SSM INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution)
Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai
Sindalagundu Post, Palani Road, Dindigul – 624 002
Ph: 0451 – 2448800 – 2448899 (100 lines) Fax: 0451 – 2557755
E-mail: info@ssmiet.com, website: www.ssmiet.com

CCS362 - SECURITY AND PRIVACY IN CLOUD

RECORD NOTEBOOK

NAME	
REGISTER NUMBER	
YEAR	III
SEMESTER	VI
DEPARTMENT	Computer Science and Engineering (Cyber Security)
ACADEMIC YEAR	2025 – 2026 (Even semester)

SSM INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution)
Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai
Sindalagundu Post, Palani Road, Dindigul – 624 002
Ph: 0451 – 2448800 – 2448899 (100 lines) Fax: 0451 – 2557755
E-mail: info@ssmiet.com, website: www.ssmiet.com

PRACTICAL RECORD BONAFIDE CERTIFICATE

REGISTER NUMBER | |

Certified that this is the bonafide record work done by

Mr./Miss……***Reg. No***……

Fifth Semester, ***Computer Science and Engineering(Cyber Security)*** *branch during the*

Academic year ***2025 - 2026*** *in the* ***CCS362 - SECURITY AND PRIVACY IN CLOUD***

Staff in-Charge ***Head of the Department***

Submitted for the Semester End Practical Examination held on ……..

Internal Examiner ***External Examiner***

SSM INSTITUTE OF ENGINEERING AND TECHNOLOGY (Autonomous)
AICTE Approved / Affiliated to Anna University / Accredited by NAAC (2029) & NBA (2025)
Dindigul – Palani Highway, Dindigul 624 002.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(CYBER SECURITY)

VISION

To develop skilled cyber security professionals equipped to secure digital landscapes, address emerging cyber challenges, and contribute to society with strong technical expertise, entrepreneurial skills, and ethical values.

MISSION

- Foster self-discipline and critical thinking in students through robust teaching and learning.
- Empower students to become proficient cyber security professionals and responsible citizens.
- Strengthen industry partnerships by establishing specialized centres for advanced skill development and practical exposure.
- Deliver knowledge for secure and innovative solutions, contributing to sustainable and ethical technology advancement.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1	Graduates can apply their technical competence in computer science to solve real world problems, with technical and people leadership.
PEO2	Graduates Conduct cutting edge research and develop solutions on problems of social relevance.
PEO3	Graduates will Work in a business environment, exhibiting team skills, work ethics, adaptability and lifelong learning.

PROGRAM OUTCOMES (POs)

Graduate Attribute

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

The Students will be able to

- Exhibit design and programming skills to build and automate business solutions using cutting edge technologies.
- Strong theoretical foundation leading to excellence and excitement towards research, to provide elegant solutions to complex problems.
- Ability to work effectively with various engineering fields as a team to design, build and develop system applications

SSM INSTITUTE OF ENGINEERING AND TECHNOLOGY (Autonomous)
AICTE Approved / Affiliated to Anna University / Accredited by NAAC (2029) & NBA (2025)
Dindigul – Palani Highway, Dindigul 624 002.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(CYBER SECURITY)

Do's and Don'ts

Laboratory Rules &Regulation:

- Students are instructed to maintain silence inside the Lab.
- Students have to sign the log-book, while entering and leaving the Lab and also they have to mention the time in and time out.
- Students have to enter and leave the Lab in their scheduled time otherwise they will be marked absent.
- Students should come with proper Lab uniform and with shoes.
- The students should properly shut down the Computer Systems before they leave the Lab.
- Students are not allowed to use CD's & DVD's, USB DRIVE etc. lf required prior permission of Laboratory in-charge is needed.
- All students will be responsible for keeping the Lab clean.
- Students should refrain from dislocating, shifting and damaging with any parts of the computer or any other device in the Lab.
- The students should not load or delete any software from the computer.
- The students should not use computers in the Lab for any personal work.
- Browsing of Internet will not be allowed in the lab beyond the stipulated hour as per time table.
- The Instructor/Lecturer will be the sole authority to judge the disciplinary behavior inside the laboratory. For violation of any of the above rules, the department reserves the right to take appropriate disciplinary action.
- Browsing of non-academic Internet sites will not be allowed in the Lab.
- Before downloading any materials please consult your instructor and save the downloaded files as per instruction given by the laboratory in- charge.

- Because of security problems, downloading software and music etc. from the Internet is strictly prohibited. Any such file found in the hard disk will be deleted without warning.
- Students should arrange the chairs properly while leaving the LAB hours.
- Students should not allow to work inside the LAB other than LAB hours. If required prior permission of Laboratory in-charge and Department in charge is needed.

CCS362 **SECURITY AND PRIVACY IN CLOUD** **L T P C**
2 0 2 3

COURSE OBJECTIVES:

- To Introduce Cloud Computing terminology, definition & concepts
- To understand the security design and architectural considerations for Cloud
- To understand the Identity, Access control in Cloud
- To follow best practices for Cloud security using various design patterns
- To be able to monitor and audit cloud applications for security

PRACTICAL EXERCISES: **30 PERIODS**

1. Simulate a cloud scenario using Cloud Sim and run a scheduling algorithm not present in Cloud Sim
2. simulate resource management using cloud sim
3. simulate log forensics using cloud sim
4. simulate a secure file sharing using a cloud sim
5. Implement data anonymization techniques over the simple dataset (masking, k-anonymization, etc)
6. Implement any encryption algorithm to protect the images
7. Implement any image obfuscation mechanism
8. Implement a role-based access control mechanism in a specific scenario
9. implement an attribute-based access control mechanism based on a particular scenario
10. Develop a log monitoring system with incident management in the cloud

COURSE OUTCOMES:

CO1: Understand the cloud concepts and fundamentals.

CO2: Explain the security challenges in the cloud.

CO3: Define cloud policy and Identity and Access Management.

CO4: Understand various risks and audit and monitoring mechanisms in the cloud.

CO5: Define the various architectural and design considerations for security in the cloud

TABLE OF CONTENTS

EX:NO: 1	Simulate a cloud scenario using cloud Sim and run a scheduling algorithm not present in cloud Sim
DATE:	

AIM:

To simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim

PROCEDURE:

1. Install Java Development Kit (JDK) and download CloudSim (v3.0.3 or later).
2. Copy all CloudSim JAR files into a single folder.
3. Create a Java file named CustomSchedulingSimulation.java and save the program.
4. Open Command Prompt and navigate to the CloudSim JAR directory.
5. Import required CloudSim and Java packages in the program.
6. Create a Datacenter with required host configurations, create a DatacenterBroker, and create Virtual Machines and Cloudlets, then submit them to the broker.
7. Implement a custom scheduling algorithm not available in CloudSim.
8. Compile the program using the command:

   javac -cp "*" CustomSchedulingSimulation.java

9. Run the program using the command:

   java -cp ".;*" CustomSchedulingSimulation

10. Observe the simulation output showing cloudlet execution details.

PROGRAM:

```
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.*;
import java.util.*;

public class CustomSchedulingSimulation {

    public static void main(String[] args) {

        CloudSim.init(1, Calendar.getInstance(), false);

        List<Pe> pe = Arrays.asList(new Pe(0,new PeProvisionerSimple(1000)));
        Host h = new Host(0,new RamProvisionerSimple(2048),
                new BwProvisionerSimple(10000),1000000,
                pe,new VmSchedulerTimeShared(pe));
```

```
        Datacenter dc;
        try {
           dc = new Datacenter("DC",
                 new DatacenterCharacteristics(
                       "x86","Linux","Xen",
                       Arrays.asList(h),10,3,0.05,0.001,0),
                 new VmAllocationPolicySimple(Arrays.asList(h)),
                 new ArrayList<>(),0);
        } catch(Exception e){ return; }

        DatacenterBroker b;
        try { b = new DatacenterBroker("B"); }
        catch(Exception e){ return; }

        b.submitVmList(Arrays.asList(
              new Vm(0,b.getId(),1000,1,512,1000,10000,
                    "Xen",new CloudletSchedulerTimeShared())
        ));

        List<Cloudlet> cl = Arrays.asList(
              new Cloudlet(0,8000,1,300,300,
                    new UtilizationModelFull(),
                    new UtilizationModelFull(),
                    new UtilizationModelFull()),
              new Cloudlet(1,2000,1,300,300,
                    new UtilizationModelFull(),
                    new UtilizationModelFull(),
                    new UtilizationModelFull())
        );
        cl.forEach(c -> c.setUserId(b.getId()));

        cl.sort(Comparator.comparingLong(Cloudlet::getCloudletLength));

        b.submitCloudletList(cl);

        CloudSim.startSimulation();
        CloudSim.stopSimulation();

        b.getCloudletReceivedList()
         .forEach(c -> System.out.println(
            "Cloudlet "+c.getCloudletId()+" finished"));
    }
}
```

OUTPUT :


```
Microsoft Windows [Version 10.0.26200.7462]
(c) Microsoft Corporation. All rights reserved.

E:\Cloudsim>javac -cp "*" CustomSchedulingSimulation.java

E:\Cloudsim>java -cp ".;*" CustomSchedulingSimulation
Initialising...
Starting CloudSim version 3.0
DC is starting...
B is starting...
Entities started.
0.0: B: Cloud Resource List received with 1 resource(s)
0.0: B: Trying to Create VM #0 in DC
0.1: B: VM #0 has been created in Datacenter #2, Host #0
0.1: B: Sending cloudlet 1 to VM #0
0.1: B: Sending cloudlet 0 to VM #0
4.1: B: Cloudlet 1 received
10.1: B: Cloudlet 0 received
10.1: B: All Cloudlets executed. Finishing...
10.1: B: Destroying VM #0
B is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
DC is shutting down...
B is shutting down...
Simulation completed.
Simulation completed.
Cloudlet 1 finished
Cloudlet 0 finished

E:\Cloudsim>
```

RESULT:

Thus, to simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim is executed successfully.

EX:NO: 2	Simulate resource management using cloud sim
DATE:	

AIM:

To simulate resource management in a cloud environment using CloudSim by allocating and utilizing virtual machines and cloudlets efficiently.

PROCEDURE:

1. Install Java Development Kit (JDK) on the system.
2. Download CloudSim (version 3.0) and keep all JAR files in one folder.
3. Create a Java file named ResourceManagementSimulation.java and save the program.
4. Open Command Prompt and navigate to the CloudSim JAR directory.
5. Import required CloudSim and Java packages in the program.
6. Create a Datacenter with required host configurations, a DatacenterBroker, Virtual Machines, and Cloudlets, and submit them to the broker.
7. Initialize the CloudSim environment using CloudSim.init().
8. Compile the program using the command:

   javac -cp "*" ResourceManagementSimulation.java

9. Run the program using the command:

   java -cp ".;*" ResourceManagementSimulation

10. Observe the output showing resource allocation and cloudlet execution.

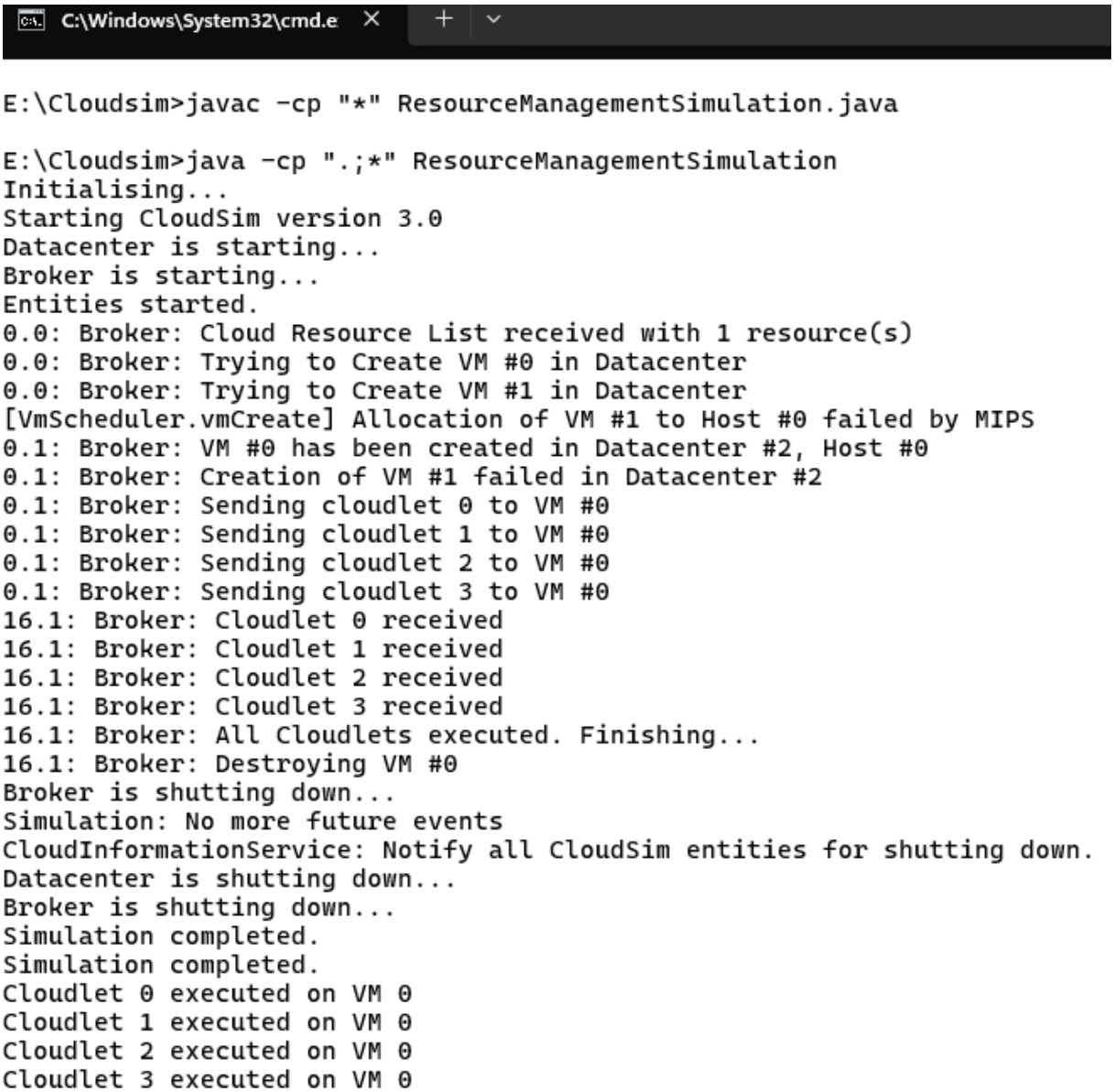
PROGRAM:

```
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.*;
import java.util.*;

public class ResourceManagementSimulation {
   public static void main(String[] args) {

      CloudSim.init(1, Calendar.getInstance(), false)
      Datacenter dc = createDatacenter();
      DatacenterBroker broker = createBroker();
```

```
    List<Vm> vmList = new ArrayList<>();
    for (int i = 0; i < 2; i++) {
        vmList.add(new Vm(
                i, broker.getId(), 1000, 1,
                512, 1000, 10000,
                "Xen", new CloudletSchedulerTimeShared()));
    }
    broker.submitVmList(vmList);


    List<Cloudlet> cloudletList = new ArrayList<>();
    for (int i = 0; i < 4; i++) {
        Cloudlet c = new Cloudlet(
                i, 4000, 1, 300, 300,
                new UtilizationModelFull(),
                new UtilizationModelFull(),
                new UtilizationModelFull());
        c.setUserId(broker.getId());
        cloudletList.add(c);
    }
    broker.submitCloudletList(cloudletList);
    CloudSim.startSimulation();
    CloudSim.stopSimulation();
    for (Cloudlet c : broker.getCloudletReceivedList()) {
        System.out.println("Cloudlet " + c.getCloudletId()
                + " executed on VM " + c.getVmId());
    }
}


static Datacenter createDatacenter() {
    List<Pe> peList =
            Arrays.asList(new Pe(0,
                    new PeProvisionerSimple(1000)));

    Host host = new Host(
            0,
            new RamProvisionerSimple(4096),
```

```
                new BwProvisionerSimple(10000),
                1000000,
                peList,
                new VmSchedulerTimeShared(peList));

        DatacenterCharacteristics ch =
                new DatacenterCharacteristics(
                    "x86", "Linux", "Xen",
                    Arrays.asList(host),
                    10, 3, 0.05, 0.001, 0);

        try {
            return new Datacenter(
                    "Datacenter",
                    ch,
                    new VmAllocationPolicySimple(Arrays.asList(host)),
                    new ArrayList<>(), 0);
        } catch (Exception e) {
            return null;
        }
    }

    static DatacenterBroker createBroker() {
        try {
            return new DatacenterBroker("Broker");
        } catch (Exception e) {
            return null;
        }
    }
}
```

OUTPUT :



RESULT :

Thus, to simulate resource management in a cloud environment using CloudSim by allocating and utilizing virtual machines and cloudlets efficiently executed successfully.

EX:NO: 3	Simulate log forensics using cloud sim
DATE:	

AIM:

To simulate log forensics in a cloud environment by analyzing log data to detect suspicious activities and anomalies.

PROCEDURE :

1. Install Java Development Kit (JDK) on the system.
2. Download CloudSim (version 3.0.3 or later) and keep all JAR files in one folder.
3. Create a Java file named LogForensicsSimulation.java and save the program.
4. Open Command Prompt and navigate to the CloudSim JAR directory.
5. Import required CloudSim and Java packages.
6. Initialize the CloudSim environment using CloudSim.init().
7. Generate sample log data representing cloud activities.
8. Analyze logs to detect suspicious activities and anomalies.
9. Compile the program using the command:

   javac -cp "*" LogForensicsSimulation.java
10. Run the program using the command:

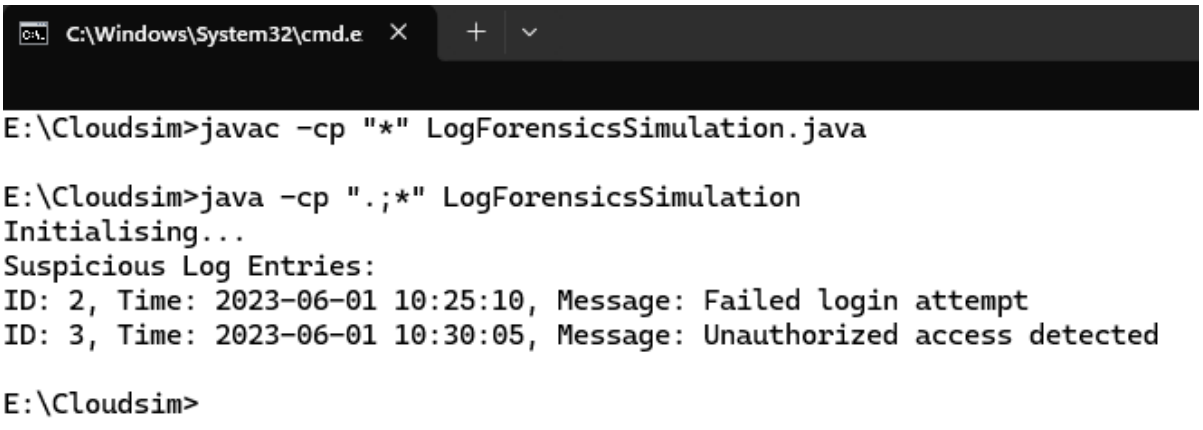
    java -cp ".;*" LogForensicsSimulation

PROGRAM :

```
import org.cloudbus.cloudsim.core.CloudSim;
import java.util.*;

class LogEntry {
   int id;
   String time;
   String message;

   LogEntry(int id, String time, String message) {
      this.id = id;
      this.time = time;
```

```
        this.message = message;
    }
}

public class LogForensicsSimulation {
    public static void main(String[] args) {
        CloudSim.init(1, Calendar.getInstance(), false);

        List<LogEntry> logs = new ArrayList<>();
        logs.add(new LogEntry(1, "2023-06-01 10:23:45", "User login uccessful"));
        logs.add(new LogEntry(2, "2023-06-01 10:25:10", "Failed login attempt"));
        logs.add(new LogEntry(3, "2023-06-01 10:30:05", "Unauthorized access detected"));
        logs.add(new LogEntry(4, "2023-06-01 10:35:20", "Normal file access"));

        System.out.println("Suspicious Log Entries:");
        for (LogEntry log : logs) {
            if (log.message.contains("Failed")
                    || log.message.contains("Unauthorized")) {
                System.out.println(
                        "ID: " + log.id +
                        ", Time: " + log.time +
                        ", Message: " + log.message
                );
            }
        }
    }
}
```

OUTPUT :



RESULT :

Thus, to simulate log forensics in a cloud environment by analyzing log data to detect suspicious activities and anomalies is executed successfully.

EX:NO: 4	Simulate a secure file sharing using a cloud sim
DATE:	

AIM :

To simulate a secure file sharing system in a cloud environment by performing secure upload and download operations.

PROCEDURE:

1. Install Java Development Kit (JDK) on the system.
2. Download CloudSim (version 3.0.3 or later) and keep all JAR files in one folder.
3. Create a Java file named SecureFileSharingSimulation.java.
4. Open Command Prompt and navigate to the CloudSim folder.
5. Import required CloudSim and Java packages.
6. Initialize CloudSim using CloudSim.init().
7. Simulate secure file upload by encrypting file data.
8. Simulate secure file download by decrypting file data.
9. Compile the program using:

   javac -cp "*" SecureFileSharingSimulation.java
10. Run the program using:

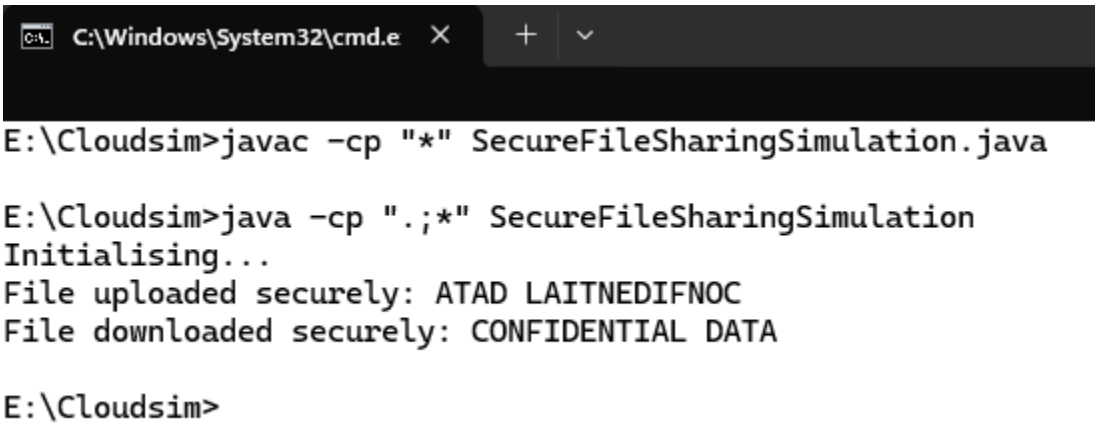
    java -cp ".;*" SecureFileSharingSimulation

PROGRAM :

```
import org.cloudbus.cloudsim.core.CloudSim;
import java.util.*;

public class SecureFileSharingSimulation {
   public static void main(String[] args) {
      long startTime = System.currentTimeMillis();
      CloudSim.init(1, Calendar.getInstance(), false);

      String fileData = "CONFIDENTIAL DATA";
      String encryptedData = encrypt(fileData);
      System.out.println("File uploaded securely: " + encryptedData);
```

```
        String decryptedData = decrypt(encryptedData);
        System.out.println("File downloaded securely: " + decryptedData);

        long endTime = System.currentTimeMillis();
        double totalTime = (endTime - startTime) / 1000.0;
        System.out.println("Total simulation time: " + totalTime + " seconds");
    }
    static String encrypt(String data) {
        return new StringBuilder(data).reverse().toString();
    }
    static String decrypt(String data) {
        return new StringBuilder(data).reverse().toString();
    }
}
```

OUTPUT :



RESULT :

Thus, to simulate a secure file sharing system in a cloud environment by performing secure upload and download operations is executed completely.

EX:NO: 5	Implement data anonymization techniques over the simple dataset (masking, kanonymization, etc)
DATE:	

AIM:

To implement data anonymization techniques such as data masking and k-anonymization on a simple dataset using Python.

PROCEDURE :

1. Install Python on the system.
2. Open Python IDLE.
3. Create a new file and save it as data_anonymization.py.
4. Define a simple dataset containing personal information.
5. Apply data masking to hide sensitive attributes.
6. Apply k-anonymization by generalizing the data.
7. Execute the program using Python IDLE.
8. Observe the anonymized dataset.
9. Verify that sensitive information is protected.
10. Display the final anonymized output.

PROGRAM :

1) Data Masking

```
import pandas as pd

data = pd.DataFrame({
   'Name': ['John Doe', 'Jane Smith', 'Johnson'],
   'Email': ['johndoe@example.com', 'janesmith@example.com', 'johnson@example.com' ],
   'Age': [25, 30, 35]
})
data['Name'] = 'XXXXXXXXXX'
data['Email'] = 'xxxxxxxxxx'
print(data)
```

OUTPUT :

```
        Name           Email         Age
0  XXXXXXXXXX  xxxxxxxxxx   25
1  XXXXXXXXXX  xxxxxxxxxx   30
2  XXXXXXXXXX  xxxxxxxxxx   35
```

2) K-Anonymization

```
import pandas as pd

data = pd.DataFrame({
   'Name': ['John Doe', 'Jane Smith', 'Michael Johnson'],
   'Zip Code': ['12345', '67890', '54321'],
   'Age': [25, 30, 35]
})

data['Name'] = 'Anonymous'
data['Zip Code'] = 'XXXXX'
print(data)
```

OUTPUT :

```
      Name        Zip Code  Age
0  Anonymous    XXXXX   25
1  Anonymous    XXXXX   30
```

RESULT :

Thus, to implement data anonymization techniques such as data masking and k-anonymization on a simple dataset using Python is executed successfully.

EX:NO: 6	Implement any encryption algorithm to protect the images
DATE:	

AIM :

To encrypt an image file using the AES (Advanced Encryption Standard) algorithm in order to protect its contents from unauthorized access.

PROCEDURE:

1. Install Python on the system.
2. Open Python IDLE.
3. Install required cryptography library using pip install pycryptodome.
4. Create a new file and save it as image_encryption.py.
5. Choose AES as the encryption algorithm.
6. Generate a secret encryption key.
7. Read the image file in binary mode.
8. Encrypt the image data using AES encryption.
9. Save the encrypted image to a new file.
10. Execute the program and observe the encrypted output.

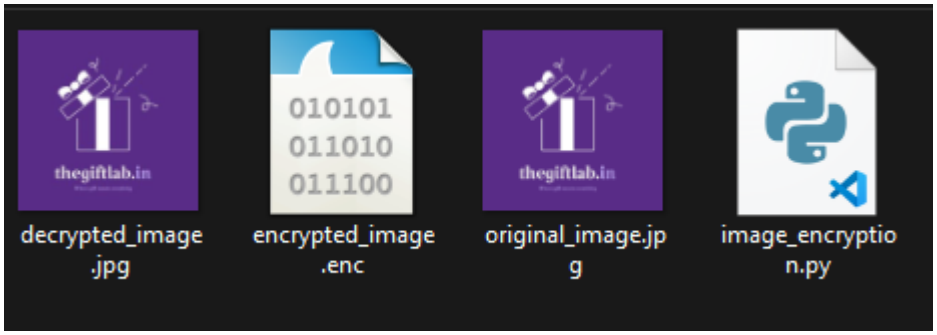
PROGRAM:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import os

key = b'ThisIsASecretKey'
with open("original_image.jpg", "rb") as file:
    image_data = file.read()

iv = os.urandom(16)
cipher = AES.new(key, AES.MODE_CBC, iv)
encrypted_data = cipher.encrypt(pad(image_data, AES.block_size))
```

```
with open("encrypted_image.enc", "wb") as file:
    file.write(iv + encrypted_data)
print("Image encrypted successfully")

cipher = AES.new(key, AES.MODE_CBC, iv)
decrypted_data = unpad(cipher.decrypt(encrypted_data), AES.block_size)
with open("decrypted_image.jpg", "wb") as file:
    file.write(decrypted_data)
print("Image decrypted successfully")
```

OUTPUT :

```
Image encrypted successfully
Image decrypted successfully
```



RESULT :

Thus, to encrypt an image file using the AES (Advanced Encryption Standard) algorithm in order to protect its contents from unauthorized access is executed successfully.

EX:NO: 7	Implement any image obfuscation mechanism
DATE:	

.

AIM:

To obfuscate an image by applying a blurring technique so that sensitive visual information is not clearly recognizable.

PROCEDURE :

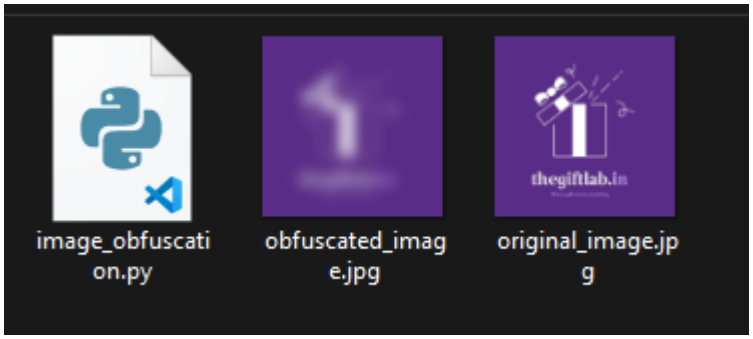
1. Install Python on the system.
2. Open Python IDLE.
3. Install the required library using pip install pillow.
4. Create a new file and save it as image_obfuscation.py.
5. Select an image file for obfuscation.
6. Load the image using Python.
7. Apply a blurring filter to the image.
8. Save the obfuscated image to a new file.
9. Execute the program using Python IDLE.
10. Observe the obfuscated output image.

PROGRAM:

```
from PIL import Image, ImageFilter

# Open the original image
image = Image.open("original_image.jpg")

# Apply blur to obfuscate the image
blurred_image = image.filter(ImageFilter.GaussianBlur(radius=40))

# Save the obfuscated image
blurred_image.save("obfuscated_image.jpg")

print("Image obfuscated successfully")
```

OUTPUT:

Image obfuscated successfully



RESULT:

Thus, to obfuscate an image by applying a blurring technique so that sensitive visual information is not clearly recognizable is executed successfully.

EX:NO: 8	Implement a role-based access control mechanism in a specific scenario
DATE:	

AIM:

To implement a Role-Based Access Control (RBAC) mechanism in a specific scenario where users are granted or denied access to resources based on their assigned roles.

PROCEDURE :

1. Install Python on the system.
2. Open Python IDLE.
3. Create a new file and save it as rbac.py.
4. Define different roles such as admin, developer, and end user.
5. Assign permissions to each role.
6. Assign roles to users using email IDs.
7. Accept the user's email as input.
8. Accept the permission to be checked as input.
9. Verify access based on the user's role.
10. Display whether access is granted or denied.

PROGRAM :

```
# Define roles and permissions
roles = {
   "admin": ["read", "write", "delete"],
   "developer": ["read", "write"],
   "end_user": ["read"]
}

# Assign roles to users
user_roles = {
   "admin@cloud.com": "admin",
   "dev@cloud.com": "developer",
   "user@cloud.com": "end_user"
}

# Get user input
email = input("Enter your email: ")
permission = input("Enter permission to check (read/write/delete): ")
```

```
# RBAC access check
if email in user_roles:
    role = user_roles[email]
    if permission in roles[role]:
        print("Access Granted")
    else:
        print("Access Denied")
else:
    print("User not found")
```

OUTPUT:

```
Enter your email: admin@cloud.com
Enter permission to check (read/write/delete): delete
Access Granted

Enter your email: user@cloud.com
Enter permission to check (read/write/delete): delete
Access Denied
```

RESULT :

Thus, to implement a Role-Based Access Control (RBAC) mechanism in a specific scenario where users are granted or denied access to resources based on their assigned roles is executed successfully.

EX:NO: 9	Implement an attribute-based access control mechanism based on a particular scenario
DATE:	

AIM:

To implement an Attribute-Based Access Control (ABAC) mechanism in a specific scenario where access decisions are made based on user attributes such as role and department.

PROCEDURE:

1. Install Python on the system.
2. Open Python IDLE.
3. Create a new file and save it as abac.py.
4. Define user attributes such as role and department.
5. Define access control policies based on attributes.
6. Accept user ID as input.
7. Retrieve attribute values for the user.
8. Check access based on attributes and policies.
9. Display whether access is allowed or denied.
10. Execute the program and observe the output.

PROGRAM :

```
# Attribute Authority (stores user attributes)
user_attributes = {
   "user1": {"role": "Manager", "department": "Sales"},
   "user2": {"role": "Employee", "department": "Sales"},
   "user3": {"role": "Admin", "department": "IT"}
}

# Access control policies
policies = [
   {"resource": "sales_data", "action": "read",
    "role": "Manager", "department": "Sales"},
   {"resource": "admin_panel", "action": "write",
    "role": "Admin"}
]

# Get user input
user_id = input("Enter user ID: ")
resource = input("Enter resource name: ")
action = input("Enter action (read/write): ")
```

```
# ABAC access check
access_granted = False
if user_id in user_attributes:
    user_role = user_attributes[user_id]["role"]
    user_dept = user_attributes[user_id]["department"]
    for policy in policies:
        if policy["resource"] == resource and policy["action"] == action:
            if (policy.get("role") == user_role and
                policy.get("department", user_dept) == user_dept):
                access_granted = True
                break

# Output result
if access_granted:
    print("Access Granted")
else:
    print("Access Denied")
```

OUTPUT:

```
Enter user ID: user1
Enter resource name: sales_data
Enter action (read/write): read
Access Granted

Enter user ID: user2
Enter resource name: sales_data
Enter action (read/write): read
Access Denied
```

RESULT:

Thus, to implement an Attribute-Based Access Control (ABAC) mechanism in a specific scenario where access decisions are made based on user attributes such as role and department is executed successfully.

EX:NO: 10	Develop a log monitoring system with incident management in the cloud
DATE:	

AIM :

To develop a log monitoring system that detects anomalies or predefined patterns in log data and generates incidents for further investigation.

PROCEDURE:

1. Install Python on the system.
2. Open Python IDLE.
3. Create a new file and save it as log_monitoring.py.
4. Define sample log data representing cloud logs.
5. Define suspicious patterns such as ERROR, FAILED, or UNAUTHORIZED.
6. Monitor logs to detect predefined patterns.
7. Generate an incident when a suspicious log is detected.
8. Display incident details such as timestamp and message.
9. Execute the program using Python IDLE.
10. Observe incident generation output.

PROGRAM:

```
# Sample cloud log data
logs = [
    "2024-01-10 10:20:30 INFO User login successful",
    "2024-01-10 10:22:15 ERROR Database connection failed",
    "2024-01-10 10:25:40 INFO File uploaded",
    "2024-01-10 10:30:05 UNAUTHORIZED access attempt detected"
]

# Incident ID counter
incident_id = 1

print("Log Monitoring Started...\n")
```

```
# Monitor logs and generate incidents
for log in logs:
    if "ERROR" in log or "FAILED" in log or "UNAUTHORIZED" in log:
        print("Incident Generated")
        print("Incident ID:", incident_id)
        print("Log Details:", log)
        print("-" * 40)
        incident_id += 1
```

.

OUTPUT: :

```
Log Monitoring Started...

Incident Generated
Incident ID: 1
Log Details: 2024-01-10 10:22:15 ERROR Database connection failed
----------------------------------------
Incident Generated
Incident ID: 2
Log Details: 2024-01-10 10:30:05 UNAUTHORIZED access attempt detected
----------------------------------------
```

RESULT :

Thus, to develop a log monitoring system that detects anomalies or predefined patterns in log data and generates incidents for further investigation is executed successfully.