Configuration Manual

MSc Research Project
Cloud Computing

Prakhar Singhwal
Student ID: x23285435

School of Computing
National College of Ireland

Supervisor: Prof. Sean Heeney

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Prakhar Singhwal		
Student ID:	x23285435		
Programme:	Cloud Computing	**Year:**	2024-25
Module:	Research Project		
Supervisor:	Prof. Sean Heeney		
Submission Due Date:	11/08/2025		
Project Title:	Configuration Manual		
Word Count:	910	**Page Count:**	6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Prakhar Singhwal

Date: 11th August 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	

Date:	
Penalty Applied (if applicable):	

Configuration Manual

Prakhar Singhwal
x23285435

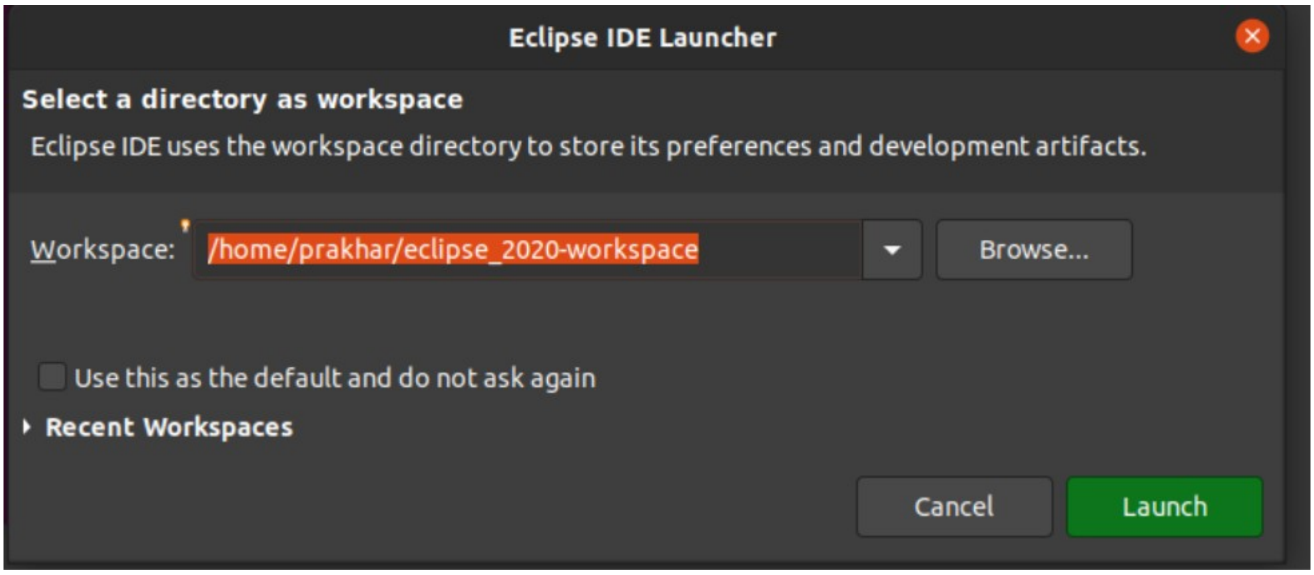
1 Preconditions

As the performance is executed using an artificial toolbox there is a dire need to set up an integrated development environment. Any of the IDE which support java can be used. To run the simulation Eclipse IDE is used. However, prior to that make sure the system has jdk 1.8 installed. After that install eclipse version 2020-03. Both these configurations are crucial because ifogsim2 only supports the above-mentioned versions. If not, the system will face many compile time errors. Additionally, it is also expected that the project-folder iFogSim-main is present on the system locally.

1. Download and install Eclipse (2020) version from Eclipse foundation website. Post that when the IDE is launched it has a logo which resembles diagram 1.

Figure 1: Eclipse start up logo

2. Construct the workspace and give it any name according to the convenience. In this case eclipse_2020-workspace is created as described in Figure 2.


Eclipse IDE Launcher

Select a directory as workspace

Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: /home/prakhar/eclipse_2020-workspace

Use this as the default and do not ask again

Recent Workspaces

Figure 2: Creating the workspace for the research

3. After the eclipse workspace is opened click on Import projects link under project explorer side menu. After clicking on it a dialog box will pop up. From that click on general and choose Existing projects into Workspace option and click next as depicted in figure 3.

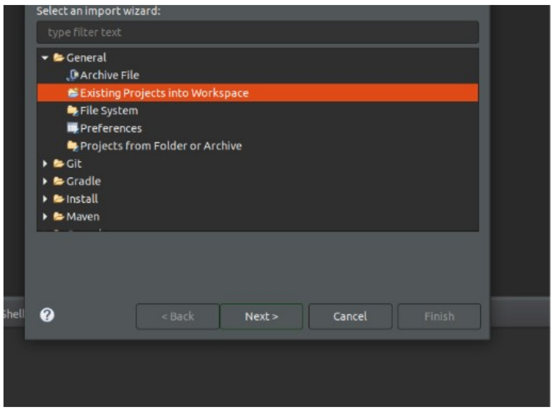
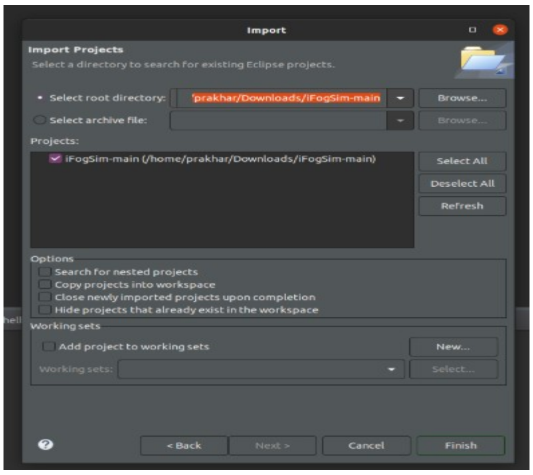


Figure 3: Import existing projects into Workspace

4. Succeeding that a new dialog box will open up. Click on the browse option and choose where on the system the project folder is present. After selecting the specified directory, it will be listed in the project section. Finally, click on the finish button as shown in diagram 4.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

[✓ Select root directory:] 'prakhar/Downloads/iFogSim-main

Select archive file:

Projects:

[✓] iFogSim-main (/home/prakhar/Downloads/iFogSim-main)

Options

[ ] Search for nested projects
[ ] Copy projects into workspace
[ ] Close newly imported projects upon completion
[ ] Hide projects that already exist in the workspace

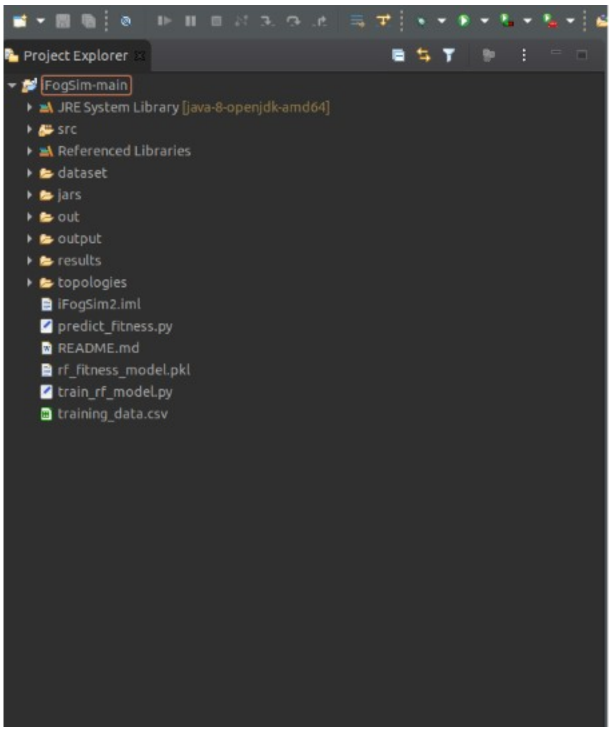
Working sets

[ ] Add project to working sets

Working sets:

Figure 4: 2nd import dialog box

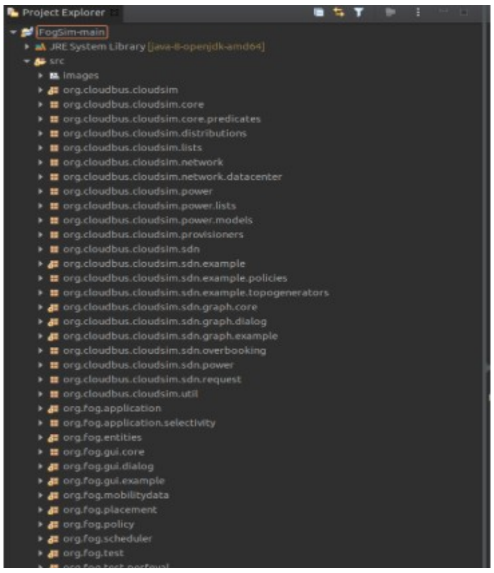
5. The entire research project folder will be seen on the left side of the menu bar under Project Explorer. The entire directory is shown in figure 5.



```
Project Explorer
▾ FogSim-main
  ▸ JRE System Library [java-8-openjdk-amd64]
  ▸ src
  ▸ Referenced Libraries
  ▸ dataset
  ▸ jars
  ▸ out
  ▸ output
  ▸ results
  ▸ topologies
    iFogSim2.iml
    predict_fitness.py
    README.md
    rf_fitness_model.pkl
    train_rf_model.py
    training_data.csv
```

Fig 5 : The entire project structure

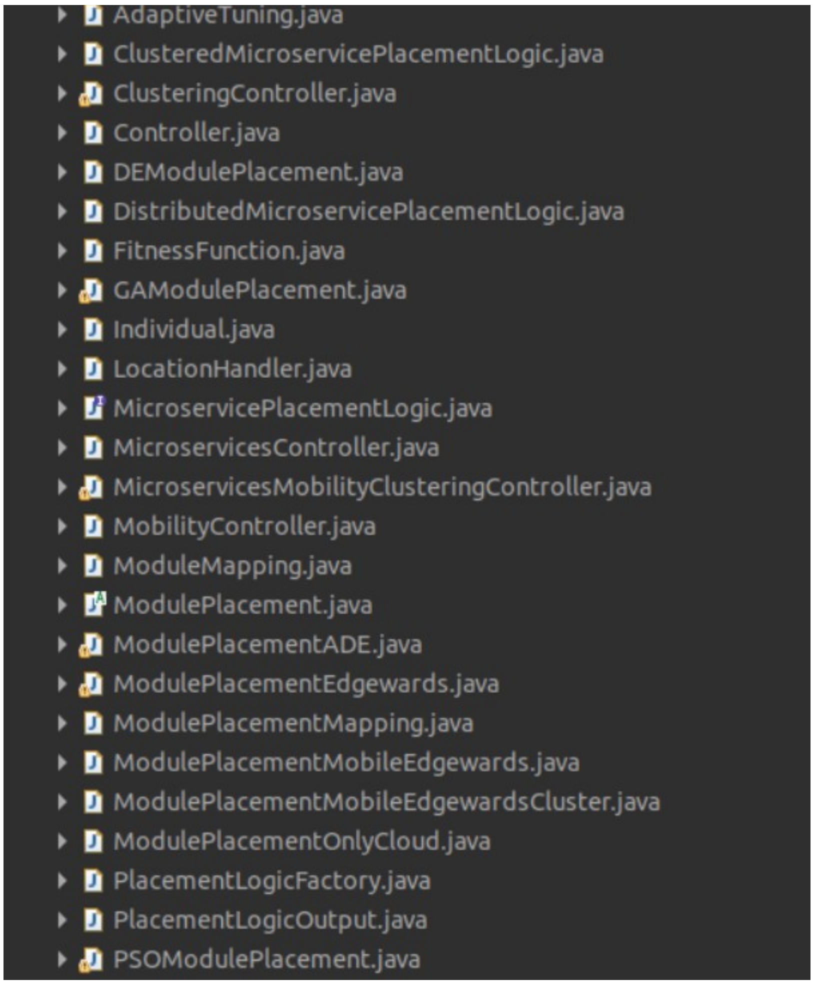
6. Navigate to src folder, all the preinstalled packages will appear here including placement code, base cloudsim classes which are inherited on ifogsim2.The main changes have been developed in placement and test package package.



```
Project Explorer
▾ FogSim-main
  ▸ JRE System Library [java-8-openjdk-amd64]
  ▾ src
    ▸ images
    ▸ 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.sdn
    ▸ org.cloudbus.cloudsim.sdn.example
    ▸ org.cloudbus.cloudsim.sdn.example.policies
    ▸ org.cloudbus.cloudsim.sdn.example.topogenerators
    ▸ org.cloudbus.cloudsim.sdn.graph.core
    ▸ org.cloudbus.cloudsim.sdn.graph.dialog
    ▸ org.cloudbus.cloudsim.sdn.graph.example
    ▸ org.cloudbus.cloudsim.sdn.overbooking
    ▸ org.cloudbus.cloudsim.sdn.power
    ▸ org.cloudbus.cloudsim.sdn.request
    ▸ org.cloudbus.cloudsim.util
    ▸ org.fog.application
    ▸ org.fog.application.selectivity
    ▸ org.fog.entities
    ▸ org.fog.gui.core
    ▸ org.fog.gui.dialog
    ▸ org.fog.gui.example
    ▸ org.fog.mobilitydata
    ▸ org.fog.placement
    ▸ org.fog.policy
    ▸ org.fog.scheduler
    ▸ org.fog.test
```

Fig 6: The packages inside the src folder

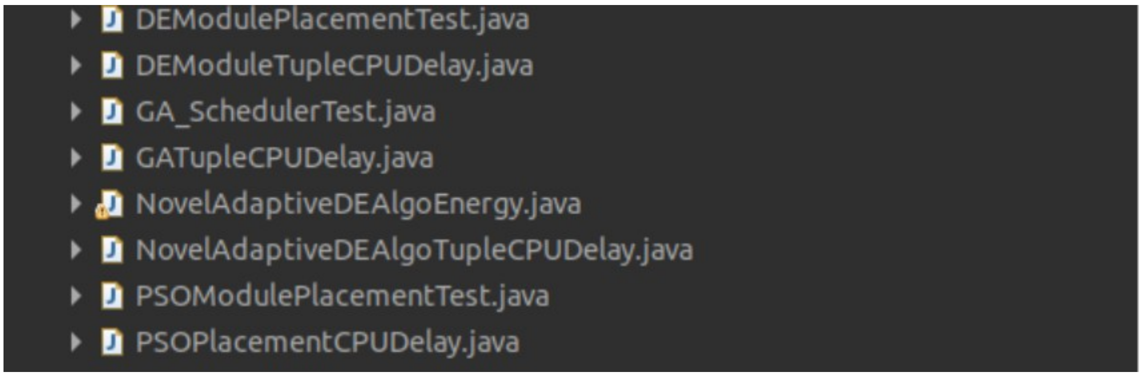
7. Adaptive Tuning, Fitness function, Individual, ModulePlacementADE clases have been implemented here which are the base for the proposed algorithm.Apart from the above mentioned classes other state of the art algorithms have also been implemented and be viewed in figure 7.



AdaptiveTuning.java
ClusteredMicroservicePlacementLogic.java
ClusteringController.java
Controller.java
DEModulePlacement.java
DistributedMicroservicePlacementLogic.java
FitnessFunction.java
GAModulePlacement.java
Individual.java
LocationHandler.java
MicroservicePlacementLogic.java
MicroservicesController.java
MicroservicesMobilityClusteringController.java
MobilityController.java
ModuleMapping.java
ModulePlacement.java
ModulePlacementADE.java
ModulePlacementEdgewards.java
ModulePlacementMapping.java
ModulePlacementMobileEdgewards.java
ModulePlacementMobileEdgewardsCluster.java
ModulePlacementOnlyCloud.java
PlacementLogicFactory.java
PlacementLogicOutput.java
PSOModulePlacement.java

Figure 7: ADE implementation

8. The main orchestrator of the proposed algorithm is defined in the org.fog.test package. It is the class from which the simulation starts. The algorithm have 2 seperate classes one for Energyconsumption and second for tuple cpu delay which can be seen clearly from figure 8.



DEModulePlacementTest.java
DEModuleTupleCPUDelay.java
GA_SchedulerTest.java
GATupleCPUDelay.java
NovelAdaptiveDEAlgoEnergy.java
NovelAdaptiveDEAlgoTupleCPUDelay.java
PSOModulePlacementTest.java
PSOPlacementCPUDelay.java

Fig 8: Main orchestrator classes in the algorithm implementation

9. The folder which contains the JSON configuration of fog topology is created as seen in the following figure 9. routerTopologyNovel is the JSON file which has the configuration of fog nodes for the proposed algorithm and the routerTopology is used for the other static algorithms such as DE,GA.

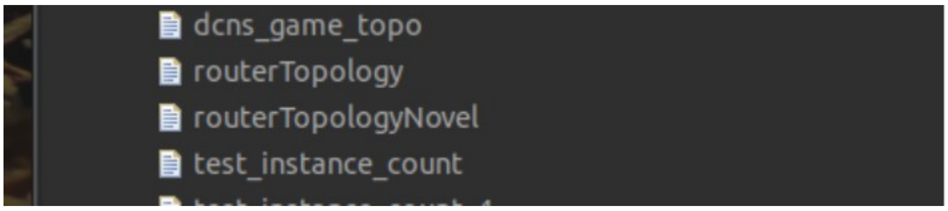
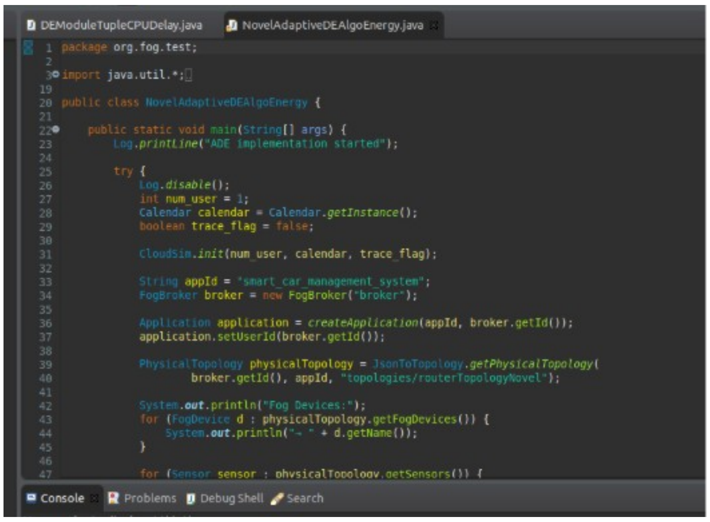


Fig 9: JSON topology for the project

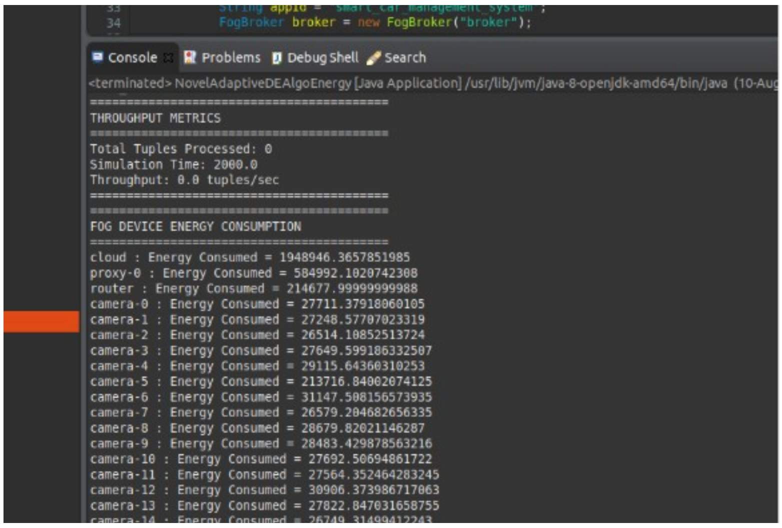
10. To execute the proposed algorithm double click on the NovelAdaptiveDEAlgoEnergy.java class. It will appear in the right-hand side of the screen.After that right click on the right-hand side and hover to RunAs and click as Java Application. Figure 10 will depict how the class is executed.



```java
package org.fog.test;

import java.util.*;

public class NovelAdaptiveDEAlgoEnergy {

    public static void main(String[] args) {
        Log.printLine("ADE implementation started");

        try {
            Log.disable();
            int num_user = 1;
            Calendar calendar = Calendar.getInstance();
            boolean trace_flag = false;

            CloudSim.init(num_user, calendar, trace_flag);

            String appId = "smart_car_management_system";
            FogBroker broker = new FogBroker("broker");

            Application application = createApplication(appId, broker.getId());
            application.setUserId(broker.getId());

            PhysicalTopology physicalTopology = JsonToTopology.getPhysicalTopology(
                    broker.getId(), appId, "topologies/routerTopologyNovel");

            System.out.println("Fog Devices:");
            for (FogDevice d : physicalTopology.getFogDevices()) {
                System.out.println("- " + d.getName());
            }

            for (Sensor sensor : physicalTopology.getSensors()) {
```

Fig 10: Need to run this class to get the simulation result

11. After running the main class, the output will be presented on the console with proper message displaying the energy consumption of all fog devices along with Tuple CPU execution delay with proper detailed responses as seen in figure 11.

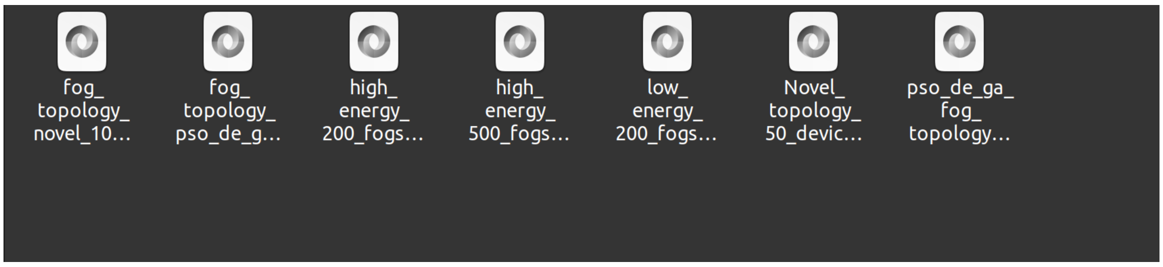


```
String appId = "smart_car_management_system";
FogBroker broker = new FogBroker("broker");
```

```
<terminated> NovelAdaptiveDEAlgoEnergy [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (10-Aug
========================================
THROUGHPUT METRICS
========================================
Total Tuples Processed: 0
Simulation Time: 2000.0
Throughput: 0.0 tuples/sec
========================================
========================================
FOG DEVICE ENERGY CONSUMPTION
========================================
cloud : Energy Consumed = 1948946.3657851985
proxy-0 : Energy Consumed = 584992.1020742308
router : Energy Consumed = 214677.99999999988
camera-0 : Energy Consumed = 27711.37918060105
camera-1 : Energy Consumed = 27248.57707023319
camera-2 : Energy Consumed = 26514.10852513724
camera-3 : Energy Consumed = 27649.599186332507
camera-4 : Energy Consumed = 29115.64360310253
camera-5 : Energy Consumed = 213716.84002074125
camera-6 : Energy Consumed = 31147.508156573935
camera-7 : Energy Consumed = 26579.204682656335
camera-8 : Energy Consumed = 28679.82021146287
camera-9 : Energy Consumed = 28483.429878563216
camera-10 : Energy Consumed = 27692.50694861722
camera-11 : Energy Consumed = 27564.352464283245
camera-12 : Energy Consumed = 30906.373986717063
camera-13 : Energy Consumed = 27822.847031658755
camera-14 : Energy Consumed = 26749.31499412243
```

Fig 11: The output of the console

12. To execute the proposed algorithm on different workloads such as 50,100 and 200 fog devices, JSON topology is used which load the data dynamically at runtime. The files will be uploaded as zip folder with name Results_Analysis along with the code artifact. Just copy the code from there and paste in the routerTopologyNovel JSON file. The files will have names as seen in figure 12.



```
fog_topology_novel_10...
fog_topology_pso_de_g...
high_energy_200_fogs...
high_energy_500_fogs...
low_energy_200_fogs...
Novel_topology_50_devic...
pso_de_ga_fog_topology...
```

Figure 12: JSON files for executing various workloads

References

Oracle. 2025. *Java SE Development Kit 8u461 downloads*. [Online] Available at: https://www.oracle.com/java/technologies/downloads/ [Accessed 10 August 2025].

Eclipse Foundation. 2020. *Eclipse IDE 2020-03 release*. [online] Available at: https://projects.eclipse.org/releases/2020-03 [Accessed 10 August 2025].