National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Deekshiya Seenivasagan
Student ID: x23268336

School of Computing
National College of Ireland

Supervisor: prof. Aqeel Kazmi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Deekshiya Seenivasagan
Student ID:	X23268336
Programme:	Cloud Computing **Year:** 2024-2025
Module:	MSCCLOUD Research Project
Lecturer:	Prof. Aqeel Kazmi
Submission Due Date:	15-September-2025
Project Title:	Optimizing Energy Efficiency and Latency in Mobile-Edge-Cloud Systems via PPO, LSTM Caching, and DVFS
Word Count:	**1165** **Page Count:** 11

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: Deekshiya Seenivasagan

Date: 15-September-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

Deekshiya Seenivasagan
Student ID: x23268336

1 Requirements

This project simulates task offloading, intelligent caching decisions and dynamically adjusts CPU frequency using DVFS in a mobile-edge-cloud environments. It is developed using Java, CloudSimPlus frameworks with machine learning models (PPO and LSTM in ONNX format). The project uses CloudSimPlus for Simulation and ONNX Runtime to load and apply machine learning models exported in ONNX format for real-time scheduling and caching. This manual provides a comprehensive guide to set up the simulation and development environment for the project. All the required tools, dependencies and configurations are covered for the successful project execution. The project is developed in Eclipse (as the primary IDE) using JDK 17 and Jupyter lab is used for training the machine learning models.

1. Prerequisites (Software requirements):

   - **Java JDK:** recommended: OpenJDK17 (Oracle, 2021)
   - **Eclipse IDE:** eclipse.org (Eclipse Foundation, 2025)
   - **CloudSim Plus:** cloudsimplus.org (CloudSim Plus, 2024)
   - **Python 3.11.9:** python.org (Python Software Foundation, 2024)
   - **Jupyter Lab** (Project Jupyter, 2025)

2. Download python and install jupyter lab via the terminal:

   - Install Python from python.org (Python Software Foundation, 2024)
   - pip install jupyterlab (Project Jupyter, 2025)

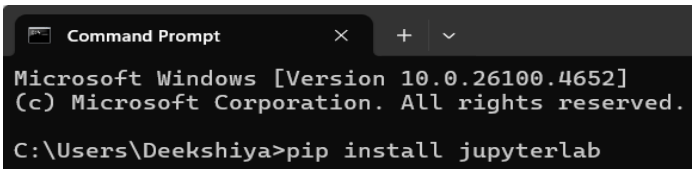

Figure1: Install Jupyter lab

   - Launch Jupyter Lab from command Prompt:

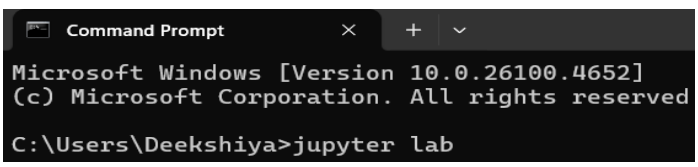

Figure2: Launch Jupyter lab

3. Download and install Eclipse IDE. Once downloaded launch the eclipse and you will see the loading page as in Figure3

Figure3: Eclipse

4. On the launch screen, set a relevant workspace name and location as you can see in the figure4

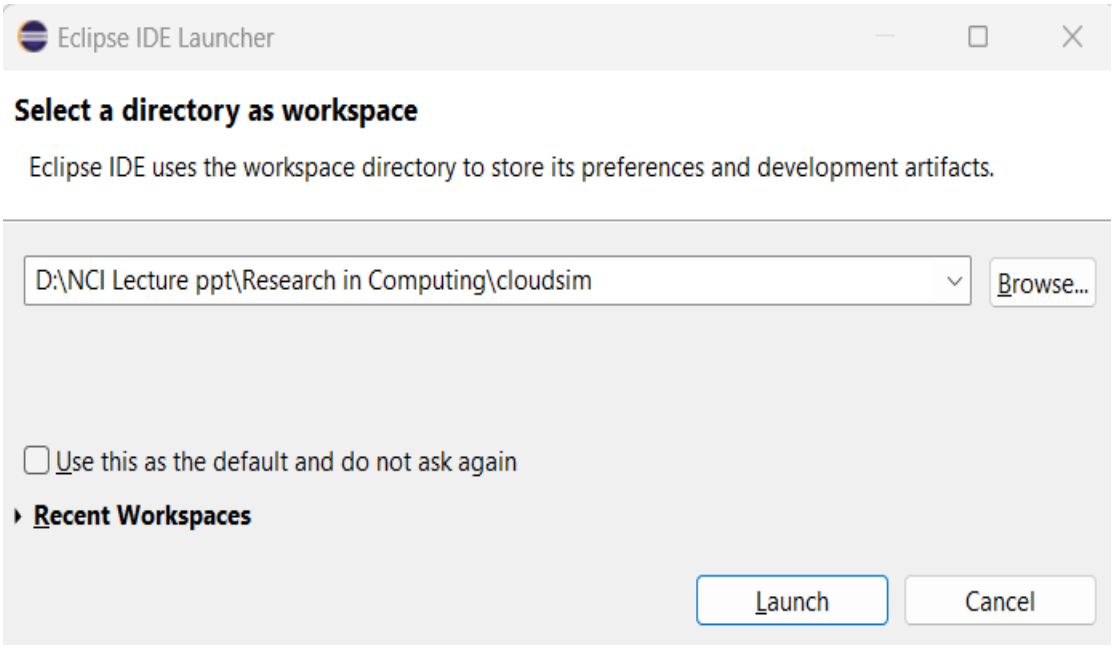

Figure4: Creating new Workspace for the project

5. Once the workspace is created, download the cloudsimplus tool from the GitHub (CloudSim Plus, 2024). After downloading unzip the folder and import them into eclipse as:
    - File – Click Import – Click Maven – Click Existing Maven Projects – Click Next – Root directory(click browse: select the unzipped cloudsimplus) as in Figure4 – click Finish

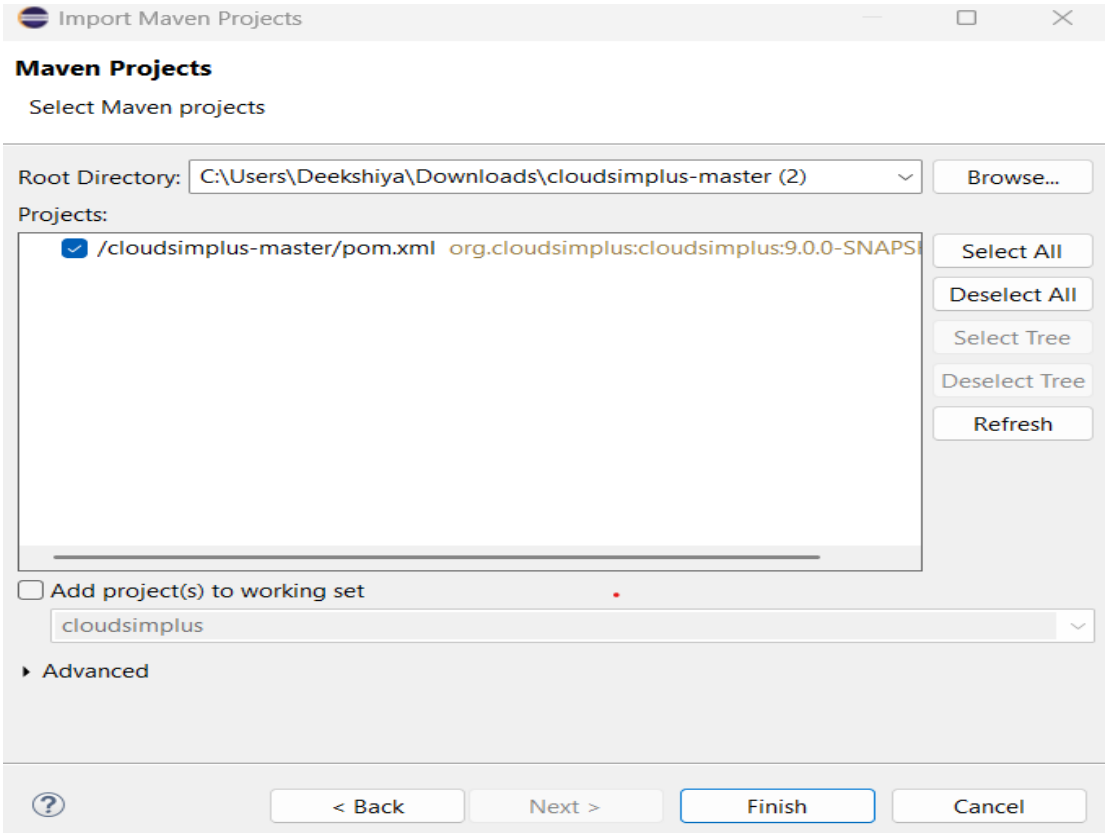

Figure5: Import the Cloudsimplus

6. Once it is successfully imported , you will see a package explorer like you see in the Figure6, expand src/main/java folder – inside that create a package called (**com.myorg.simulation**) – click Finish

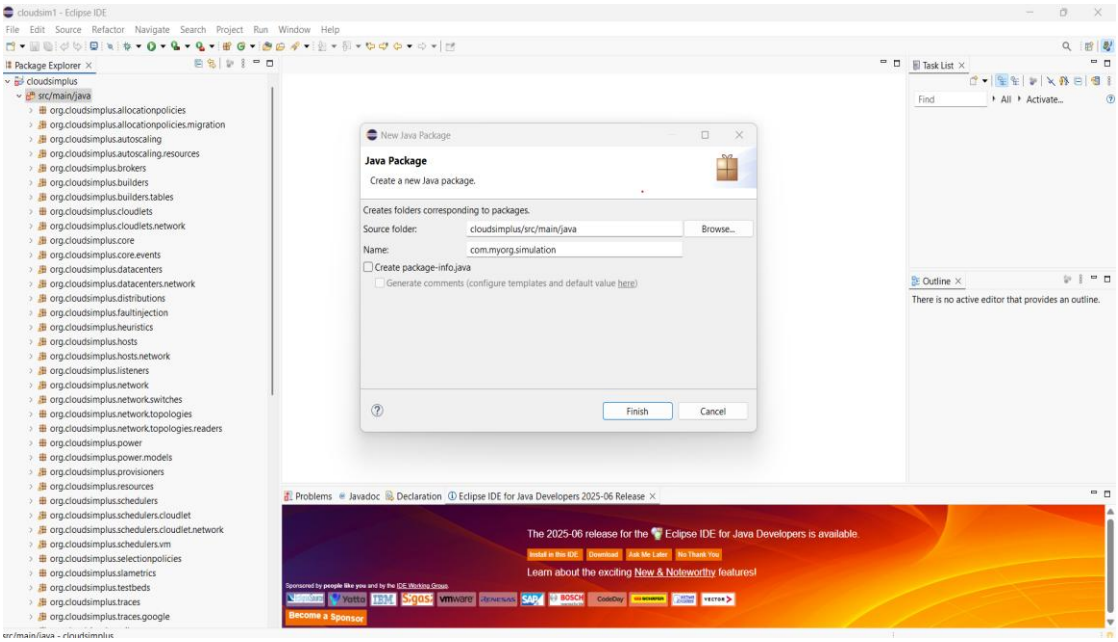

Figure6: Package Creation inside src/main/java

7. After creating the package (**com.myorg.simulation)** in the eclipse, copy and paste the provided (main.java and Smartbroker.java) files from the zip file

(x23268336_Artifacts – CloudSimPlus_RequiredFiles) into this package directory, as shown in the figure7 below:

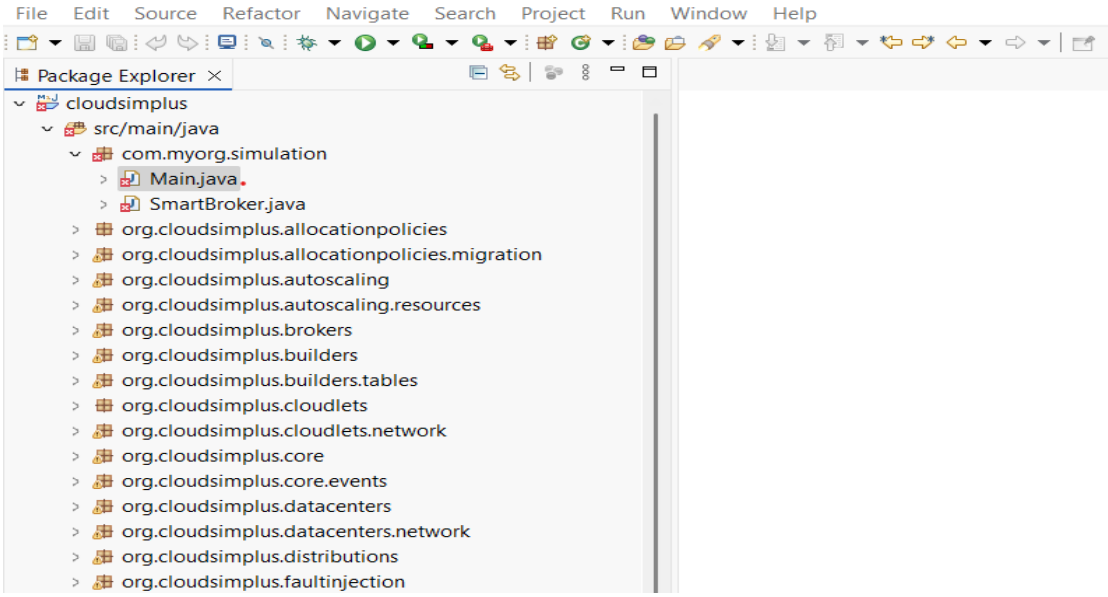

Figure7: Java files placed inside the com.myorg.simulation package.

8. Now, we need to add the required dependencies in the pom.xml. Paste the provided dependencies under the <dependencies> section as shown in Figure8. After updating the dependencies in pom.xml, Save the file and update Maven (Project - Update Maven Project) (CloudSim Plus, 2021; Microsoft, 2023; TensorFlow, 2023; Eclipse Deeplearning4j, 2021; JSON.org, 2024).

```
<!--  TensorFlow Java  -->
 <dependency>
        <groupId>org.tensorflow</groupId>
        <artifactId>tensorflow-core-platform</artifactId>
        <version>0.5.0</version>
</dependency>
<!--  ONNX Runtime  -->
<dependency>
        <groupId>com.microsoft.onnxruntime</groupId>
        <artifactId>onnxruntime</artifactId>
        <version>1.15.0</version>
</dependency>
<dependency>
        <groupId>org.cloudsimplus</groupId>
        <artifactId>cloudsim-plus</artifactId>
        <version>6.3.0</version>
</dependency>
<!--  ND4J backend for loading ONNX into DL4J  -->
<dependency>
        <groupId>org.nd4j</groupId>
        <artifactId>nd4j-native-platform</artifactId>
        <version>1.0.0-M1.1</version>
</dependency>
<dependency>
        <groupId>org.json</groupId>
        <artifactId>json</artifactId>
        <version>20240303</version>
</dependency>
```

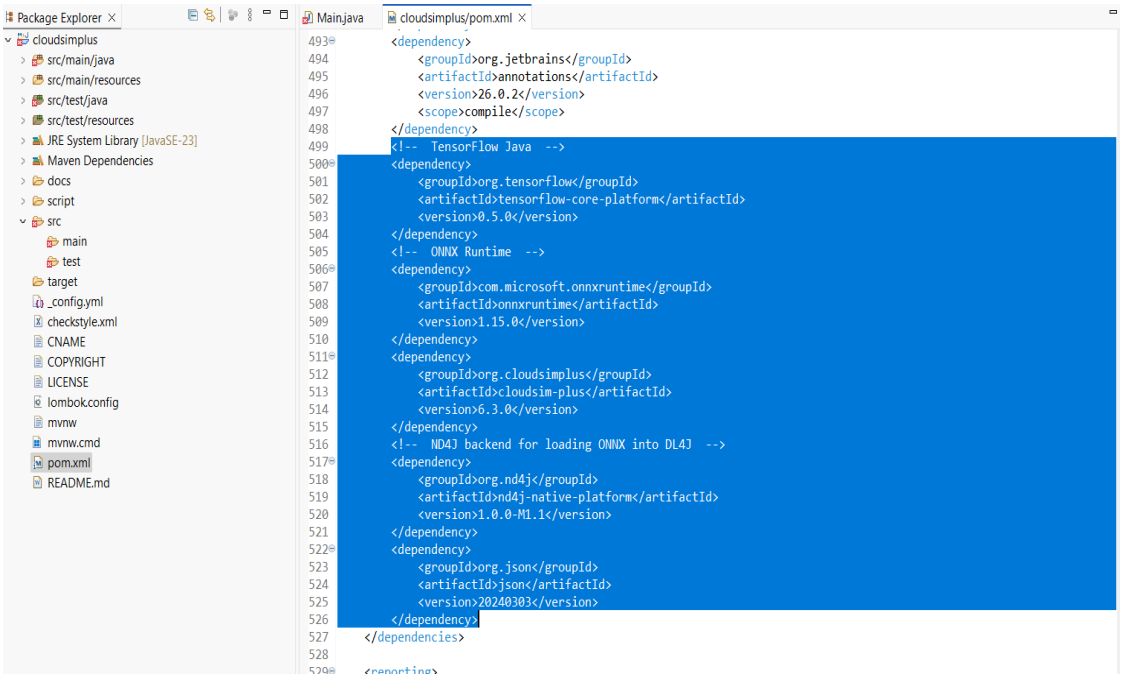

Figure8: Adding Dependencies in pom.xml

9. After successfully adding the required dependencies, now in the module-info.java, add **(requires com.microsoft.onnxruntime and requires org.json)** as shown in the Figure9. Comment out com.google.gson if it throws error in module-info.java

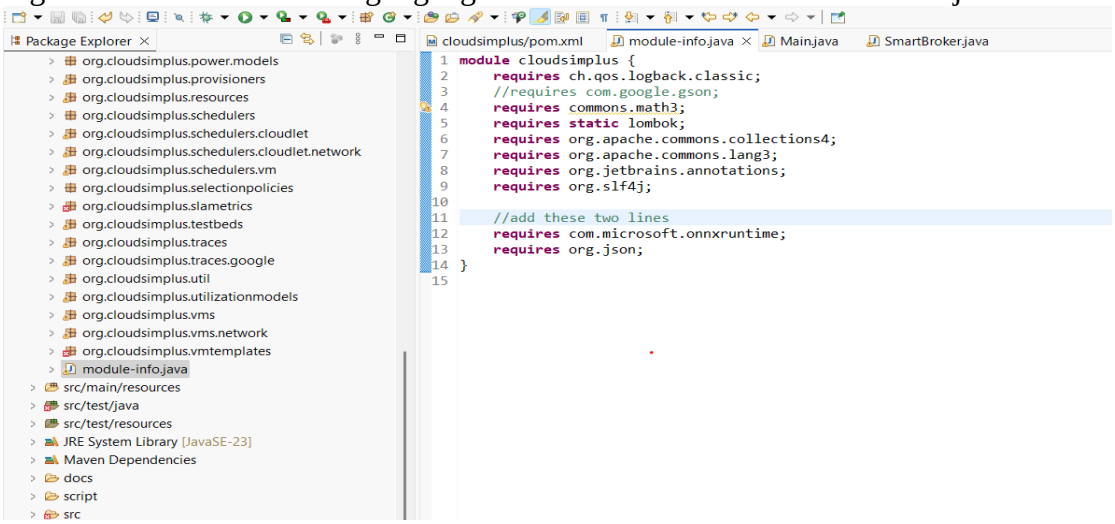

Figure9: module-info.java

10. After successfully adding the required dependencies, we need to download the ONXX models and CSV provided in the zip file (x23268336_Artifacts – CloudSimPlus_RequiredFiles) .
    - To create folder right click on the project – New – Folder – (Folder Name)
    - Create a folder called models in cloudsimplus as shown in Figure10. Place the ONNX models (lstm_model_more.onnx & ppo_with_lstm_cache30_bonus2_1.onnx) in the models folder
    - Create a folder called data in cloudsimplus as shown in Figure10. Place the CSV file (access_logs_for_sim_more.csv) in the data folder.
    - Similarly create folders results and logs.

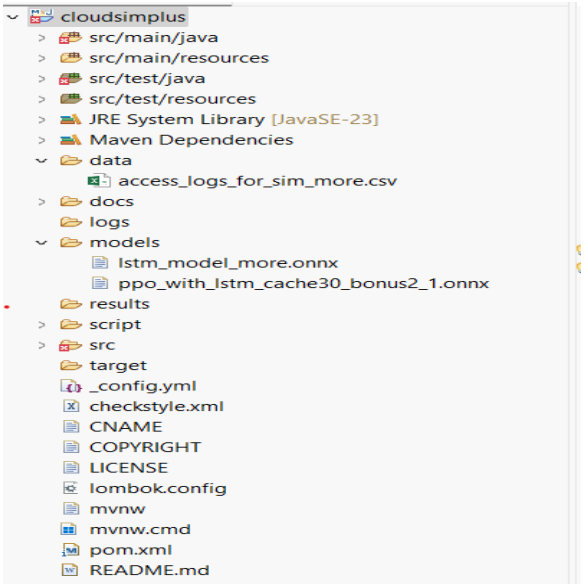

Figure10: Folder creation

11. In main.java, ensure paths are properly configured to the source code. Ensure below lines points correctly to the files:
    - String lstmPath = "models/lstm_model_more.onnx";
    - String ppoPath = "models/ppo_with_lstm_cache30_bonus2_1.onnx";
    - String logPath = "data/access_logs_for_sim_more.csv";

12. After verifying, in Eclipse right click the Main.java file and select run as java application. Simulation will execute and the outputs files in CSV format will be saved in the /results/ folder as in Figure11.

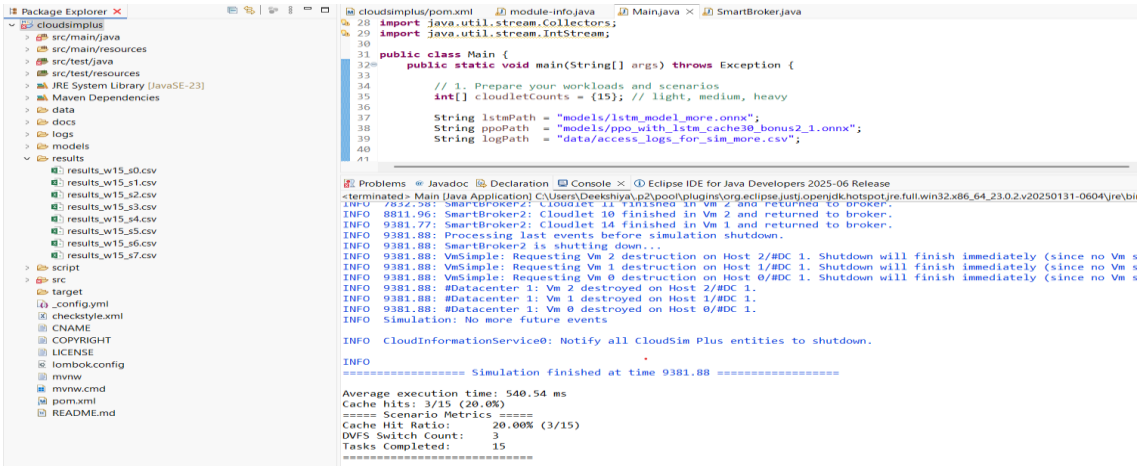

Figure11: Output can be seen in the console

13. Configuration Parameters:
    - Workloads: Set the cloudlet workload in Main.java: int[] cloudletCounts = {20, 70, 155}; We can also change the cloudlet counts.
    - Scenario Flags: PPO, LSTM, DVFS, etc. can be controlled via scenario loops or flags in Main.java.
    - Model and Data Paths: Update at the top of Main.java if you move or rename files.

2 Result and Analysis

After running the simulation in Eclipse (Eclipse Foundation, 2025), output CSV files are generated in the /results/ directory. To analyze these results:

- Copy the CSV files from the /results/ folder in the eclipse to the corresponding results folder in the Jupyter Lab workspace (Project Jupyter, 2025)
- Open the jupyter lab.
- Upload and launch the provided analysis script (**analyze_results.ipynb**) by clicking on "Upload Files" in Jupyter Lab.
- Click on Run - Run selected cell (or use the "Run" button in the toolbar) to execute each cell of the notebook and visualize your results.

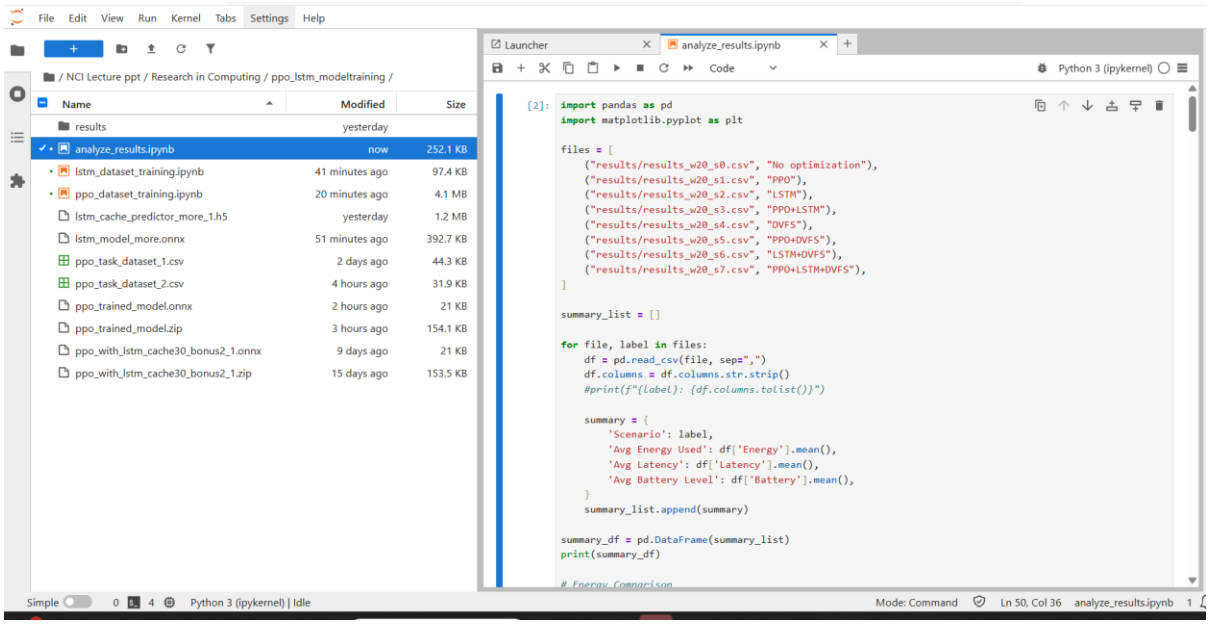

Figure 12: Running the analyze_results.ipynb script in Jupyter Lab to analyze and summarize CloudSim Plus simulation results.

3 Training Machine Learning models (optional)

All training scripts are implemented in **Jupyter Lab** (Project Jupyter, 2024). For this research, two machine learning models were trained to optimize task offloading and caching:

- **LSTM (Long Short-Term Memory)**: Used for predicting cache hit/miss and user demand patterns.
- **PPO (Proximal Policy Optimization)**: A reinforcement learning model for resource allocation and CPU frequency scaling decisions.

1. Download python and install jupyter lab via the terminal.

2. Data Preparation: The input data for training is provided in the CSV format:
   - lstm_access_logs_more.csv for LSTM training.
   - ppo_task_dataset_1.csv for PPO training.

3. Model Training Steps:

To retrain the models, open the corresponding notebook in Jupyter Lab, modify data or parameters as needed, re-run all cells, and re-export the model to ONNX. Use the new ONNX model for simulation in CloudSimPlus.

A. LSTM Model Training:
- Open the notebook (lstm_dataset_training.ipynb) in Jupyter Lab.
- Load the dataset (lstm_access_logs_more.csv).
- Preprocess data: Normalize, create input sequences, and split into training/testing.
- Define and train the LSTM model.
- Export the trained model to ONNX format (Output: lstm_model_more.onnx)

B. PPO Model Training:
- Open (ppo_dataset_training.ipynb) in Jupyter Lab.
- Load the dataset (ppo_task_dataset_2.csv).
- Preprocess data as needed for RL training.
- Define the environment and PPO agent. Train the PPO agent on the environment.
- Export the trained policy to ONNX format (output: ppo_with_lstm_cache30_bonus2_1.onnx)

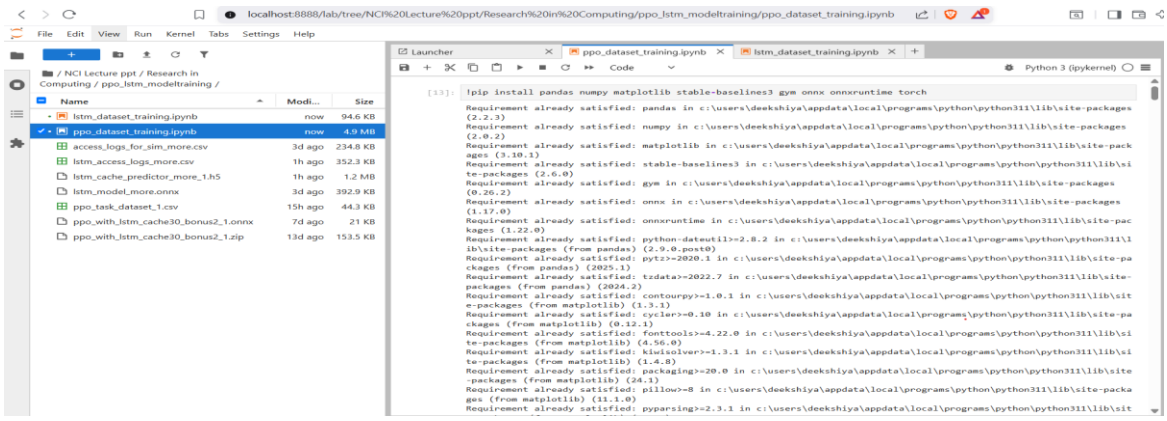

Figure13: Jupyter Lab

4 Repositories

- Project Source Code: https://github.com/Deekshiya/x2326833_Research_Project.git

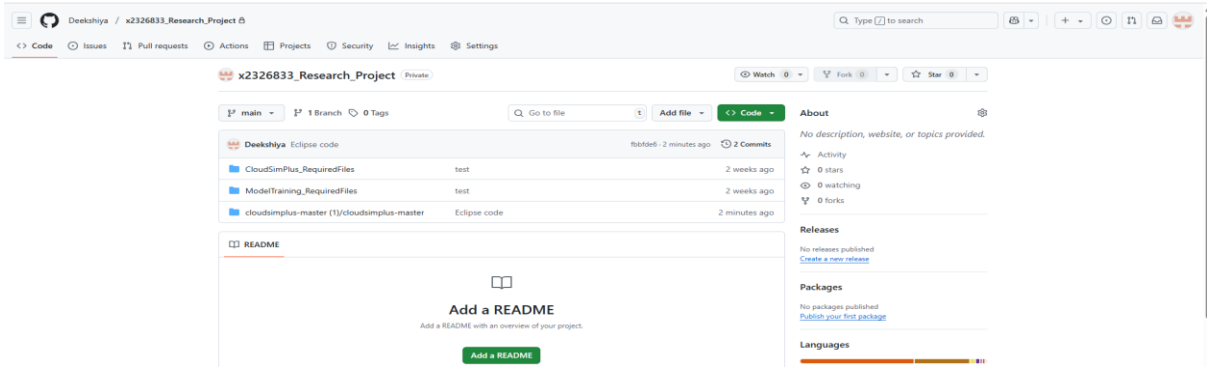

Figure14: GitHub Repository (GitHub, 2025).

References

Eclipse Foundation. (2025). *Eclipse downloads* [Computer software]. https://www.eclipse.org/downloads/

GitHub. (2025). *GitHub* [Website]. https://github.com/

Python Software Foundation. (2024). *Python* (Version 3.11.9) [Computer software]. https://www.python.org/downloads/release/python-3119/

Project Jupyter. (2025). *JupyterLab* [Computer software]. https://jupyter.org/

Oracle. (2021). *Java downloads* [Computer software]. https://www.oracle.com/java/technologies/downloads/

CloudSim Plus. (2024). *CloudSim Plus: Cloud computing simulation framework* [Computer software]. https://cloudsimplus.org/

Microsoft. (2023). *ONNX Runtime* (Version 1.15.0) [Maven Central Repository]. https://central.sonatype.com/artifact/com.microsoft.onnxruntime/onnxruntime/1.15.0

TensorFlow. (2023). *tensorflow-core-platform* (Version 0.5.0) [Maven Central Repository]. https://central.sonatype.com/artifact/org.tensorflow/tensorflow-core-platform/0.5.0

CloudSim Plus. (2021). *cloudsim-plus* (Version 6.3.0) [Maven Central Repository]. https://central.sonatype.com/artifact/org.cloudsimplus/cloudsim-plus/6.3.0

Eclipse Deeplearning4j. (2021). *nd4j-native-platform* (Version 1.0.0-M1.1) [Maven Central Repository]. https://central.sonatype.com/artifact/org.nd4j/nd4j-native-platform/1.0.0-M1.1

JSON.org. (2024). *json* (Version 20240303) [Maven Central Repository]. https://central.sonatype.com/artifact/org.json/json/20240303