Experiment 2.1

Student Name: **Saksham Goyal**   UID: **21BCS7981**
Branch: **CSE**   Section/Group: **CC_632-A**
Semester: **6ᵗʰ**   Date of Performance: **19/02/24**
Subject Name: **Cloud Computing and**   Subject Code: **21CSP-378**
                Distributed Systems Lab

1. Aim: Simulate a cloud scenario using Matlab and run a scheduling algorithm.

2. Objective:
   i. To learn about scheduling algorithms.
   ii. To learn about cloud computing.

3. Input/Apparatus Used:Input:
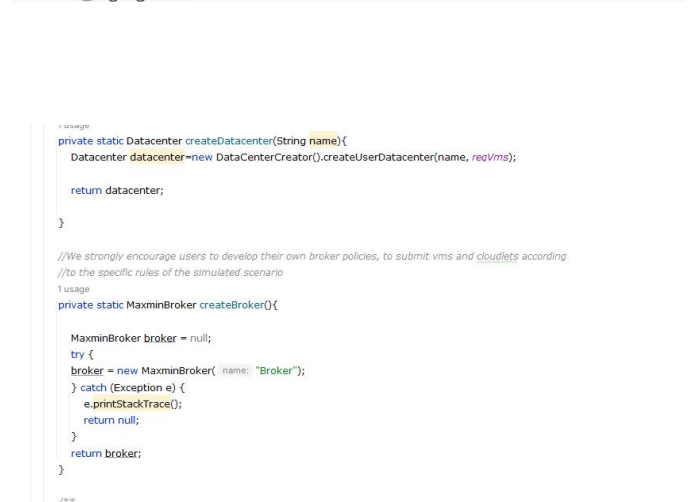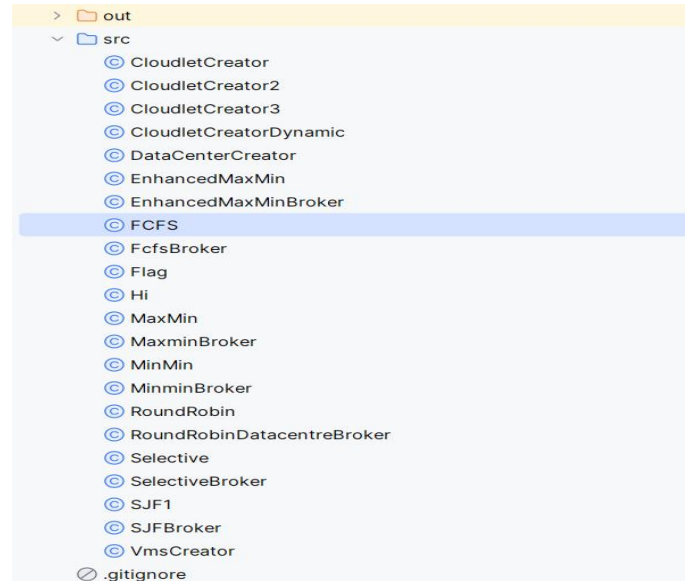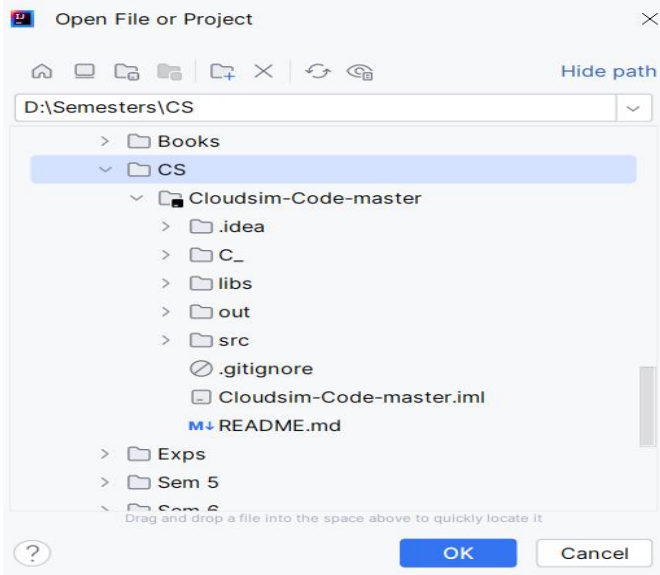   i. **Hardware Requirements**: - Minimum 4GB RAM, 120 GB hard Disk, processor with 2.1 MHz

4. Theory:-

MATLAB: MATLAB is a high-level programming language and interactive environment primarily used for numerical computation, data analysis, and visualization. It offers powerful tools for matrix computation, numerical analysis, data visualization, algorithm development, and integration with other languages.
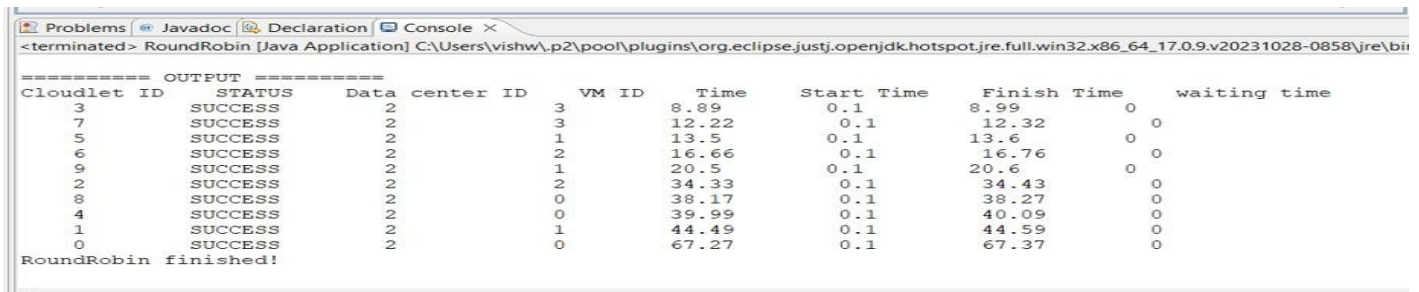
Scheduling Algorithms: Scheduling algorithms are essential in computer science and operating systems for managing tasks and resources efficiently. Common algorithms include FCFS, SJF, priority scheduling, round-robin, multi-level queues, and deadline-based scheduling. Each algorithm aims to optimize metrics like throughput, latency, fairness, and resource utilization based on specific system requirements and objectives.

5. PROCEDURE:
  i. Now first of all we need a few of the installations like IDE for JAVA(Eclipse), JDK, cloud sim package and JRE
  ii. open that installer now and wait for the installer to start other two things are jdk java development kit and the java runtime environment which is needed for running the java applications
  iii. click on the option for eclipse id for java developers it will automatically provide a workspace where that particular all the eclipse projects will be residing
  iv. You can choose any of the workspace which you have defined in your computer so choose that one eclipse id is starting
  v. create a new java project so you can name it sjf scheduler or anything you want and finish
  vi. Run the project.

```
                                                          1 usage
1 usage                                                   private static Datacenter createDatacenter(String name){
private static void printCloudletList(List<Cloudlet> list) {   Datacenter datacenter =new DataCenterCreator().createUserDatacenter(name, reqVms);
    int size = list.size();
    Cloudlet cloudlet;                                        return datacenter;

    String indent = "   ";                                }
    Log.printLine();
    Log.printLine("========== OUTPUT ==========");        //We strongly encourage users to develop their own broker policies, to submit vms and cloudlets according
    Log.printLine("Cloudlet ID" + indent + "STATUS" + indent +  //to the specific rules of the simulated scenario
        "Data center ID" + indent + "VM ID" + indent + "Time" + indent + "Start Time" + indent + "Finish Time");   1 usage
                                                          private static MaxminBroker createBroker(){
    DecimalFormat dft = new DecimalFormat( pattern: "###.##");
    for (int i = 0; i < size; i++) {                          MaxminBroker broker = null;
        cloudlet = list.get(i);                               try {
        Log.print(indent + cloudlet.getCloudletId() + indent + indent);   broker = new MaxminBroker( name: "Broker");
                                                              } catch (Exception e) {
        if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){   e.printStackTrace();
            Log.print("SUCCESS");                                 return null;
                                                              }
        Log.printLine( indent + indent + cloudlet.getResourceId() + indent + indent + indent + cloudlet.getVmId() +   return broker;
            indent + indent + dft.format(cloudlet.getActualCPUTime()) + indent + indent + dft.format(cloudlet.getExecStartTime())+   }
            indent + indent + dft.format(cloudlet.getFinishTime()));
        }
    }
}
```

6. Result: Thus the Scheduling algorithm is implemented.


```
Problems  Javadoc  Declaration  Console X
<terminated> RoundRobin [Java Application] C:\Users\vishw\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin

========== OUTPUT ==========
Cloudlet ID    STATUS     Data center ID    VM ID     Time    Start Time    Finish Time    waiting time
    3          SUCCESS          2              3       8.89       0.1          8.99           0
    7          SUCCESS          2              3      12.22       0.1         12.32           0
    5          SUCCESS          2              1      13.5        0.1         13.6            0
    6          SUCCESS          2              2      16.66       0.1         16.76           0
    9          SUCCESS          2              1      20.5        0.1         20.6            0
    2          SUCCESS          2              2      34.33       0.1         34.43           0
    8          SUCCESS          2              0      38.17       0.1         38.27           0
    4          SUCCESS          2              0      39.99       0.1         40.09           0
    1          SUCCESS          2              1      44.49       0.1         44.59           0
    0          SUCCESS          2              0      67.27       0.1         67.37           0
RoundRobin finished!
```

7. Learning Outcomes:
    i. Learnt Scheduling algorithms.
    ii. Learnt about cloud computing.
    iii. Learnt about Cloud Sim.