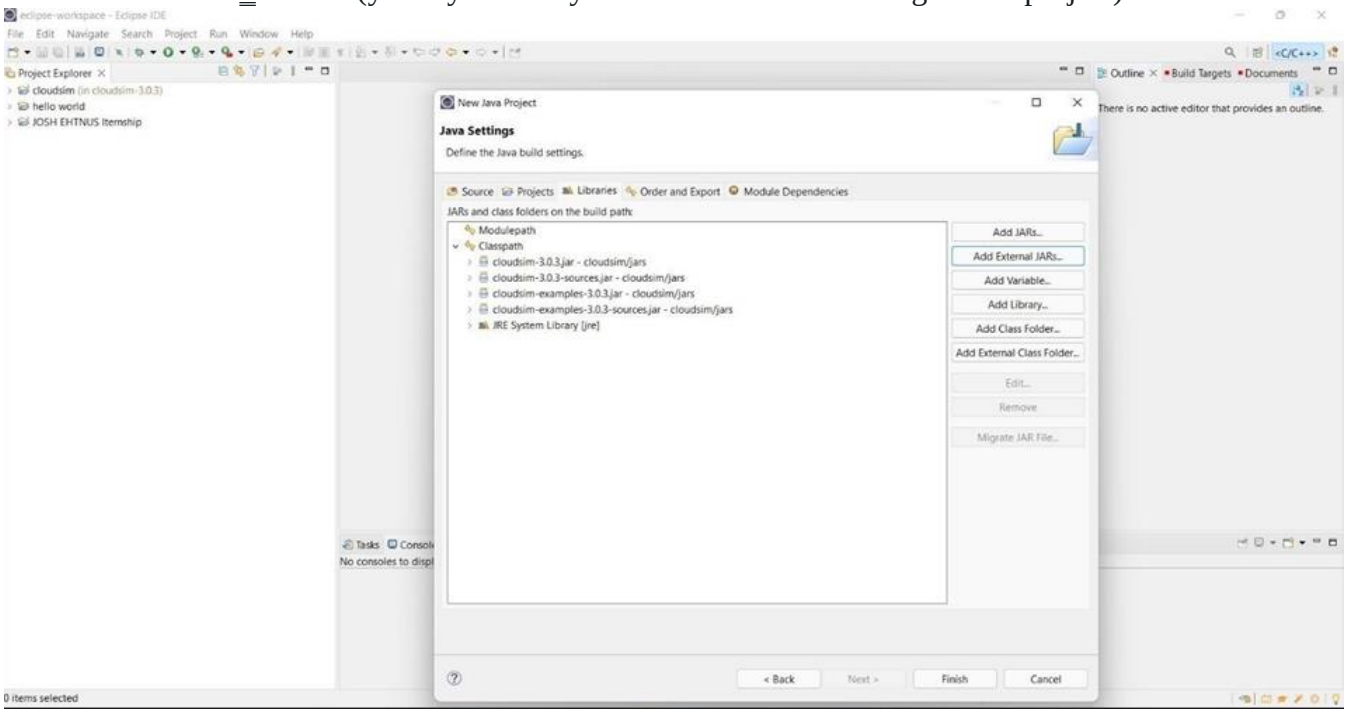
Step 12: Now open *‘Libraries’* tab and if you do not find commons-math3-3.x.jar *(here ‘x’ means the minor version release of the library which could be 2 or greater*) in the list then simply click on *_Add External Jar’* (commons-math3-3.x.jar will be included in the project from this step)



Step 13: Once you have clicked on _*Add External JAR's*_ Open the path where you have unzipped the commons-math binaries and select _*Commons-math3-3.x.jar*‘ and click on open.



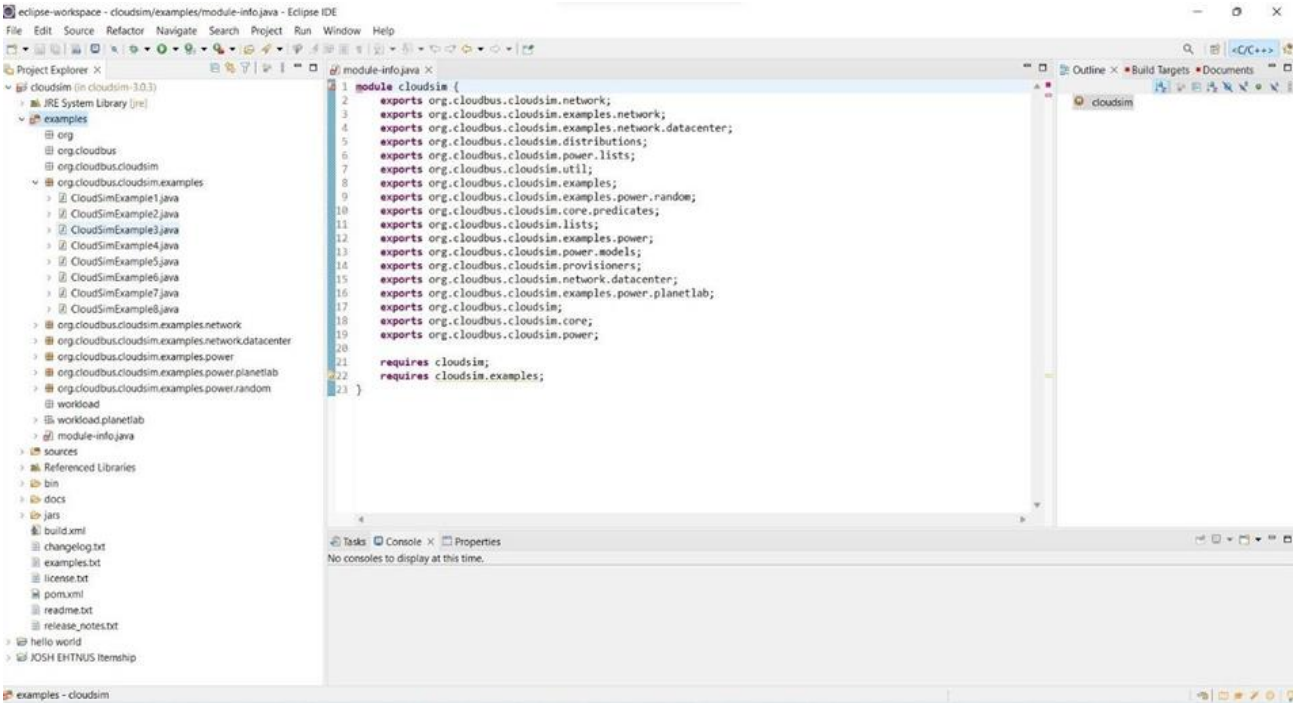
Step 14: Ensure external jar that you opened in the previous step is displayed in the list and then click on _*Finish*‘ (your system may take 2-3 minutes to configure the project)

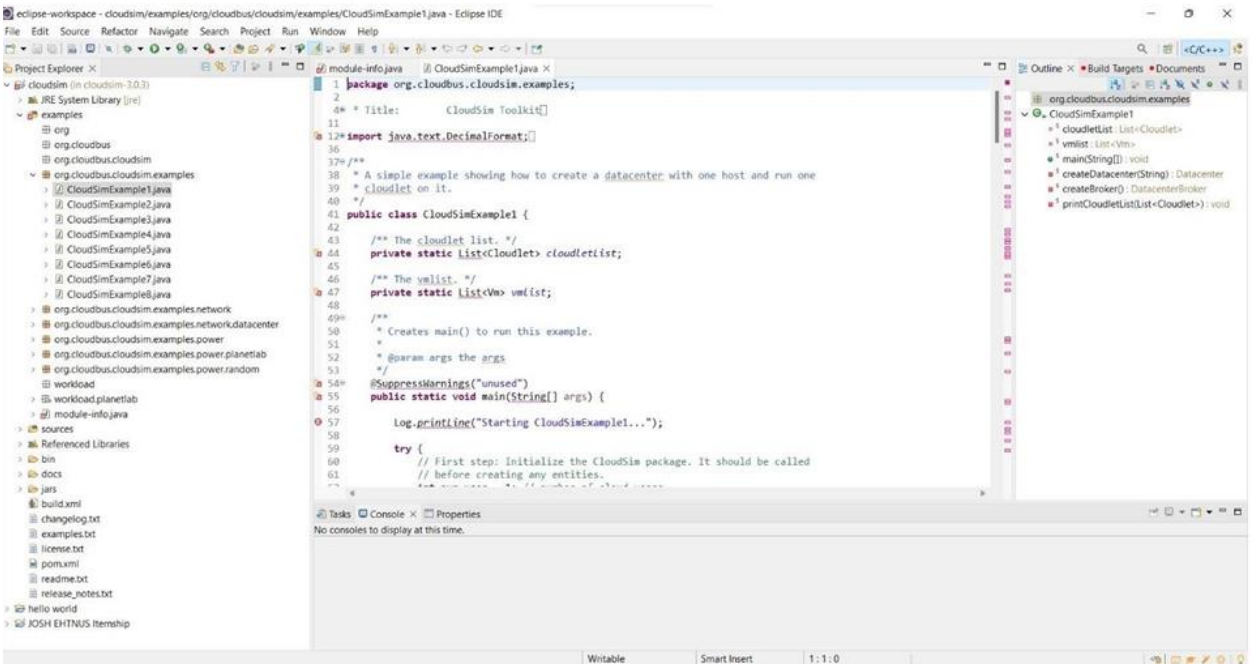


Step 15: Once the project is configured you can open the _*Project Explorer*_and start exploring the Cloudsim project. Also for the first time eclipse automatically start building the workspace for newly configured Cloudsim project, which may take some time depending on the configuration of the computer system.

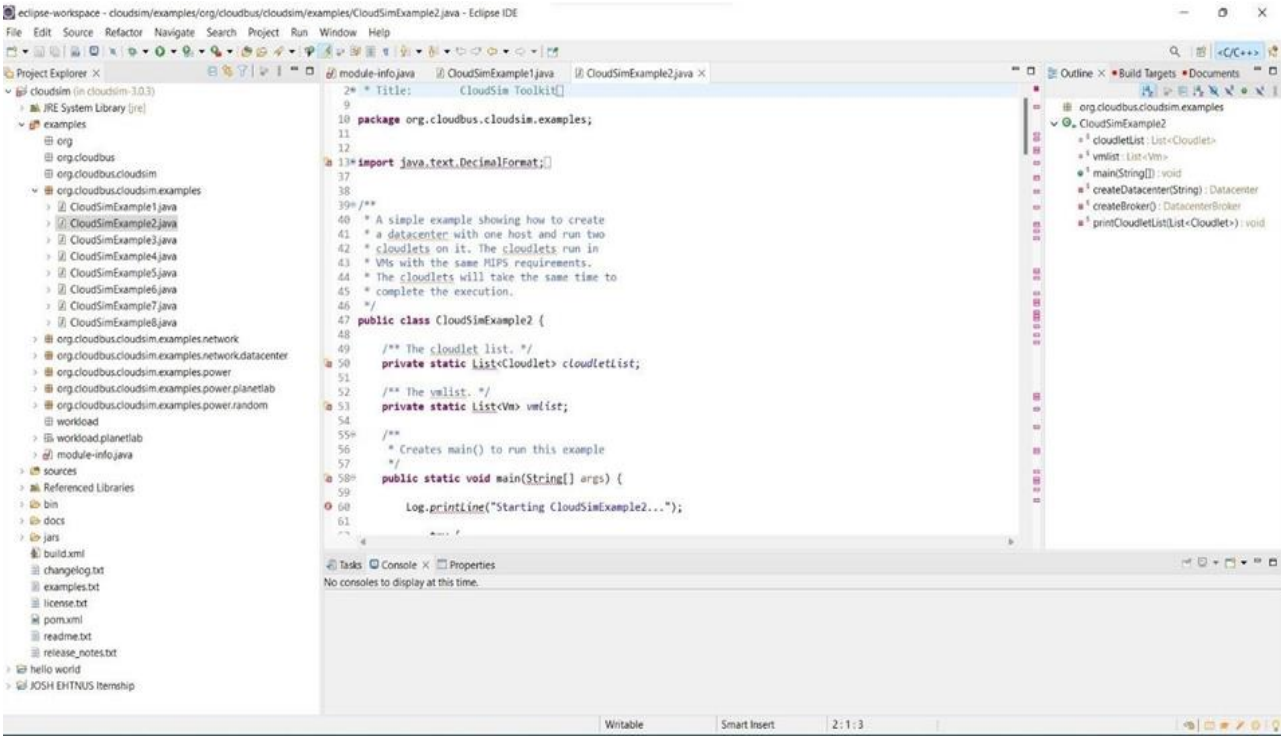
Following is the final screen which you will see after Cloudsim is configured.

Step 16: Now just to check you within the _**Project Explorer**_, you should navigate to the _**examples**_ folder, then expand the package _*org.cloudbus.cloudsim.examples*_ and double click to open the _*CloudsimExample1.java*_



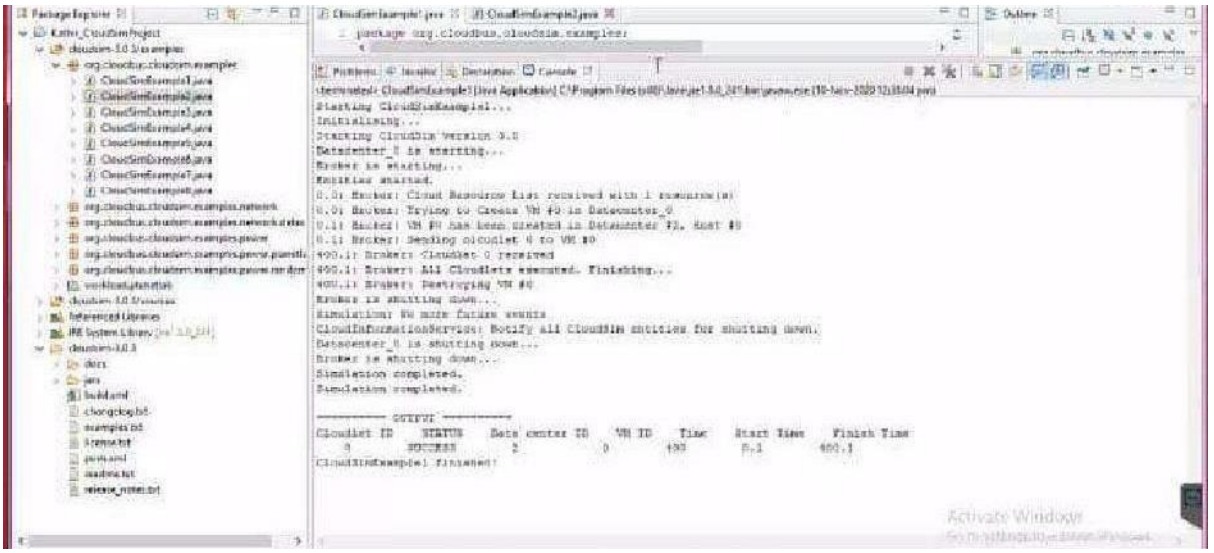

eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSimExample1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
cloudsim (in cloudsim-3.0.3)
JRE System Library [jre]
examples
org
org.cloudbus
org.cloudbus.cloudsim
org.cloudbus.cloudsim.examples
CloudSimExample1.java
CloudSimExample2.java
CloudSimExample3.java
CloudSimExample4.java
CloudSimExample5.java
CloudSimExample6.java
CloudSimExample7.java
CloudSimExample8.java
org.cloudbus.cloudsim.examples.network
org.cloudbus.cloudsim.examples.network.datacenter
org.cloudbus.cloudsim.examples.power
org.cloudbus.cloudsim.examples.power.planetlab
org.cloudbus.cloudsim.examples.power.random
workload
workload.planetlab
module-info.java
sources
Referenced Libraries
bin
docs
jars
build.xml
changelog.txt
examples.txt
license.txt
pom.xml
readme.txt
release_notes.txt
hello world
JOSH EHTNUS Iternship
module-info.java
CloudSimExample1.java
package org.cloudbus.cloudsim.examples;
* Title: CloudSim Toolkit
import java.text.DecimalFormat;
/**
* A simple example showing how to create a datacenter with one host and run one
* cloudlet on it.
*/
public class CloudSimExample1 {
/** The cloudlet list. */
private static List<Cloudlet> cloudletList;
/** The vmlist. */
private static List<Vm> vmlist;
/**
* Creates main() to run this example.
*
* @param args the args
*/
@SuppressWarnings("unused")
public static void main(String[] args) {
Log.printLine("Starting CloudSimExample1...");
try {
// First step: Initialize the CloudSim package. It should be called
// before creating any entities.
Outline
Build Targets
Documents
org.cloudbus.cloudsim.examples
CloudSimExample1
cloudletList : List<Cloudlet>
vmlist : List<Vm>
main(String[]) : void
createDatacenter(String) : Datacenter
createBroker() : DatacenterBroker
printCloudletList(List<Cloudlet>) : void
Tasks
Console
Properties
No consoles to display at this time.
Writable
Smart Insert
1 : 1 : 0

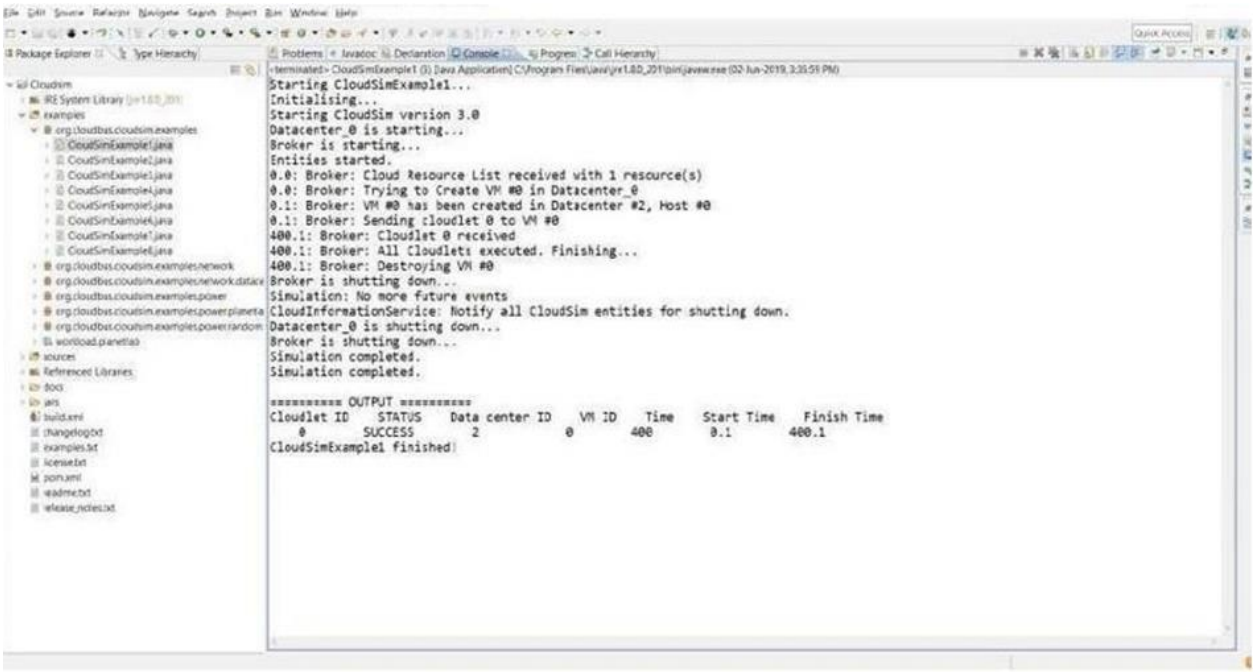
•


eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSimExample2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
cloudsim (in cloudsim-3.0.3)
JRE System Library [jre]
examples
org
org.cloudbus
org.cloudbus.cloudsim
org.cloudbus.cloudsim.examples
CloudSimExample1.java
CloudSimExample2.java
CloudSimExample3.java
CloudSimExample4.java
CloudSimExample5.java
CloudSimExample6.java
CloudSimExample7.java
CloudSimExample8.java
org.cloudbus.cloudsim.examples.network
org.cloudbus.cloudsim.examples.network.datacenter
org.cloudbus.cloudsim.examples.power
org.cloudbus.cloudsim.examples.power.planetlab
org.cloudbus.cloudsim.examples.power.random
workload
workload.planetlab
module-info.java
sources
Referenced Libraries
bin
docs
jars
build.xml
changelog.txt
examples.txt
license.txt
pom.xml
readme.txt
release_notes.txt
hello world
JOSH EHTNUS Iternship
module-info.java
CloudSimExample1.java
CloudSimExample2.java
* Title: CloudSim Toolkit
package org.cloudbus.cloudsim.examples;
import java.text.DecimalFormat;
/**
* A simple example showing how to create
* a datacenter with one host and run two
* cloudlets on it. The cloudlets run in
* VMs with the same MIPS requirements.
* The cloudlets will take the same time to
* complete the execution.
*/
public class CloudSimExample2 {
/** The cloudlet list. */
private static List<Cloudlet> cloudletList;
/** The vmlist. */
private static List<Vm> vmlist;
/**
* Creates main() to run this example
*/
public static void main(String[] args) {
Log.printLine("Starting CloudSimExample2...");
Outline
Build Targets
Documents
org.cloudbus.cloudsim.examples
CloudSimExample2
cloudletList : List<Cloudlet>
vmlist : List<Vm>
main(String[]) : void
createDatacenter(String) : Datacenter
createBroker() : DatacenterBroker
printCloudletList(List<Cloudlet>) : void
Tasks
Console
Properties
No consoles to display at this time.
Writable
Smart Insert
2 : 1 : 3

Step 17: Now navigate to the Eclipse menu _*Run ->Run*_ or directly use a keyboard shortcut *'Ctrl + F11'* to execute the _*CloudsimExample1.java*_.

Step 18: If it is successfully executed it should be displaying the following type to output in the console window of the Eclipse IDE.





gate

Result:

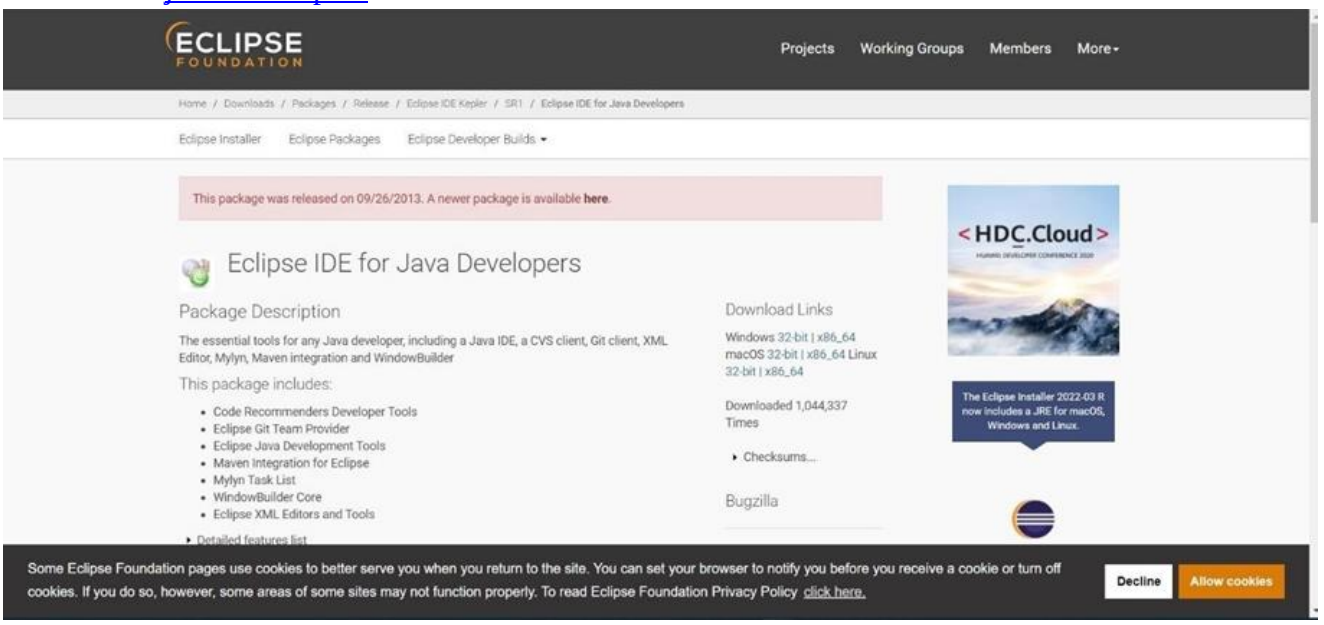
Thus the cloudsim is simulated using Eclipse Environment successfully.

Ex No. 3 b	Simulate a cloud scenario using CloudSim and running a scheduling algorithm

Procedure to import Eclipse, running scheduling algorithms in your system

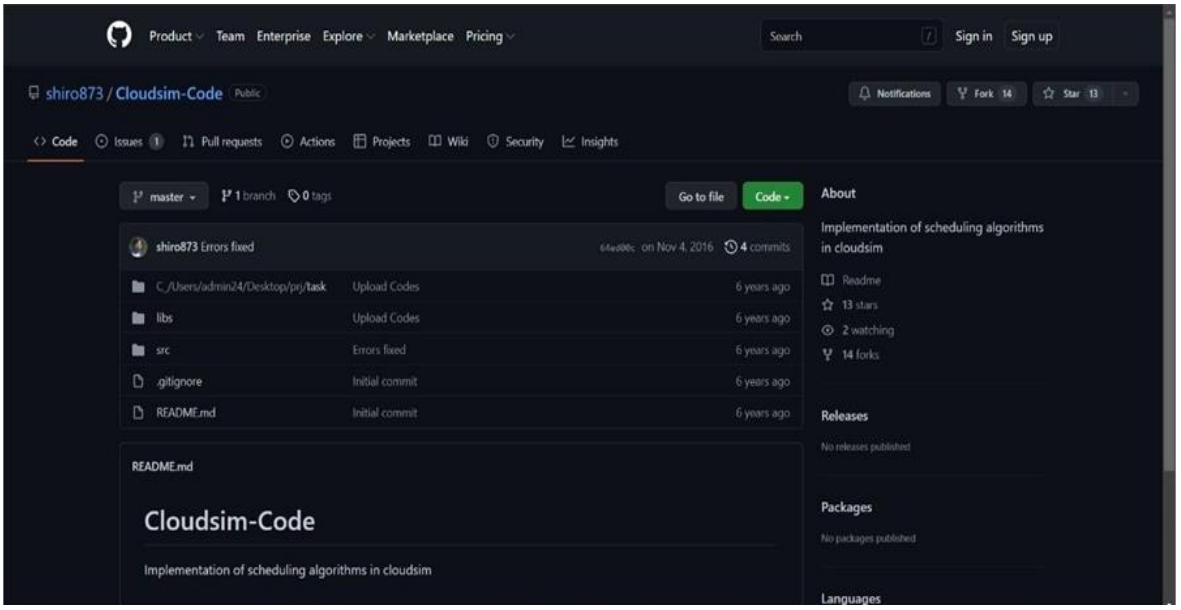
Step 1: Link to download Eclipse and download Eclipse for Windows 64bit into your Local machine

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



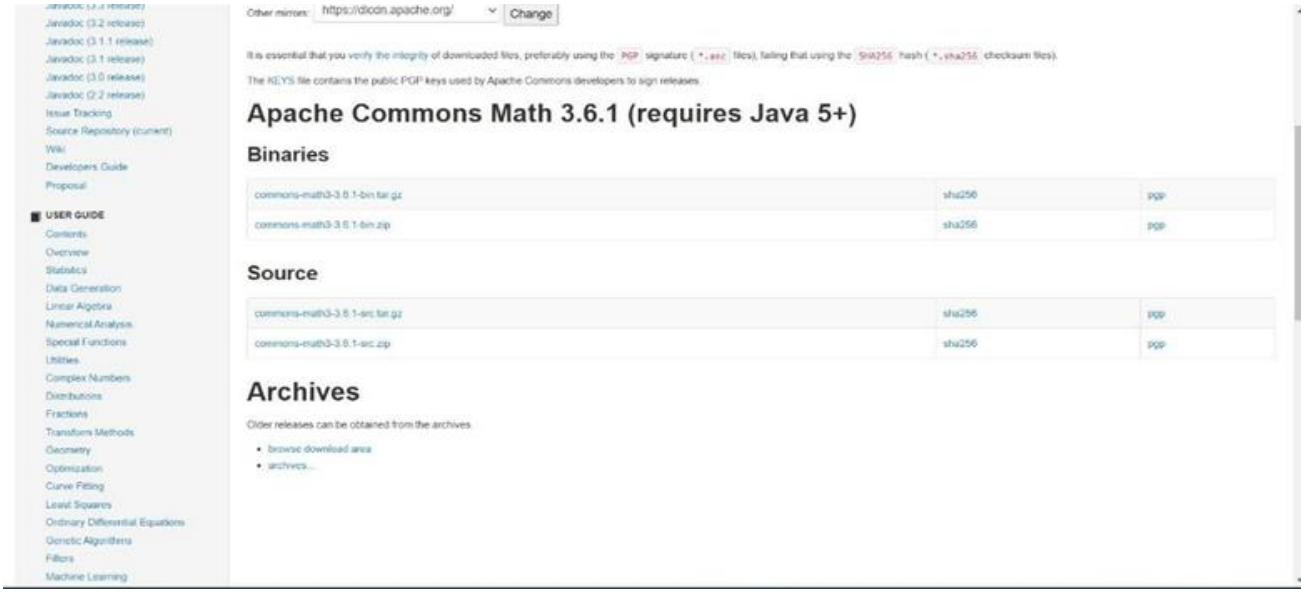
Step 2: Download scheduling source code **cloudsim-code-master** from git hub repository in your local machine

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

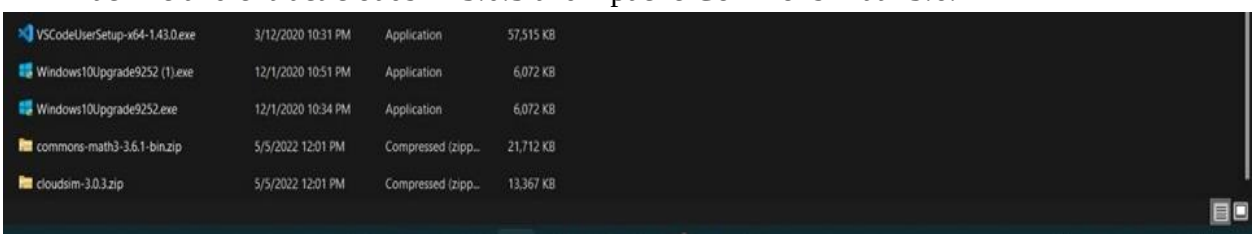


Step 3: Download commons-maths3-3.6.1 from git hub repository in your local machine

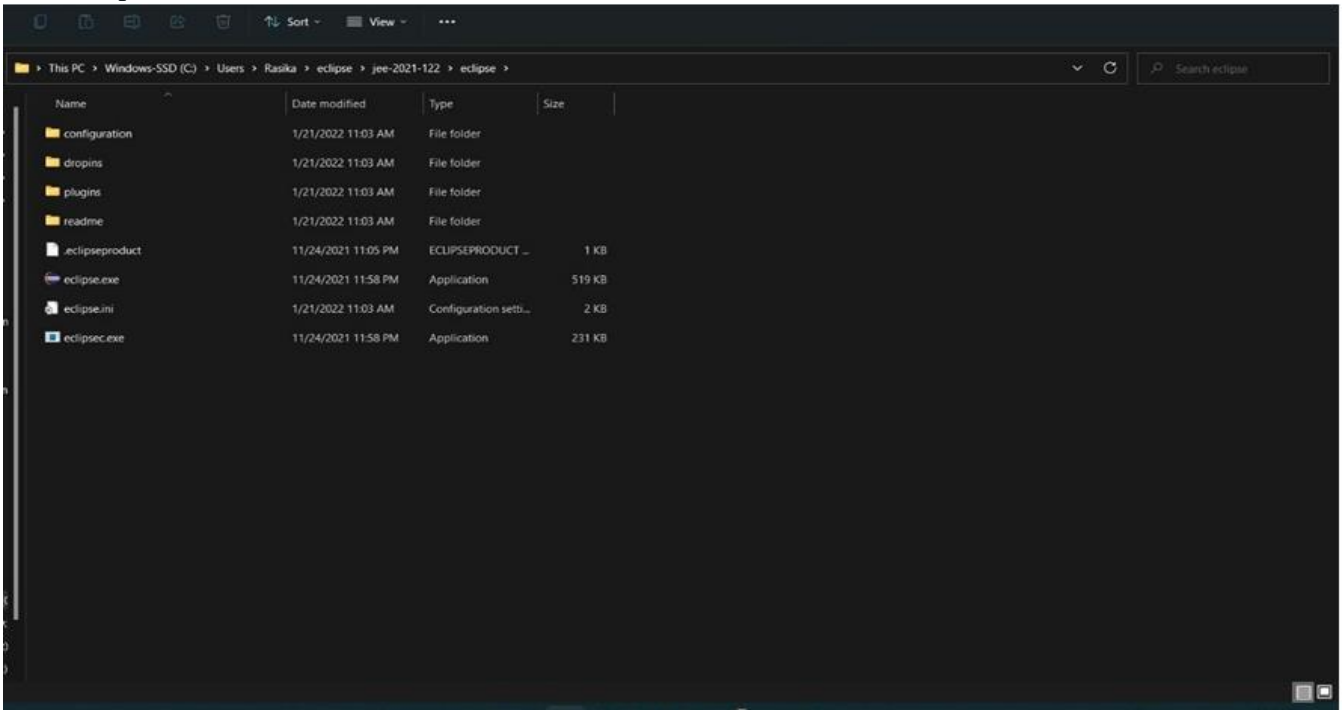
https://commons.apache.org/proper/commons-math/download_math.cgi



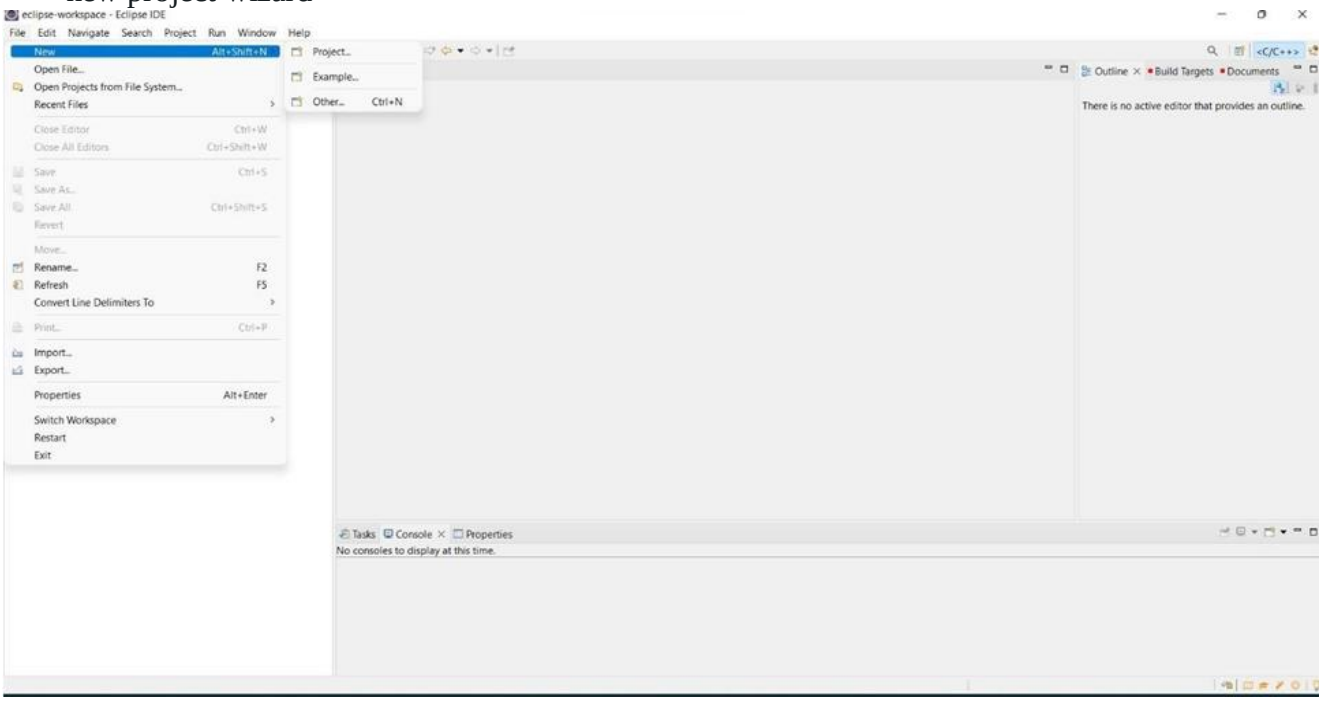
Step 4: Downloaded Eclipse, cloudsim-3.0.3 and Apache Commons Math 3.6.1 in your local machine and extract cloudsim-3.0.3 and Apache Commons Math 3.6.1



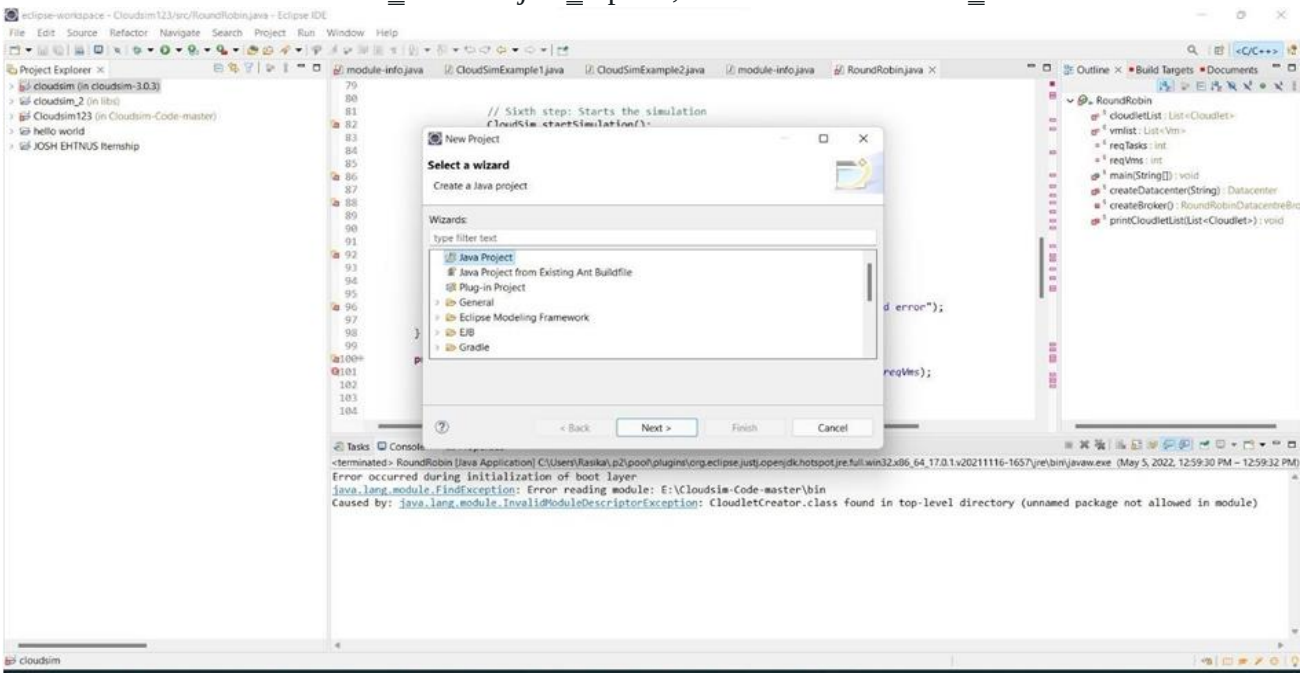
Step 5: First of all, navigate to the folder where you have unzipped the eclipse folder and open Eclipse.exe



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

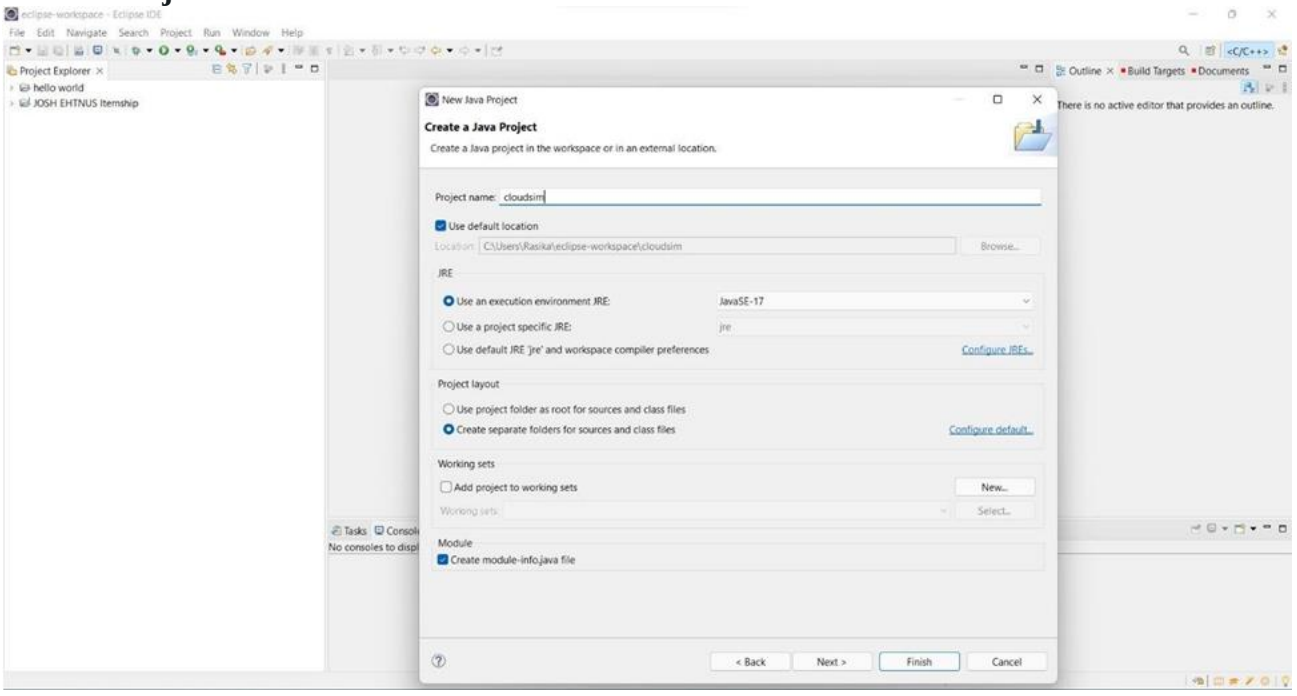


Step 7: A _*New Project*_ wizard should open. There are a number of options displayed and you have to find & select the _*Java Project*_ option, once done click *'Next*_

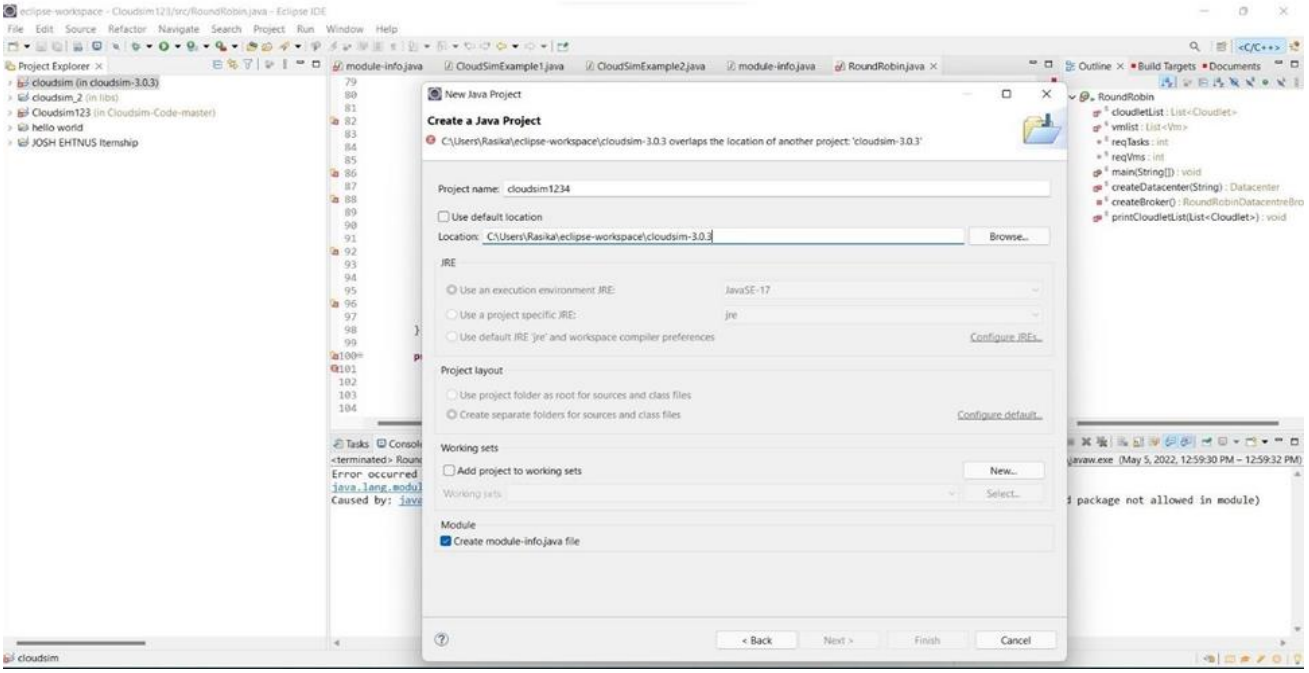


Step 8: Now a detailed new project window will open, here you will provide the project name and the path of CloudSim-master-code project source code, which will be done as follows:

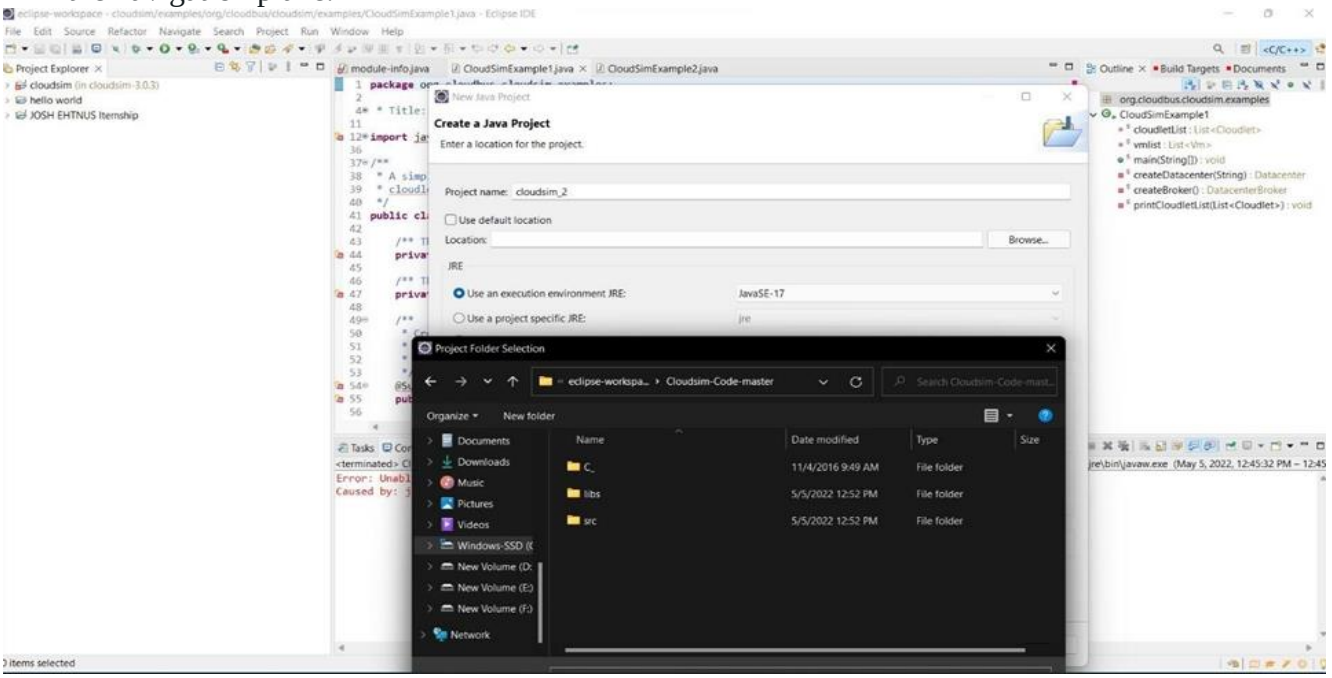
Project Name: CloudSim



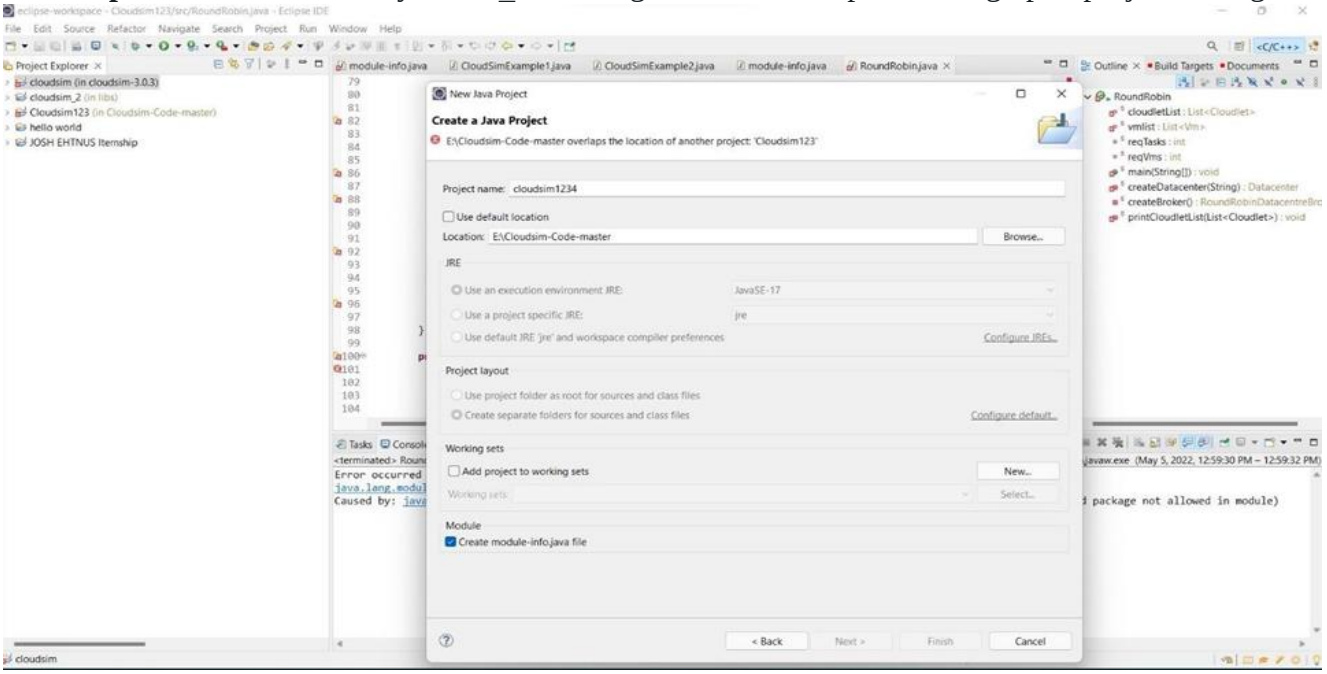
Step 9: Unselect the *'Use default location'* option and then click on *'Browse'* to open the path where you have unzipped the Cloudsim-code-master project and finally click Next to set project settings.



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

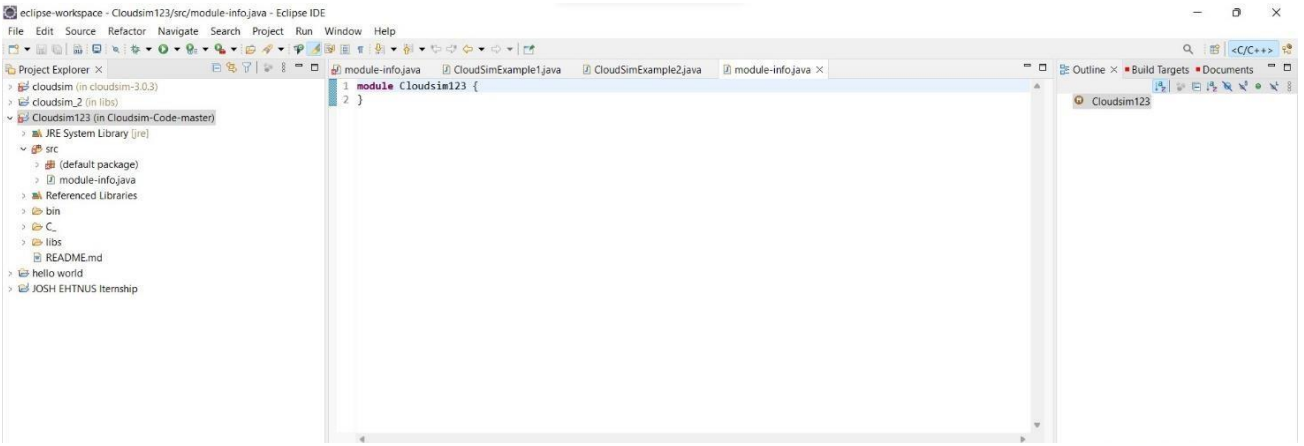


Step 11: Once done finally, click _Next‘ to go to the next step i.e. setting up of project settings

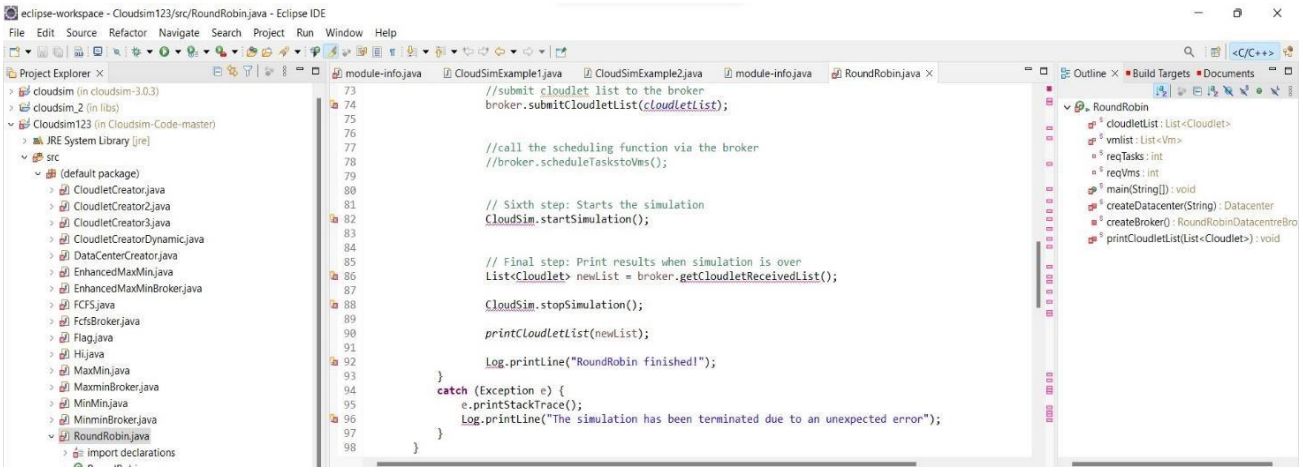


Step 12: Once the project is configured you can open the _*Project Explorer*_ and start exploring the Cloudsim project. Also for the first time eclipse automatically start building the workspace for newly configured Cloudsim project, which may take some time depending on the configuration of the computer system.

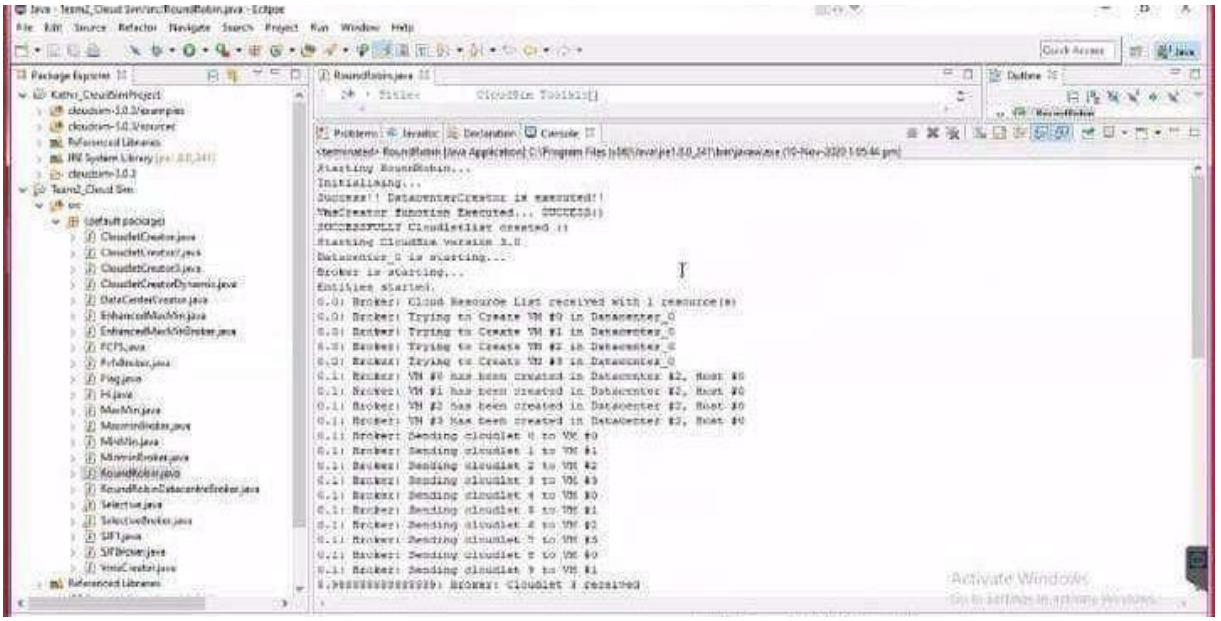
Following is the final screen which you will see after Cloudsim is configured.

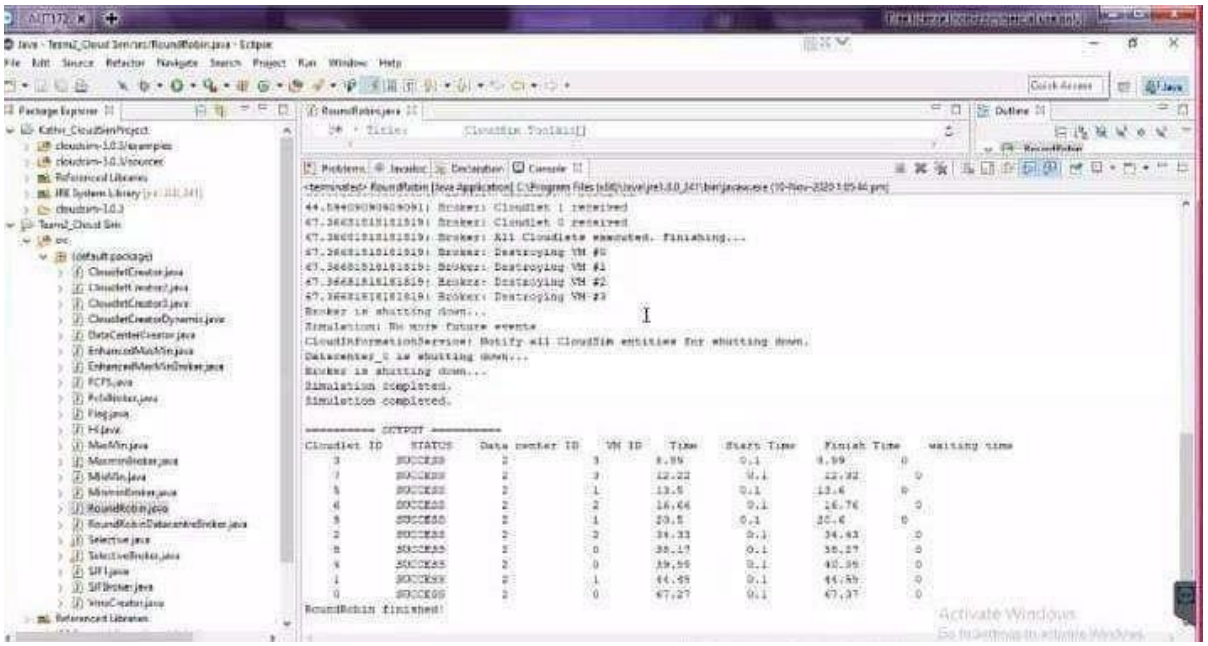


Step 13: Now just to check you within the _**Project Explorer**_, you should navigate to the _**src**_ folder, then expand the package _*default package*_ and double click to open the _R*oundRobin.java*_.



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. 4	Procedure File Transfer in Client & Server using virtual machine.

Aim:

To procedure File Transfer in Client & Server using virtual machine

Steps: Steps to perform File Transfer in Client & Server using virtual machine.

Step 1: Open a virtual machine to do file transfer.
Step 2: Write the java program for FTP Client and FTP Server. Step 3: Run the program.

Source Code:

FTPClient.java

```
import java.io.*; import java.net.*; import java.util.*; public class
FTPClient{ public static void main(String args[])throws
IOException        { try { int number;
                        Socket s=new Socket("127.0.0.1",10087);
                        Scanner sc=new Scanner(System.in);
                        System.out.println("Enter the file name:");
                        String fn=sc.next();
                        DataOutputStream dos=new DataOutputStream(s.getOutputStream());
                        dos.writeUTF(fn);
                        DataInputStream dis=new DataInputStream(s.getInputStream());
                        String input=(String)dis.readUTF();
                        FileInputStream fis=new FileInputStream(input);
                        System.out.println("Even Numbers in the" +fn+" are");
                        int i=0;
                        while((i=fis.read())!=-1){
                                System.out.println((char)i);
                        }
                        s.close(); }
                catch(Exception e){
                        System.out.println("Port not available "+e); }
        }
}
```

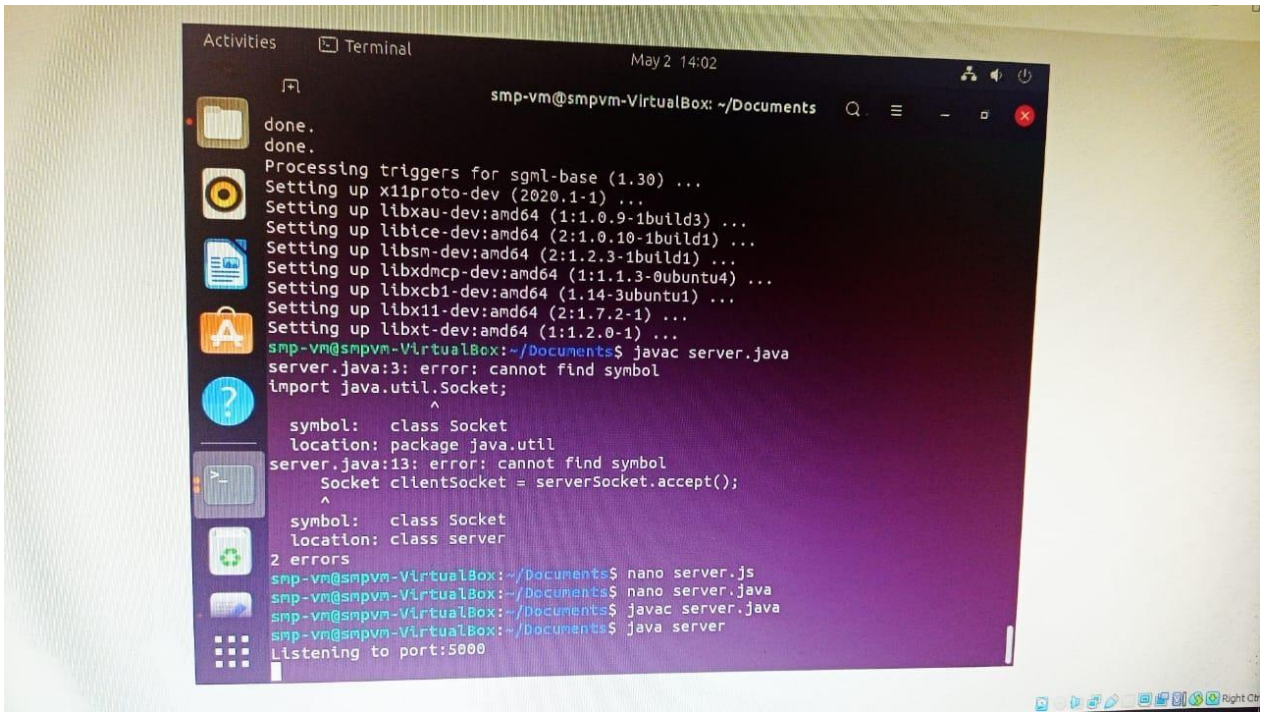
FTPServer.java

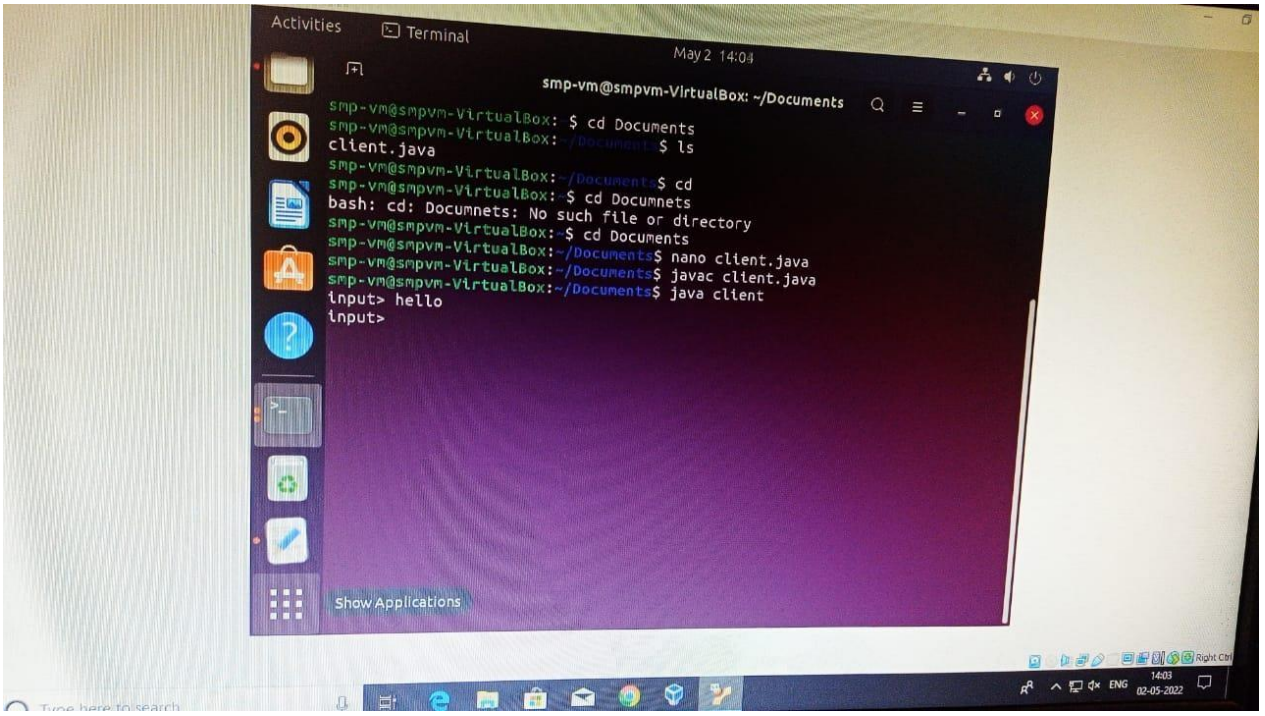
```
import java.io.*; import java.net.*; import java.util.*; public class
FTPServer{ public static void main(String args[])throws
IOException{ try{ int num;
                        Scanner sc=new Scanner(System.in);
                        ServerSocket ss=new ServerSocket(10087);
                        Socket s=ss.accept();
                        System.out.println("Waiting. ....");
                        DataInputStream dis=new DataInputStream(s.getInputStream());
                        String input=(String)dis.readUTF();
                        DataOutputStream dos=new DataOutputStream(s.getOutputStream());
                        FileInputStream fis = new FileInputStream("out.txt");
                        FileOutputStream fos = new
                        FileOutputStream(input); while((num=fis.read())!= -
                        1) { if(num%2==0) { fos.write(num);
                                }
                        }
                        dos.writeUTF(input);
                        System.out.println("File is sent to
                client"); ss.close(); s.close(); }
                catch(Exception e) {
                        System.out.println("Port not available"+e); }
        }
}
```

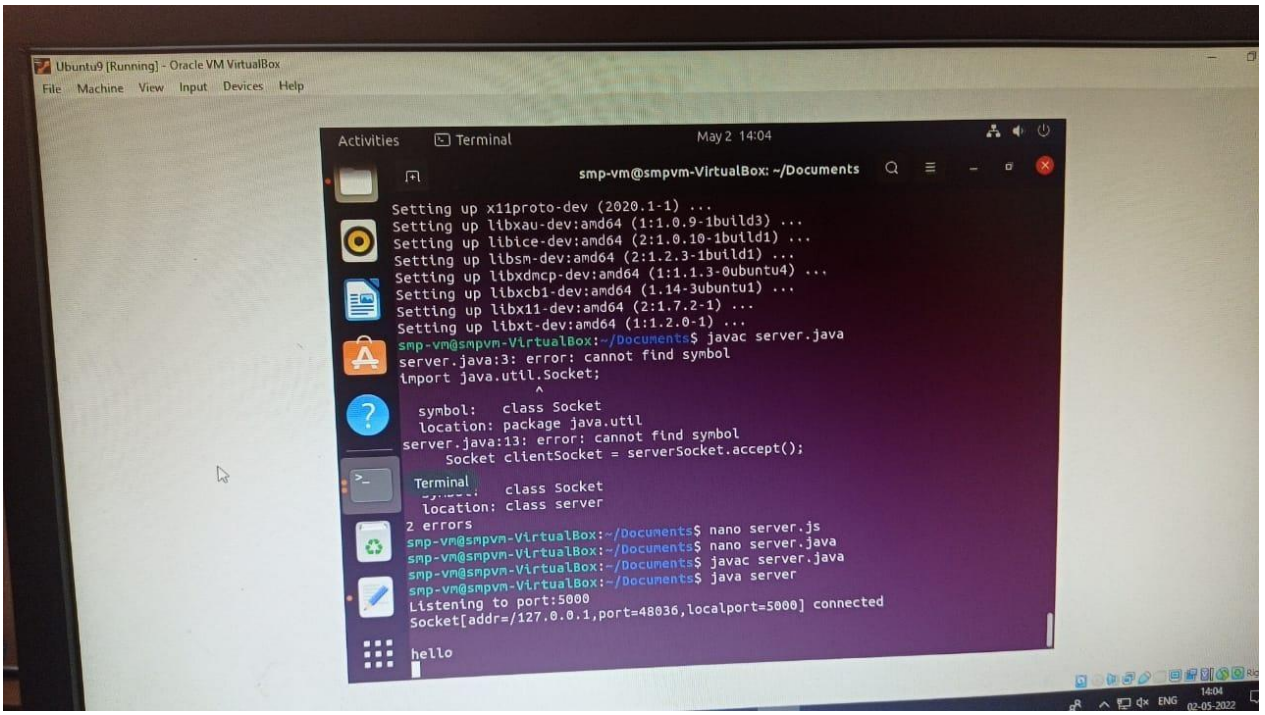
Out.txt

```
1 2
3
4 5
6
7 8
9
```

Output:


Activities
Terminal
May 2 14:02
smp-vm@smpvm-VirtualBox: ~/Documents
done.
done.
Processing triggers for sgml-base (1.30) ...
Setting up x11proto-dev (2020.1-1) ...
Setting up libxau-dev:amd64 (1:1.0.9-1build3) ...
Setting up libice-dev:amd64 (2:1.0.10-1build1) ...
Setting up libsm-dev:amd64 (2:1.2.3-1build1) ...
Setting up libxdmcp-dev:amd64 (1:1.1.3-0ubuntu4) ...
Setting up libxcb1-dev:amd64 (1.14-3ubuntu1) ...
Setting up libx11-dev:amd64 (2:1.7.2-1) ...
Setting up libxt-dev:amd64 (1:1.2.0-1) ...
smp-vm@smpvm-VirtualBox:~/Documents$ javac server.java
server.java:3: error: cannot find symbol
import java.util.Socket;
^
symbol: class Socket
location: package java.util
server.java:13: error: cannot find symbol
Socket clientSocket = serverSocket.accept();
^
symbol: class Socket
location: class server
2 errors
smp-vm@smpvm-VirtualBox:~/Documents$ nano server.js
smp-vm@smpvm-VirtualBox:~/Documents$ nano server.java
smp-vm@smpvm-VirtualBox:~/Documents$ javac server.java
smp-vm@smpvm-VirtualBox:~/Documents$ java server
Listening to port:5000


Activities
Terminal
May 2 14:04
smp-vm@smpvm-VirtualBox: ~/Documents
smp-vm@smpvm-VirtualBox: $ cd Documents
smp-vm@smpvm-VirtualBox: $ ls
client.java
smp-vm@smpvm-VirtualBox:~/Documents$ cd
smp-vm@smpvm-VirtualBox: $ cd Documnets
bash: cd: Documnets: No such file or directory
smp-vm@smpvm-VirtualBox:~$ cd Documents
smp-vm@smpvm-VirtualBox:~/Documents$ nano client.java
smp-vm@smpvm-VirtualBox:~/Documents$ javac client.java
smp-vm@smpvm-VirtualBox:~/Documents$ java client
input> hello
input>
Show Applications
ENG
14:03
02-05-2022



Result:

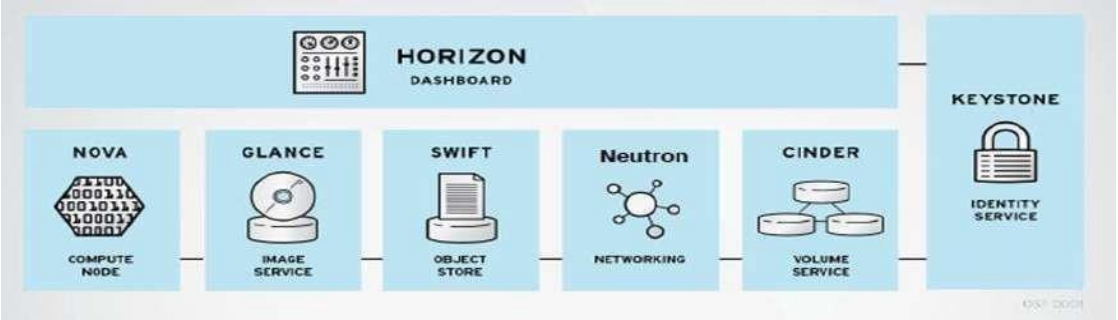
Thus the program to the File transfer operation using virtual machine was successfully executed and verified.

Ex No. 5	Find a procedure to launch virtual machine using Openstack

Introduction:

- OpenStack was introduced by Rackspace and NASA in July 2010.
- OpenStack is an Infrastructure as a Service known as Cloud Operating System, that take resources such as Compute, Storage, Network and Virtualization Technologies and control those resources at a data center level
- The project is building an open source community - to share resources and technologies with the goal of creating a massively scalable and secure cloud infrastructure.
- The software is open source and limited to just open source APIs such as Amazon.

The following figure shows the OpenStack architecture



OpenStack architecture

- It is modular architecture
- Designed to easily scale out
- Based on (growing) set of core services **The major components are**

1. **Keystone**
2. **Nova**
3. **Glance**
4. **Swift**
5. **Quantum**
6. **Cinder**

- **KEYSTONE :**
  - Identity service ○ Common authorization framework
    ○ Manage users, tenants and roles
  - Pluggable backends (SQL,PAM,LDAP, IDM etc)

- **NOVA**
  - Core compute service comprised of
    - ✦ Compute Nodes – hypervisors that run virtual machines
      - Supports multiple hypervisors KVM,Xen,LXC,Hyper-V and ESX
    - ✦ Distributed controllers that handle scheduling, API calls, etc
  - Native OpenStack API and Amazon EC2 compatible API
  - **GLANCE**
  - Image service ○ Stores and retrieves disk images (Virtual machine templates) ○ Supports RAW,QCOW,VHD,ISO,OVF & AMI/AKI
  - Backend Storage : File System, Swift, Gluster, Amazon S3
- **SWIFT**
  - Object Storage service ○ Modeled after Amazon‘s Service
  - Provides simple service for storing and retrieving arbitrary data ○ Native API and S3 compatible API
- **NEUTRON**
  - Network service
  - Provides framework for Software Defined Network ○ Plugin architecture
    - ✦ Allows intergration of hardware and software based network solutions
  - Open vSwitch, Cisco UCS,Standard Linux Bridge,NiCira NVP

- **CINDER**

  ○ Block Storage (Volume) service ○ Provides block storage for Virtual machines(persistent disks) ○ Similar to Amazon EBS service ○ Plugin architecture for vendor extensions

  ✦ NetApp driver for cinder

- **HORIZON**
  - ○ Dashboard
  - ○ Provides simple self service UI for end-users ○ Basic cloud administrator functions
    - ✦ Define users, tenants and quotas
    - ✦ No infrastructure management
- **HEAT OpenStack Orchestration**
  - ○ Provides template driven cloud application orchestration ○ Modeled after AWS Cloud Formation ○ Targeted to provide advanced functionality such as high availability and auto scaling
  - ○ Introduced by Redhat

• **CEILOMETER –** OpenStack Monitoring and Metering ○ Goal: To Provide a single infrastructure to collect measurements from an entire OpenStack Infrastructure; Eliminate need for multiple agents attaching to multiple OpenStack Projects ○ Primary targets metering and monitoring: Provided extensibility

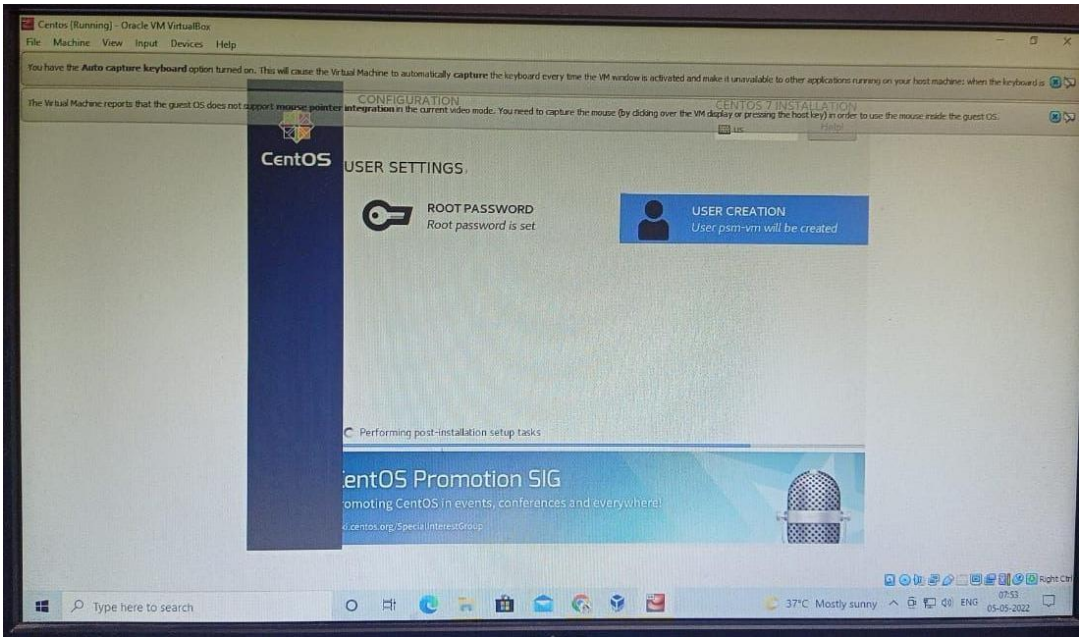
✞ **Steps in Installing Openstack Step 1:**

• Download and Install Oracle Virtual Box latest version & Extension package

○ https://virtualbox.org/wiki/downloads

Step 2:

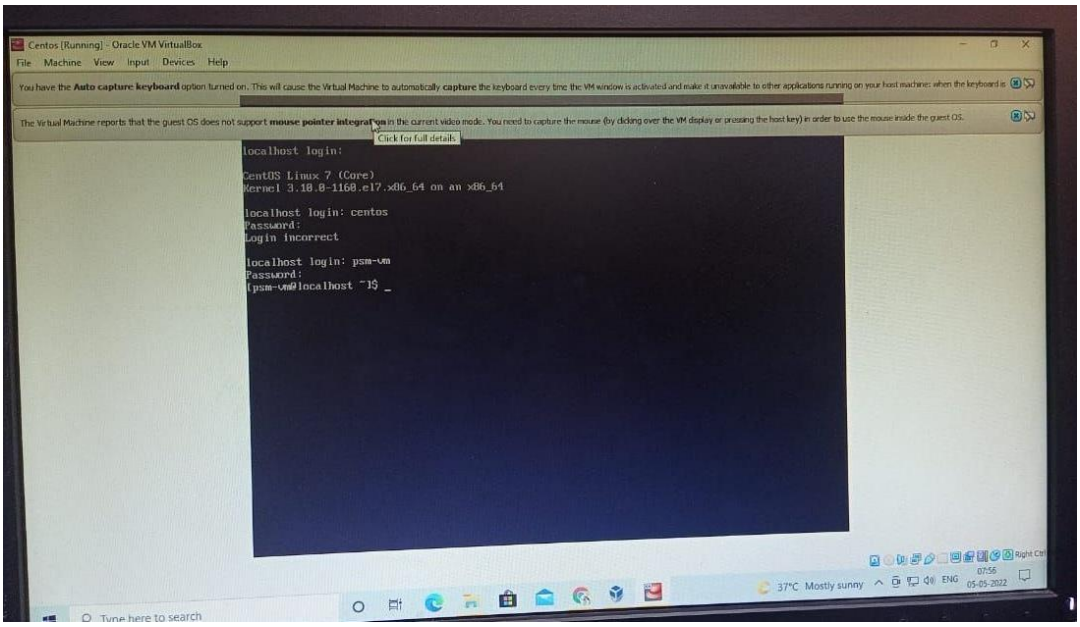
- Download CentOS 7 OVA(Open Virtual Appliance) from ○ Link : https://linuxvmimages.com/images/centos-7
- Import CentOS 7 OVA(Open Virtual Appliance) into Oracle Virtual Box



Step 3: Login into CenOS 7

- Login Details ○ **User name : centos** ○ **Password : centos**
- To change into root user in Terminal

#sudosu–

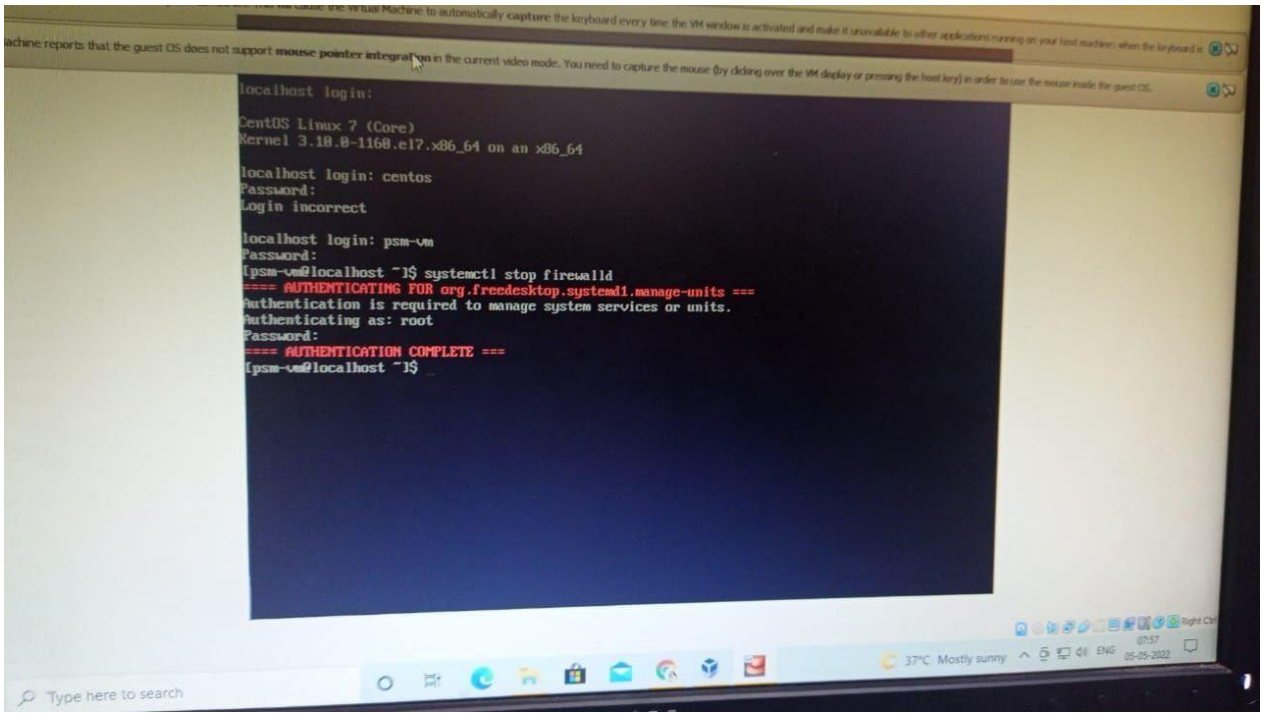


Step 4: Installation Steps for OpenStack

Step5: Command to disable and stop firewall

 # systemctl disable firewalld

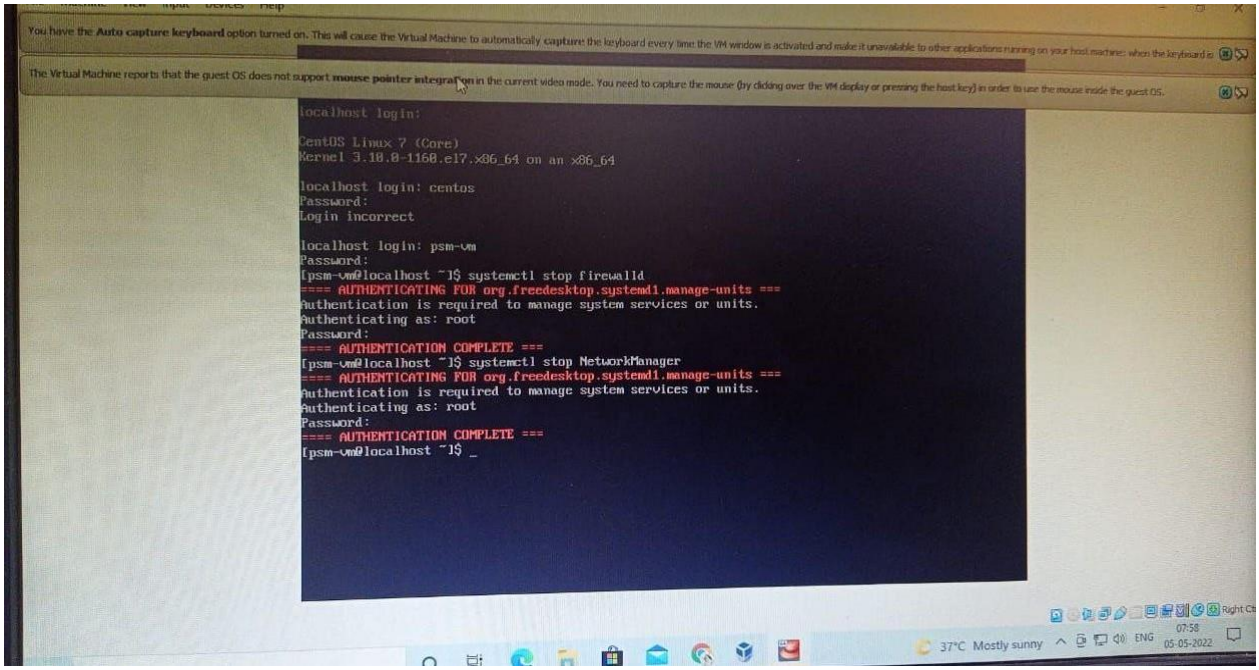
#systemctl stop firewalld



Step 6: Command to disable and stop Network Manager

 # systemctl disable NetworkManager

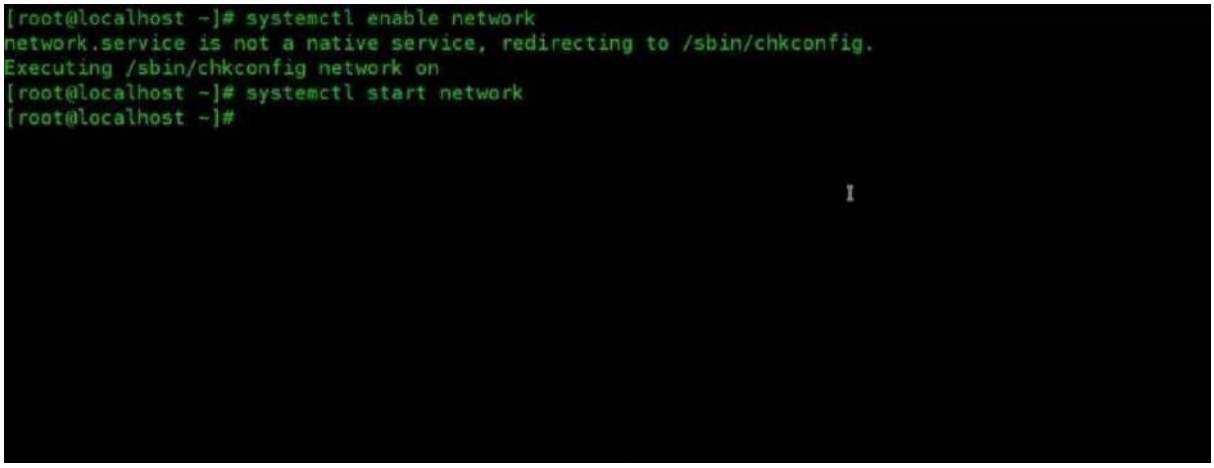
 # systemctl stop NetworkManager



Step 7: Enable and start Network

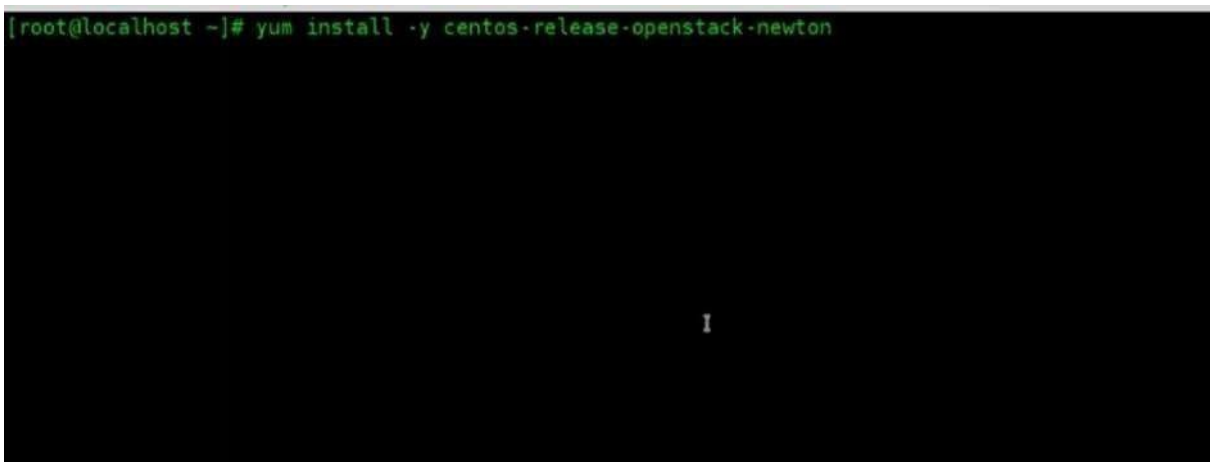
#systemctl enable network

#systemctl start network



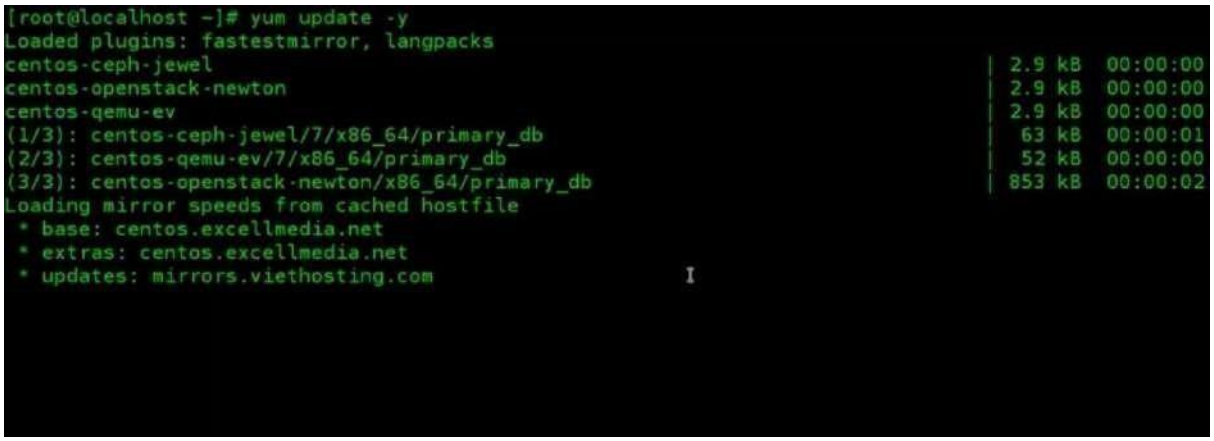
Step 8: OpenStack will be deployed on your Node with the help of **PackStack** package provided by **rdo** repository (**RPM Distribution of OpenStack**).In order to enable **rdo** repositories on Centos **7** run the below command.

#yum install -y https://rdoproject.org/repos/rdo-release.rpm



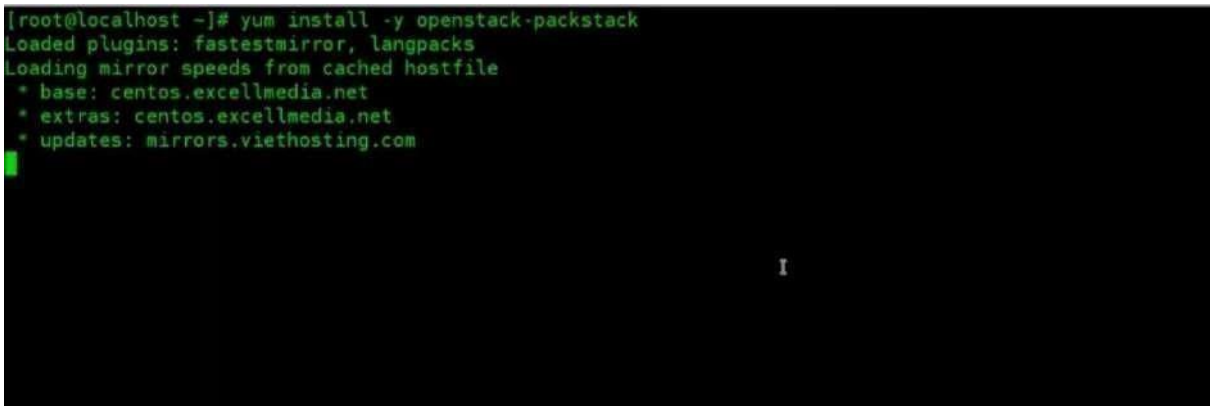
Step 9: Update Current packages

#yum update –y



Step 10: Install OpenStack Release for CentOS

#yum install –y openstack-packstack



Step 11: Start packstack to install OpenStack Newton

#packstak --allinone

```
[root@localhost ~]# packstack --allinone
Welcome to the Packstack setup utility

The installation log file is available at: /var/tmp/packstack/20170314-065810-b8cxch/openstack-setup.lo
g
Packstack changed given value  to required value /root/.ssh/id_rsa.pub

Installing:
Clean Up                                             [ DONE ]
Discovering ip protocol version                      [ DONE ]
Setting up ssh keys                                  [ DONE ]
Preparing servers                                    [ DONE ]
Pre installing Puppet and discovering hosts' details [ DONE ]
Preparing pre-install entries                        [ DONE ]
Setting up CACERT                                    [ DONE ]
Preparing AMQP entries                               [ DONE ]
Preparing MariaDB entries                            [ DONE ]
Fixing Keystone LDAP config parameters to be undef if empty[ DONE ]
Preparing Keystone entries                           [ DONE ]
Preparing Glance entries                             [ DONE ]
Checking if the Cinder server has a cinder-volumes vg[ DONE ]
Preparing Cinder entries                             [ DONE ]
Preparing Nova API entries                           [ DONE ]
```

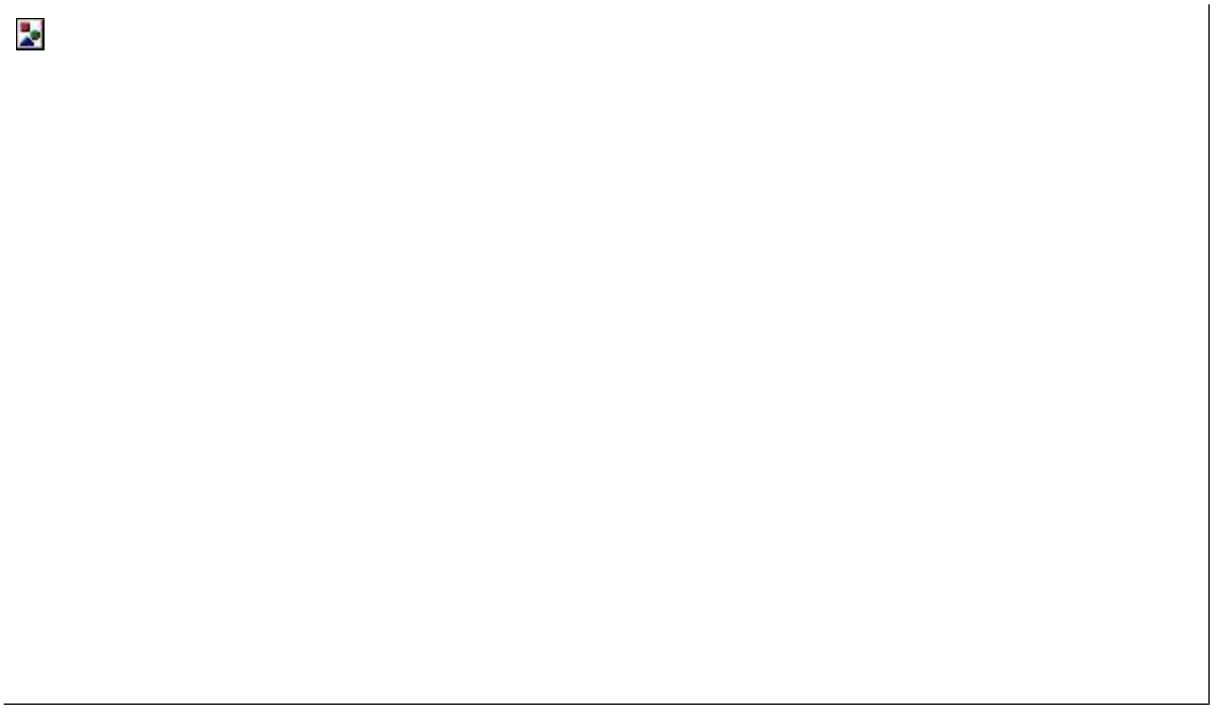
Step 12: Note the user name and password from keystonerc_admin

#cat keystonerc_admin

```
[root@localhost ~]# ls
anaconda-ks.cfg       keystonerc_admin  packstack-answers-20170314-065812.txt
initial-setup-ks.cfg  keystonerc_demo
[root@localhost ~]# cat keystonerc_admin
unset OS_SERVICE_TOKEN
    export OS_USERNAME=admin
    export OS_PASSWORD=cdc897f8cb7f4dda
    export OS_AUTH_URL=http://10.0.2.15:5000/v2.0
    export PS1='[\u@\h \W(keystone_admin)]\$ '

export OS_TENANT_NAME=admin
export OS_REGION_NAME=RegionOne
    [root@localhost ~]#
```

Step 13: Click the URL and enter the user name and password to start OpenStack

OpenStack is successfully launched in your machine



Result:

Thus the OpenStack Installation is executed successfully.

Title: Design an Assignment to retrieve, verify, and store user credentials using Firebase Authentication, the Google App Engine standard environment, and Google Cloud Data store

Before you begin

Step-1: Install Git,

```
udo apt          gi
```

Step-II: Install Python 2.7

Step-III: Sign in to your Google Account

If you don't already have one, sign up for a new account.

Step-IV: Select or create a GCP project

Note: If you don't plan to keep the resources you create in this tutorial, create a new project instead of selecting an existing project. After you finish, you can delete the project, removing all resources associated with the project and tutorial.

GO TO THE MANAGE RESOURCES PAGE

Step-V: Install and initialize the Cloud SDK.

If you have already installed and initialized the SDK to a different project, set the gcloud project to the App Engine project ID you're using for Firenotes. See Managing Cloud SDK Configurations for specific commands to update a project with the gcloud tool.

Step-VI: Cloning the sample app

To download the sample to your local machine:

Clone the sample application repository to your local machine:

git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git

Alternatively, you can download the sample as a zip file and extract it.

Navigate to the directory that contains the sample code:

cd python-docs-samples/appengine/standard/firebase/firenotes

Step-VII: Adding the Firebase Authentication user interface

To configure FirebaseUI and enable identity providers:

1. Add Firebase to your app by following these steps:

a. Create a Firebase project in the Firebase console.

   If you don't have an existing Firebase project, click **Add project** and enter either an existing Google Cloud Platform project name or a new project name.

   If you have an existing Firebase project that you'd like to use, select that project from the console.

b. From the project overview page, click **Add Firebase to your web app**. If your project already has an app, select **Add App** from the project overview page.

c. Use the Initialize Firebase section of your project's customized code snippet to fill out the following section of the frontend/main.js file:

GO to given Path appengine/standard/firebase/firenotes/frontend/main.js

appengine/standard/firebase/firenotes/frontend/main.js

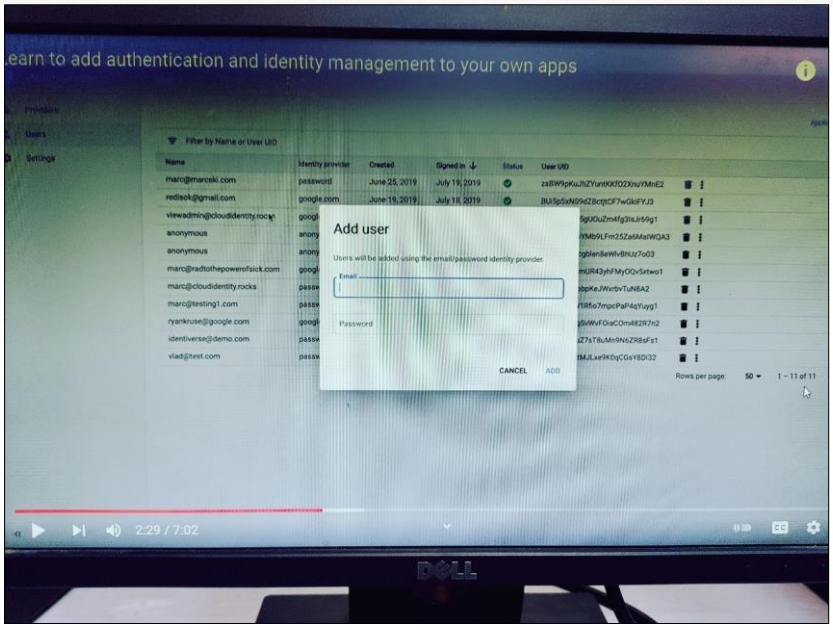
VIEW ON GITHUB

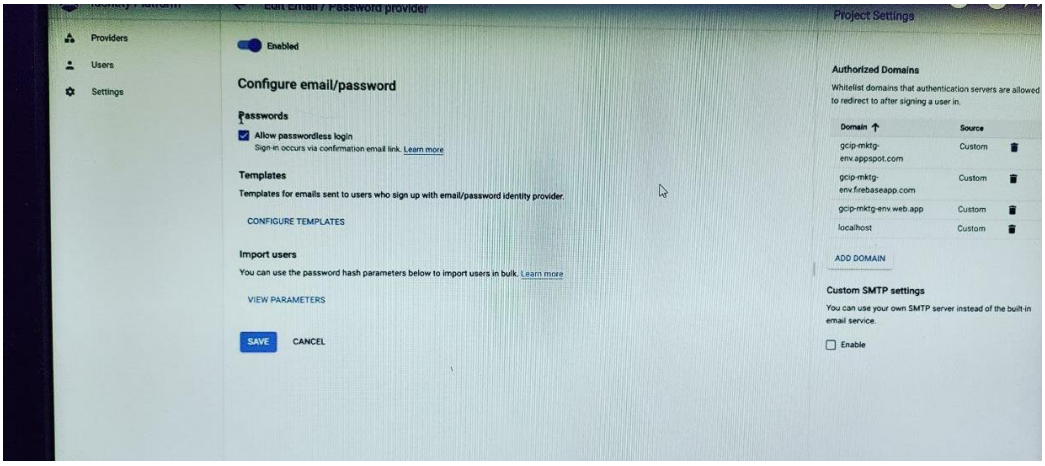
```
// Obtain the following from the "Add Firebase to your web app" dialogue
    initialize Firebase
        fig = {
  apiKey:    <API_KEY>
  authDomain:          ID>.firebaseapp.com",
  databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
  project_id:              ID>
  storageBucket: "<BUCKET>.appspot.com",
  messagingSenderId: "<MESSAGING_SENDER_ID>"
};
```

2. Edit the **backend/app.yaml** file and enter your Firebase project ID in the environment variables:

VIEW ON GITHUB

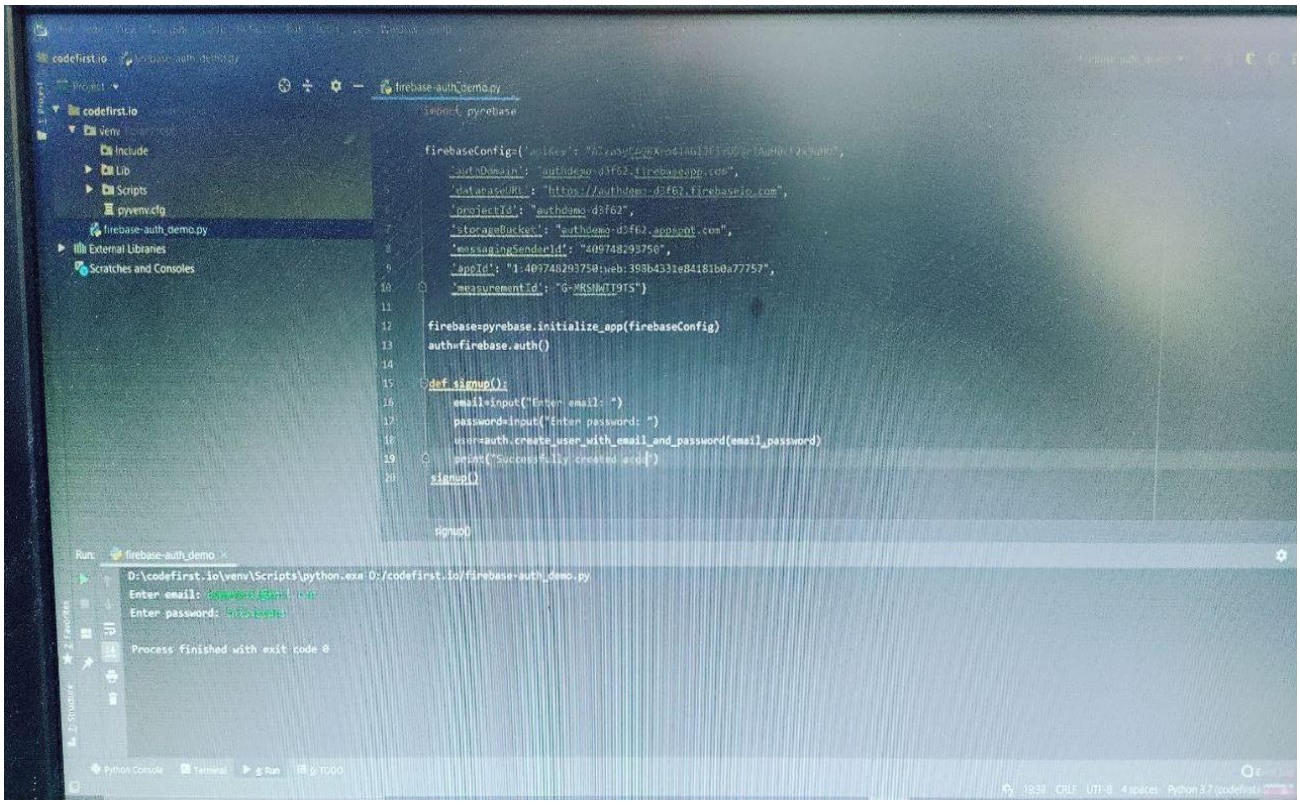
```
runtime
api
threadsafe
  url  backend

handlers
- url  /.*
```



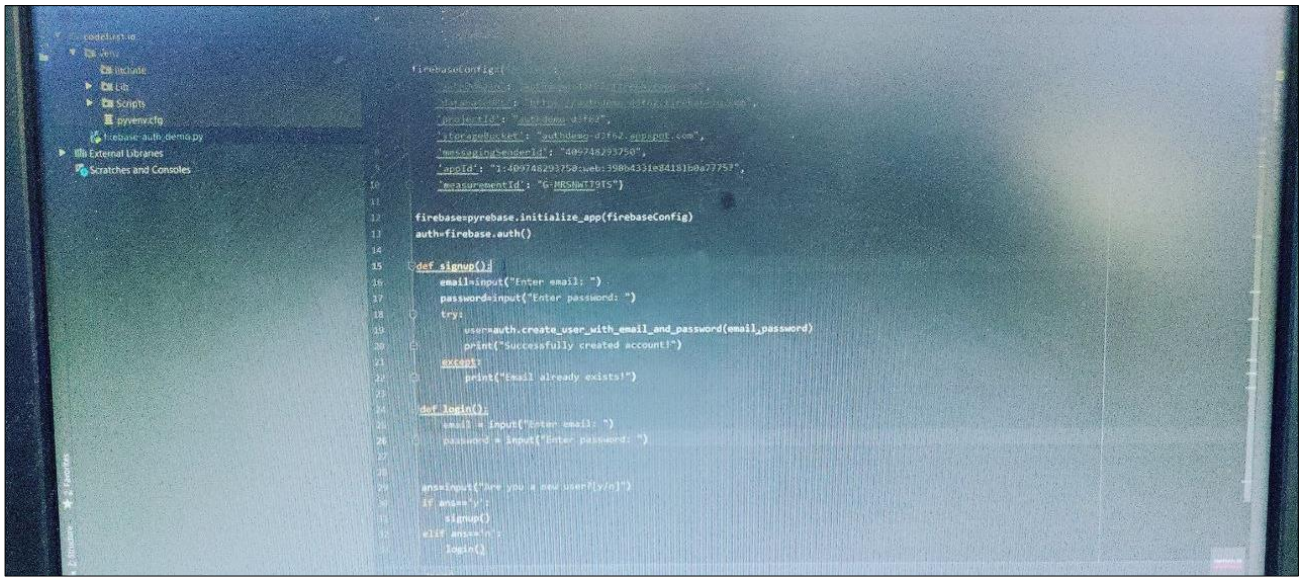


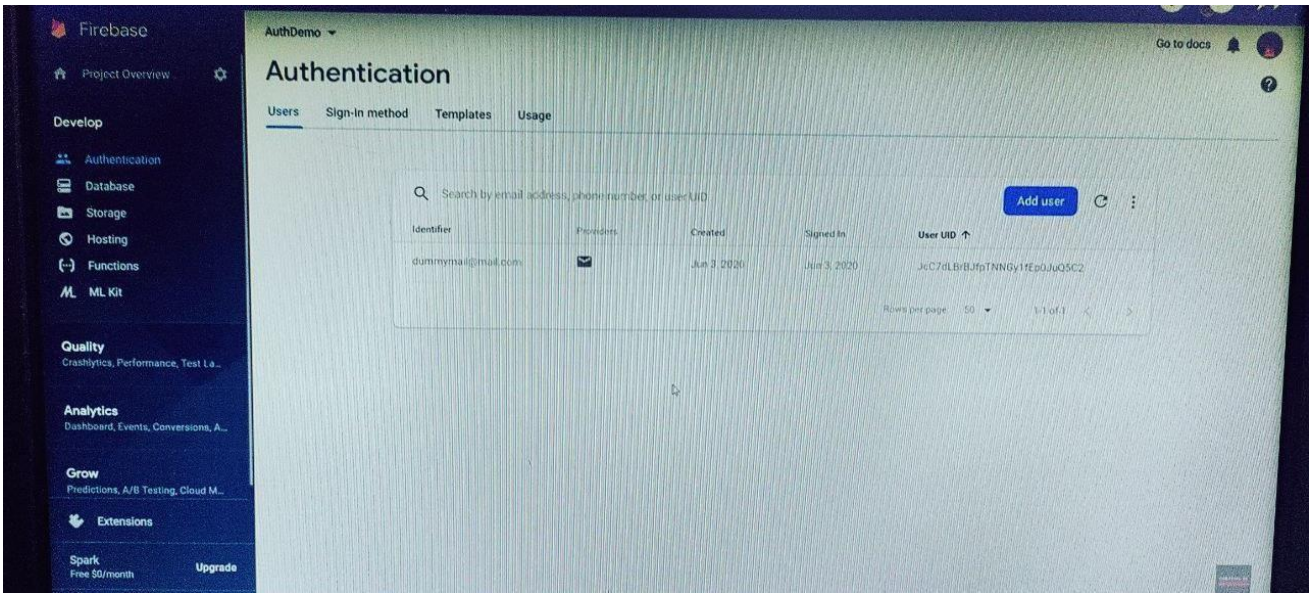
```
ript  main.app

 variables:
  Repl  with your Fireba  project ID.
FIREBASE                  D>
```

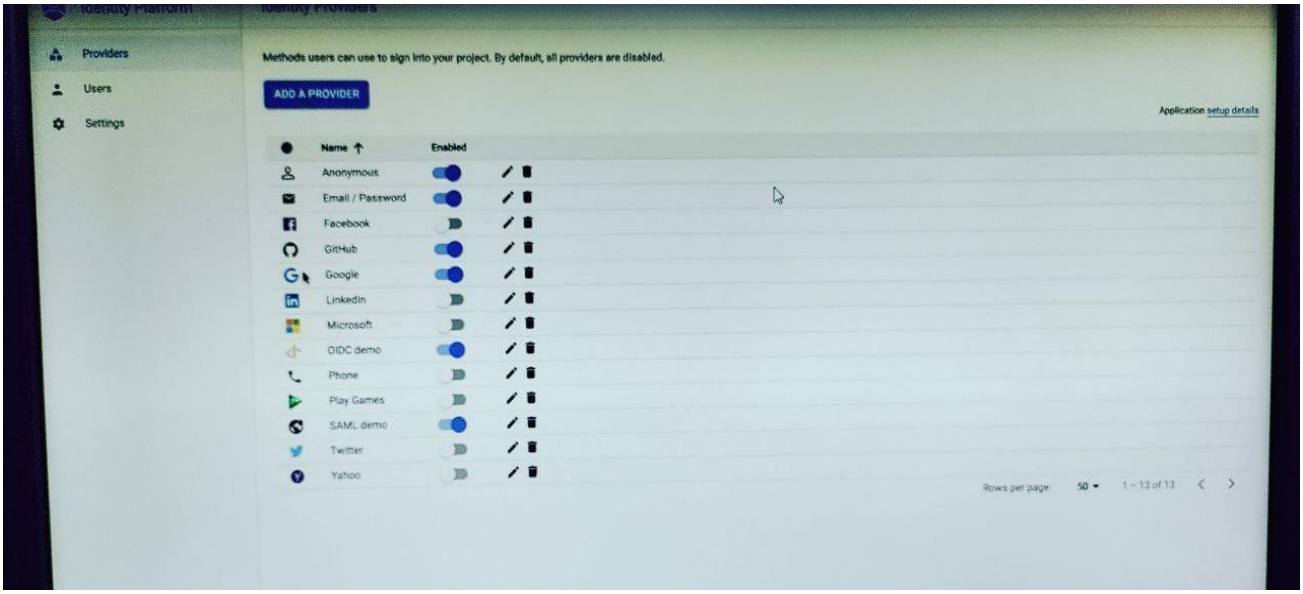
3. In the **frontend/main.js** file, configure the providers you want to offer your users. FirebaseUI login widget by selecting which



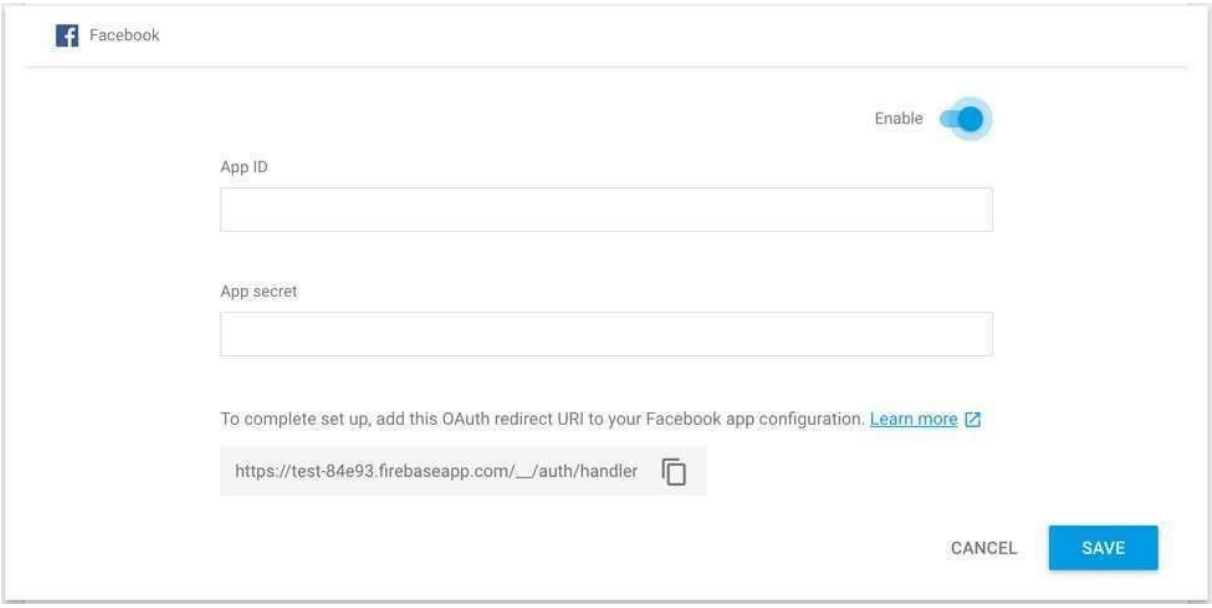
4. **Enable the providers you have chosen to keep in the Firebase console by clicking Authentication > Sign-in method. Then, under** Sign-in providers**, hover the cursor over a provider and click the pencil icon.**


Include
Lib
Scripts
pyvenv.cfg
firebase_auth_demo.py
External Libraries
Scratches and Consoles
firebaseConfig={
'storageBucket': "authdemo-d1fe2.appspot.com",
'messagingSenderId': "409748293750",
'appId': "1:409748293750:web:350b4331e8418160e7775f",
'measurementId': "G-MRSNWT79TS"}
firebase=pyrebase.initialize_app(firebaseConfig)
auth=firebase.auth()
def signup():
email=input("Enter email: ")
password=input("Enter password: ")
try:
user=auth.create_user_with_email_and_password(email,password)
print("Successfully created account!")
except:
print("Email already exists!")
def login():
email = input("Enter email: ")
password = input("Enter password: ")
ans=input("Are you a new user?(y/n)")
if ans=='y':
signup()
elif ans=='n':
login()


Firebase
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test La...
Analytics
Dashboard, Events, Conversions, A...
Grow
Predictions, A/B Testing, Cloud M...
Extensions
Spark
Free $0/month
Upgrade
AuthDemo
Go to docs
Authentication
Users
Sign-in method
Templates
Usage
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
dummymail@mail.com
Jun 3, 2020
Jun 3, 2020
JcC7dLBrBJfpTNNGy1fEp0JuQ5C2
Rows per page: 50
1-1 of 1



- Toggle the **Enable** button and, for third-party identity providers, enter the provider ID and secret from the provider's developer site. The Firebase docs give specific instructions in the "Before you begin" sections of theFacebook, Twitter, and GitHub guides. After enabling a provider, click **Save**.



- In the Firebase console, under **Authorized Domains**, click **Add Domain** and enter the domain of your app on App Engine in the following format:

```
[PROJECT_ID].appspot.com
```

Do not include `http://` before the domain name.

Reference:

https://cloud.google.com/appengine/docs/standard/python/authenticating-users-firebase-appengine