National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Harsha Venkata Naga Vardhan Neerukonda
Student ID: 22173536

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Harsha Venkata Naga Vardhan Neerukonda
Student ID:	22173536
Programme:	Cloud Computing **Year:** Jan 2023
Module:	MSc Research Project
Lecturer:	Vikas Sahni
Submission Due Date:	14/12/2023
Project Title:	A Comprehensive Framework for Ensuring Secure Sharing of Electronic Health Records in AWS Environment
Word Count:	826 **Page Count:** 13

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: N.H.V.N.Vardhan

Date: 14/12/2023

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

Harsha Venkata Naga Vardhan Neerukonda
Student ID: 22173536

1 Introduction

This document serves as a detailed configuration manual for the replication of the work "A Comprehensive Framework for Ensuring Secure Sharing of Electronic Health Records in AWS Environment." This work involves a multiple components and configurations, and the following sections provide a step-by-step guide for successfully reproducing this work.

Research Work: A comprehensive framework for ensuring the secure sharing of electronic health records (EHR) in an AWS environment. The primary objective is to enhance the security of EHR sharing through the utilization of the Ciphertext-Policy Attribute Based Encryption (CPABE) mechanism. Additionally, a deduplication technique has been designed to efficiently optimize storage costs on Amazon S3.

2 System Configuration

2.1 Hardware Configuration

Below Fig1 specifies the hardware configuration that are required to develop and run the application.

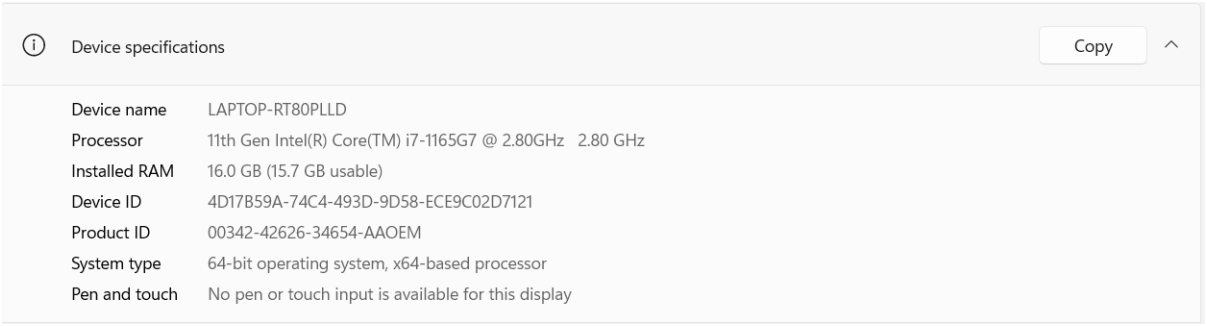

Fig1. Hardware Requirements

2.2 Software Configuration

Below are the list of software's and the AWS services that are used.

IDE: Eclipse IDE.
Server: Apache Tomcat.
Database: MySQL
Amazon IAM: To regulate access permissions to the S3 buckets within the development.
Amazon S3: For storing EHR reports securely.

3 Environment Setup

3.1 Installing and Configuring Eclipse with Tomcat

- Download and install Eclipse IDE.
- Set up Apache Tomcat as the server for deploying the web-based application.
- Configure Eclipse to work with Tomcat.
- The following figures from Fig2. To Fig7. shows the process of installing the Eclipse and configuring the Eclipse with Tomcat Server.

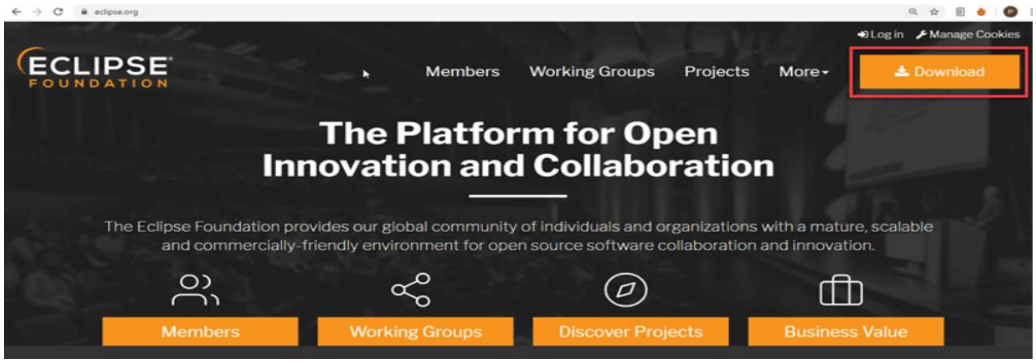

Fig2. Eclipse Page

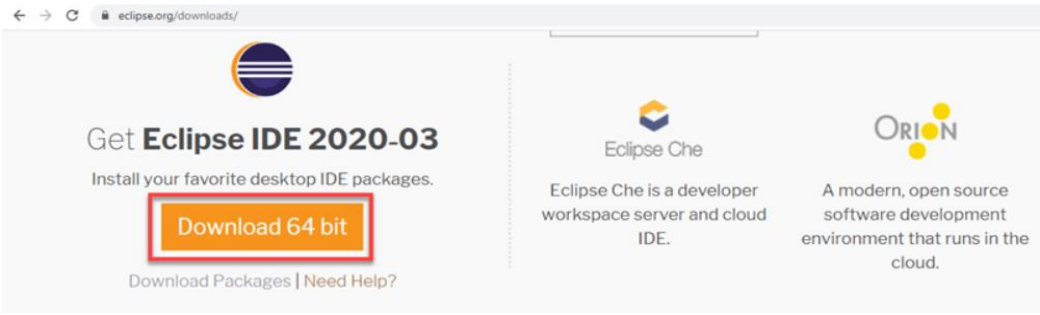

Fig3. Eclipse Download

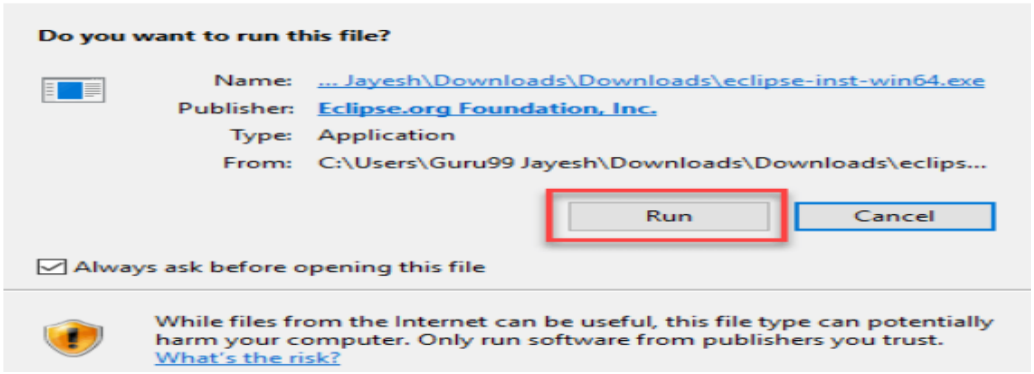

Fig4. Run Eclipse

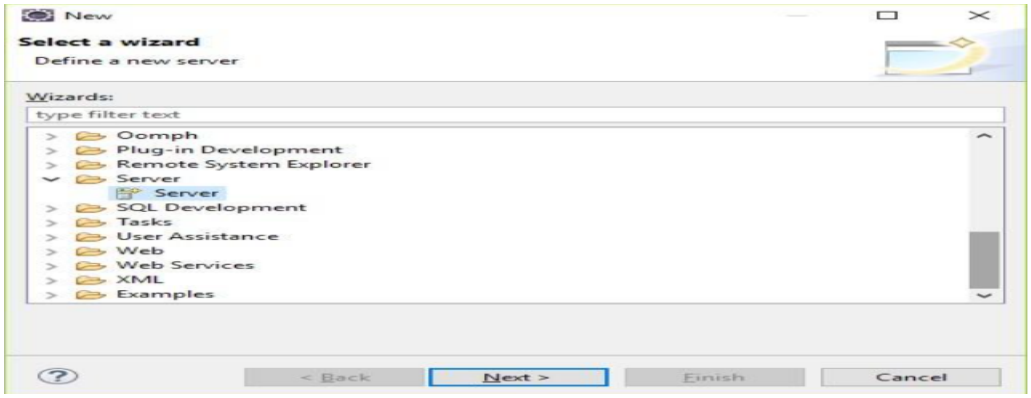

Fig5. Integrating Tomcat Server

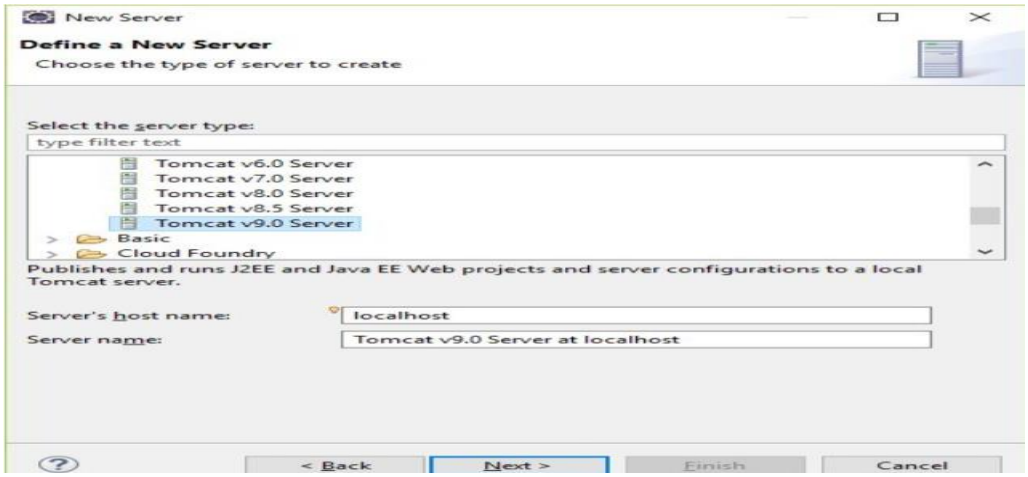

Fig6. Tomcat Version

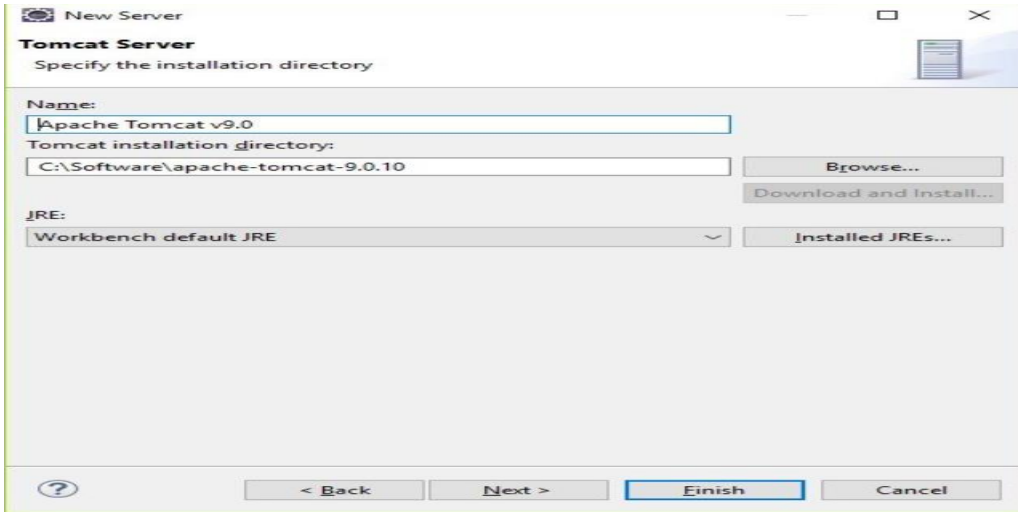

Fig7. Tomcat Server

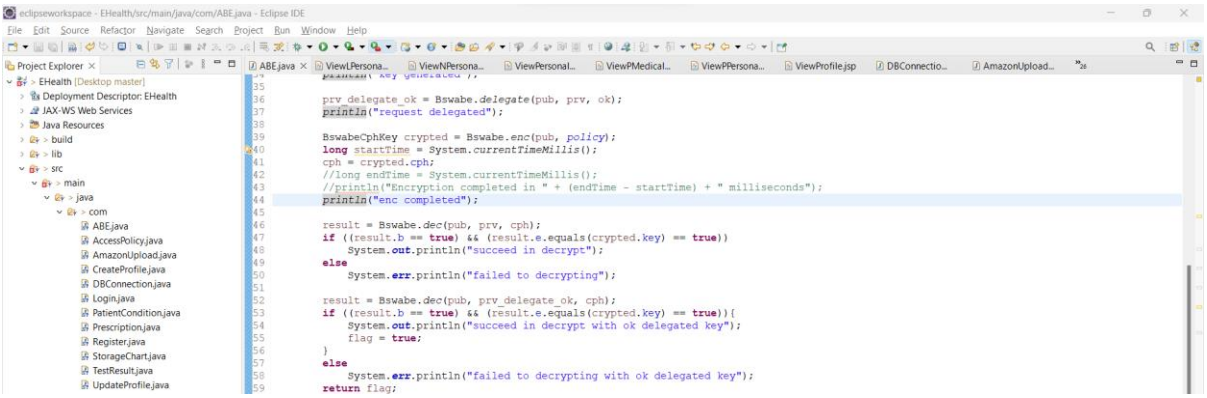

Fig 8. Import the Code artifact into the Eclipse.

3.2 MySQL Database:

- Visit the MySQL Community Downloads page.
- Download the MySQL Installer for Windows.
- Run the downloaded installer, and it will guide you through the installation process.
- Choose the setup type (Typical, Complete, Custom).
- During installation, you'll be prompted to configure MySQL Server.
- Set a root password and choose other configuration options as needed.
- Once installed, MySQL Server should start automatically.
- You can also start it manually using the MySQL Notifier or by going to the Services application and starting the MySQL service.
- Open a command prompt and type mysql -u root -p. Enter the root password when prompted.
- You should now be in the MySQL shell.
- The following figures from Fig9. to Fig15. shows the installation and configuration of MYSQL.

Fig9. MySQL Page

Fig10. MySQL Windows Installer

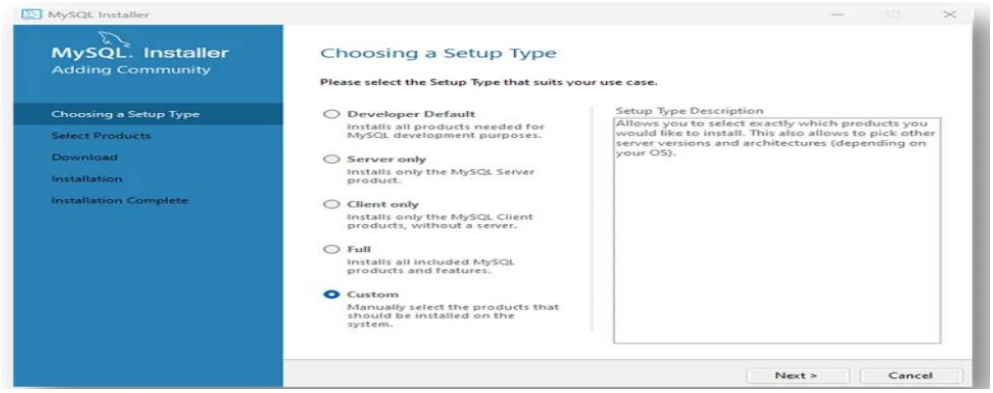

Fig11. MySQL Setup

Fig12. MySQL Products

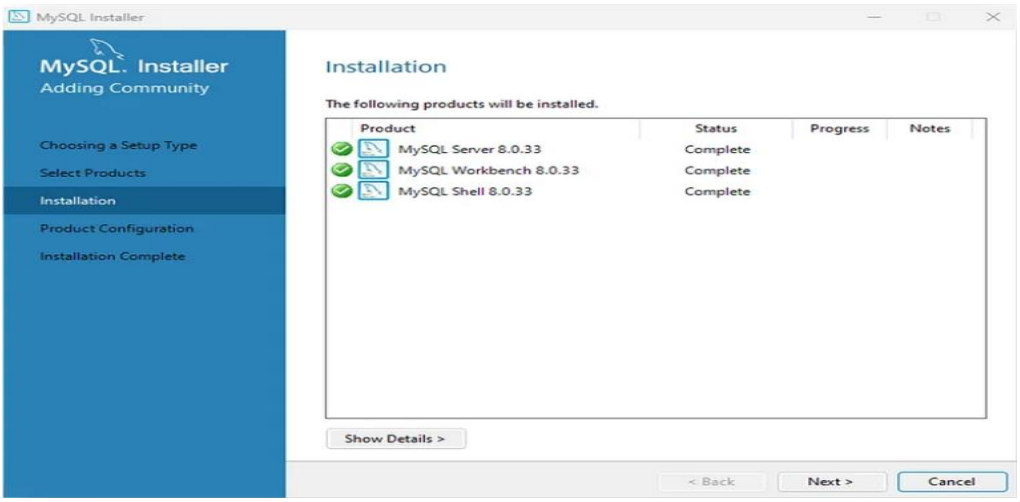

Fig13. Installation

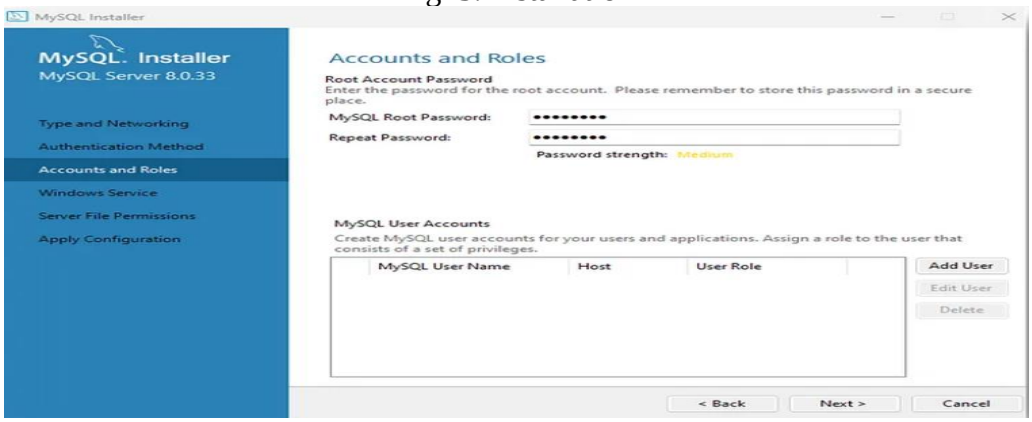

Fig14. mySQL Account

Fig15. MySQL Workbench

- Below Fig16. shows the commands that needed to be executed in the MySQL database in order to create the database and the tables.

```
create database ehealth;
use ehealth;
create table newuser(username varchar(50),password varchar(50),usertype varchar(50),contactno varchar(12),address varchar(50));

create table Createprofile(name varchar(50),dob date,age varchar(50),gender varchar(10),SSno varchar(50),profile_id int,username varchar(50),problem varchar(50));

create table accesspolicy(friend varchar(50),faccess varchar(50),physician varchar(50),paccess varchar(50),nurse varchar(50),naccess varchar(50),lab varchar(50),laccess varchar(50),insurance
varchar(50),iaccess varchar(50),username varchar(50));

create table prescription(pname varchar(50),physician varchar(50),prescription varchar(250),pre_date date);


create table patientcondition(pname varchar(50),nurse varchar(50),pcondition varchar(250),pre_date date,bp double,rr double,hr double,pr double);

create table testResult(pname varchar(50),labtechnician varchar(50),testname varchar(100),results varchar(250),report_name varchar(100),report longblob,pre_date date);
```

Fig16. Commands

The below Fig17. shows the DB endpoint configured in the code.

```
public static Connection getCon()throws Exception {
    try{
        Class.forName("com.mysql.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/ehealth","root","root");
    }catch(Exception e){
        e.printStackTrace();
    }
    return con;
}
```

Fig17. DB EndPoint

4 AWS IAM Creation:

- Log in to the AWS Management Console using https://aws.amazon.com/.
- Navigate to the IAM service.
- In the IAM dashboard, select "Users" from the left-hand menu.
- Click on the "Create user" button.
- Enter a username for the new IAM user.
- On the "Set permissions" page, choose "Attach policies directly."
- Search for and attach the policy named "AmazonS3FullAccess" to grant full access to S3.
- Review the configuration and click "Next".
- Review the user details and permissions one last time.
- Click "Create user."
- Once the user is created, go to security credentials and click on create Access key.
- Next, click "Download .csv" to download the user's access key ID and secret access key.
- Configure the access keys for the IAM user in the code.
- Below Figures from Fig18. to Fig 24. shows the step by step process of creating IAM user.

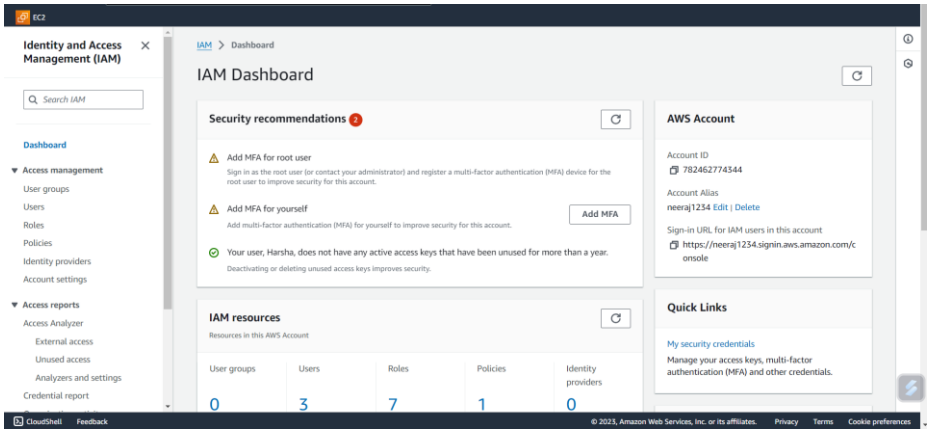

Fig18. IAM Dashboard

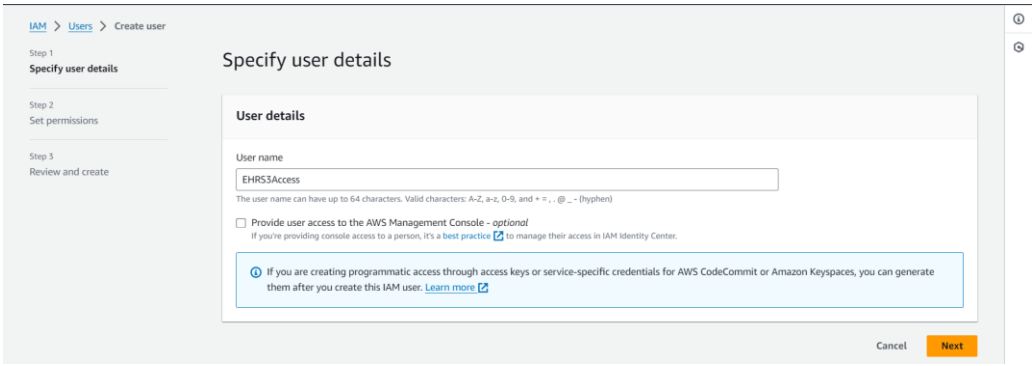

Fig19. User Details

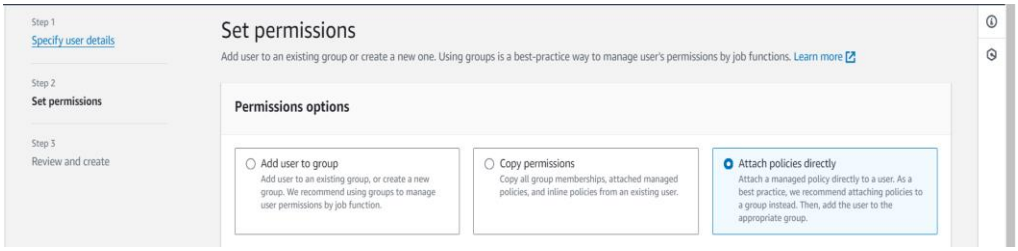

Fig20. Permissions

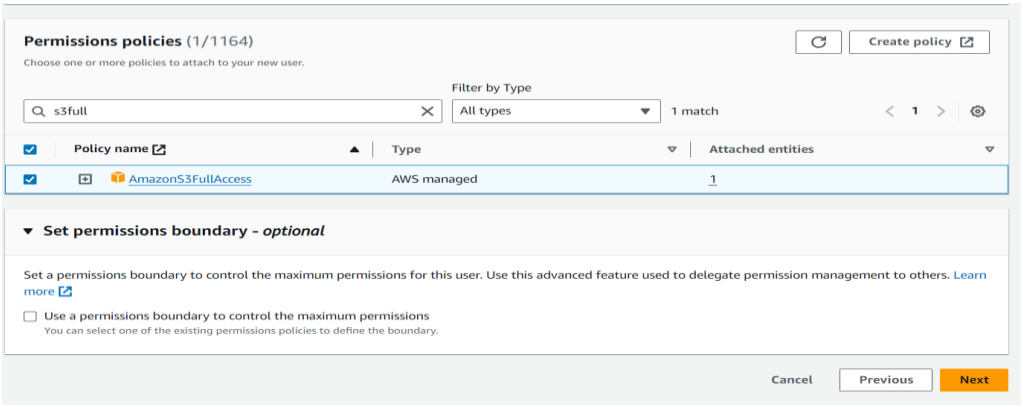

Fig21. Policies

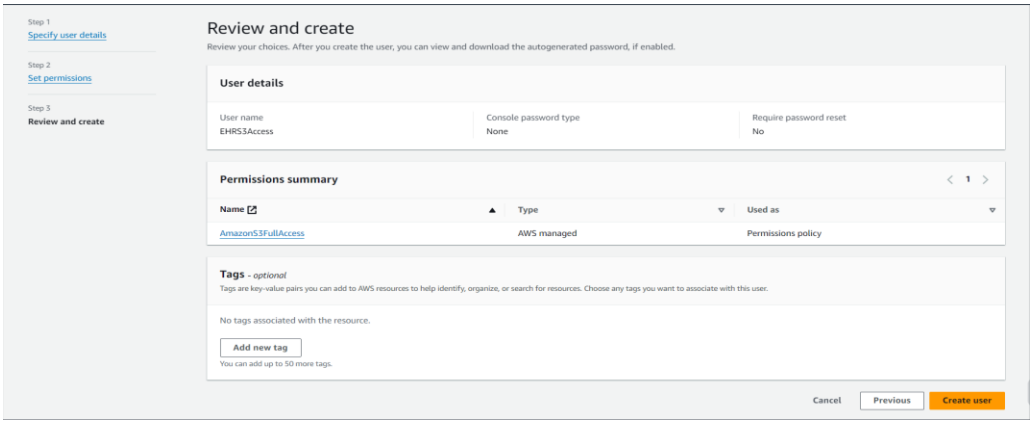

Fig22. Create User

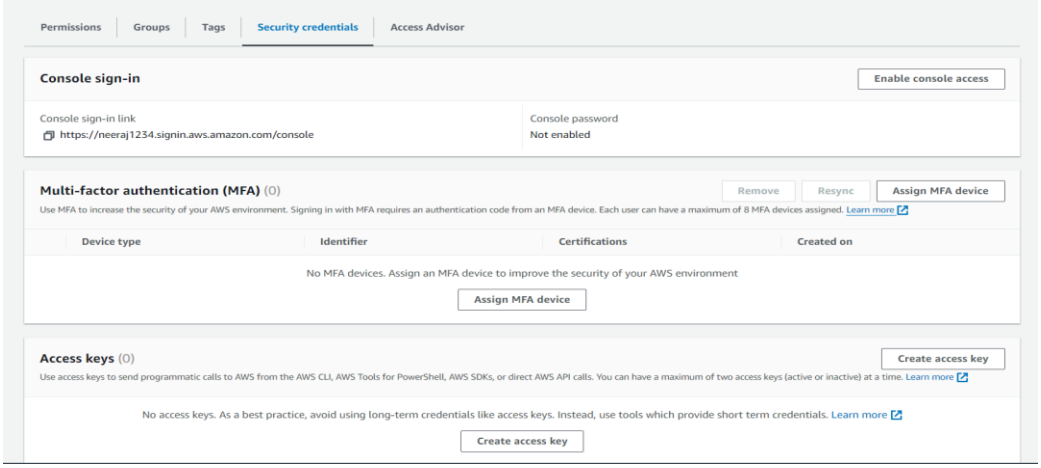

Fig23. Access key

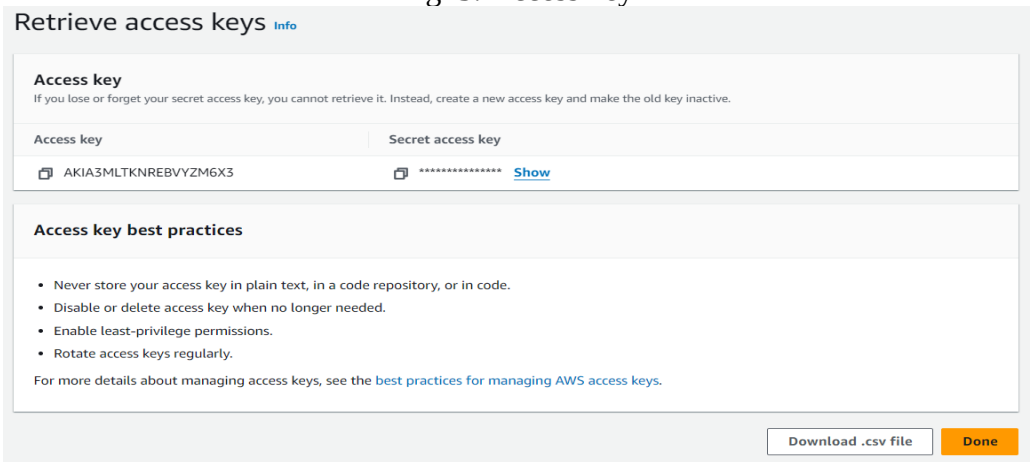

Fig24. Create Access key

- Below Fig25. shows replacing the keys with the IAM keys in the code.

```
public void upload() throws Exception {
    System.out.println(user.toUpperCase());
    BasicAWSCredentials credentials = new BasicAWSCredentials("AKIA3MLTKNREAJG4KBQD","I0U8eIgVrKRqIFkO2TxP
    AmazonS3 s3Client = new AmazonS3Client(credentials);
    if(!s3Client.doesBucketExist("ehr-"+user)){
        Bucket bucket = s3Client.createBucket("ehr-"+user,"us-east-2");
    }
    s3Client.putObject("ehr-"+user,file.getName(),file);
    msg = "success";
}
```

Fig25. Access Key usage in the code

5 Execution:

- Right click on the code artifact in the eclipse and click on Run server.
- Select Tomcat server and click on Finish.
- The following figures from Fig 26. to Fig28. shows the deployment of the application.

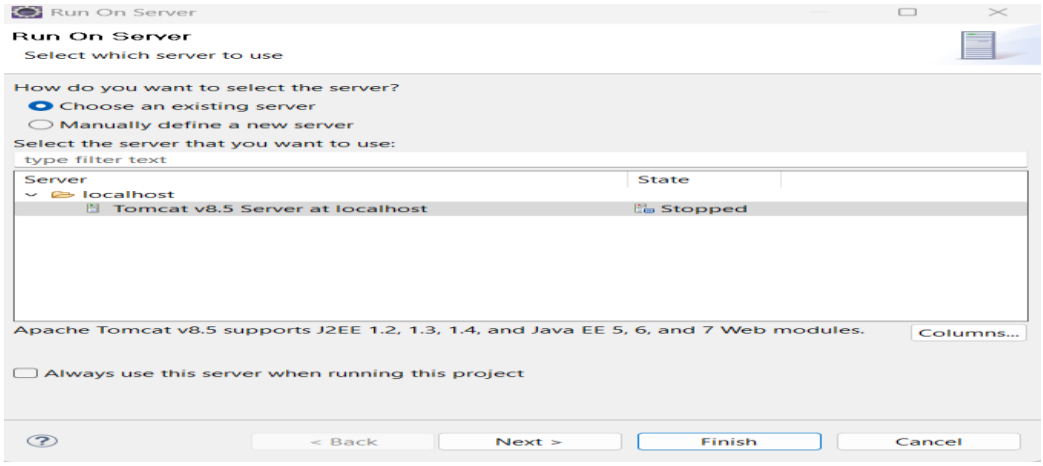

Fig26. Tomcat Server

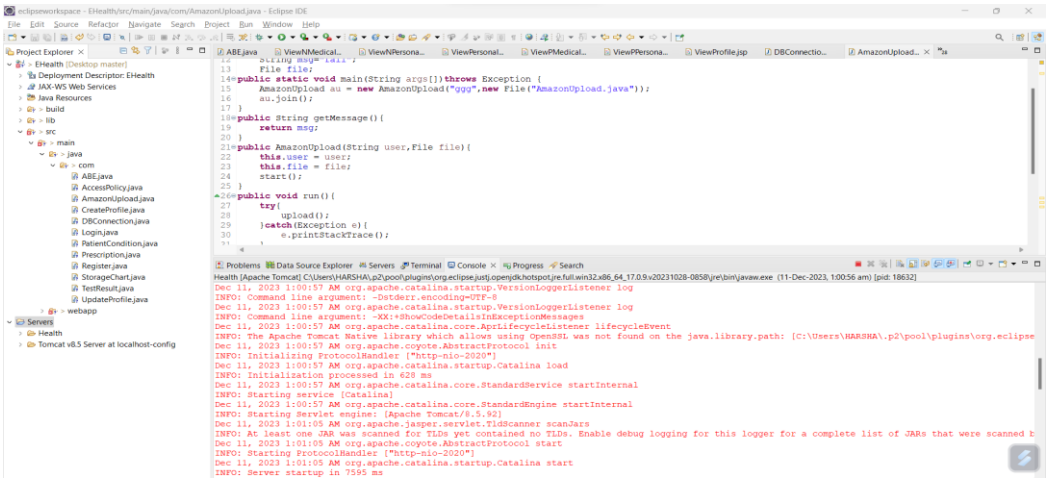

Fig27. Application Deployment

Fig28. Application

Fig29. Report Upload

From the above Fig29. shows that when the user uploads the report it automatically creates the S3 bucket in the AWS and uploads the report to the S3 bucket as shown in the below Fig30. and Fig31.

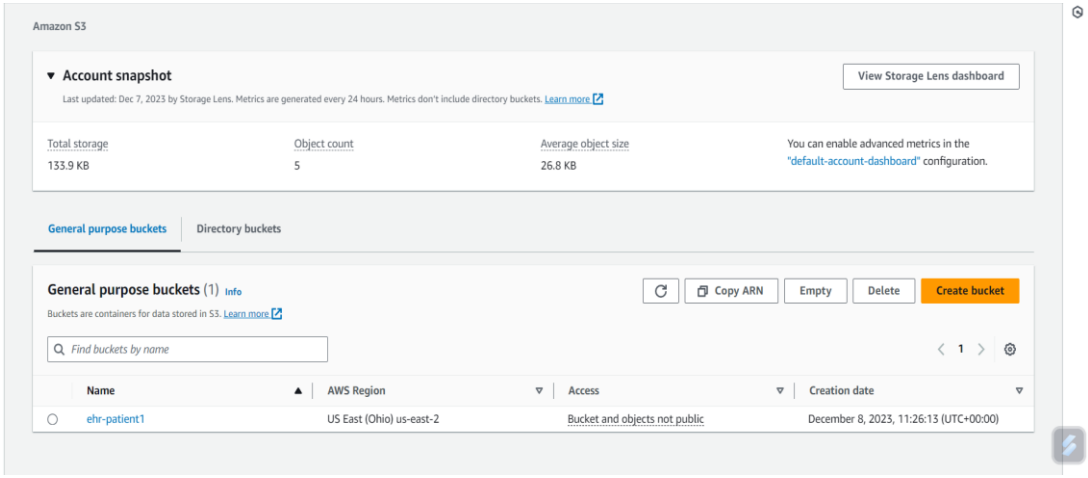

Fig30. S3 Bucket

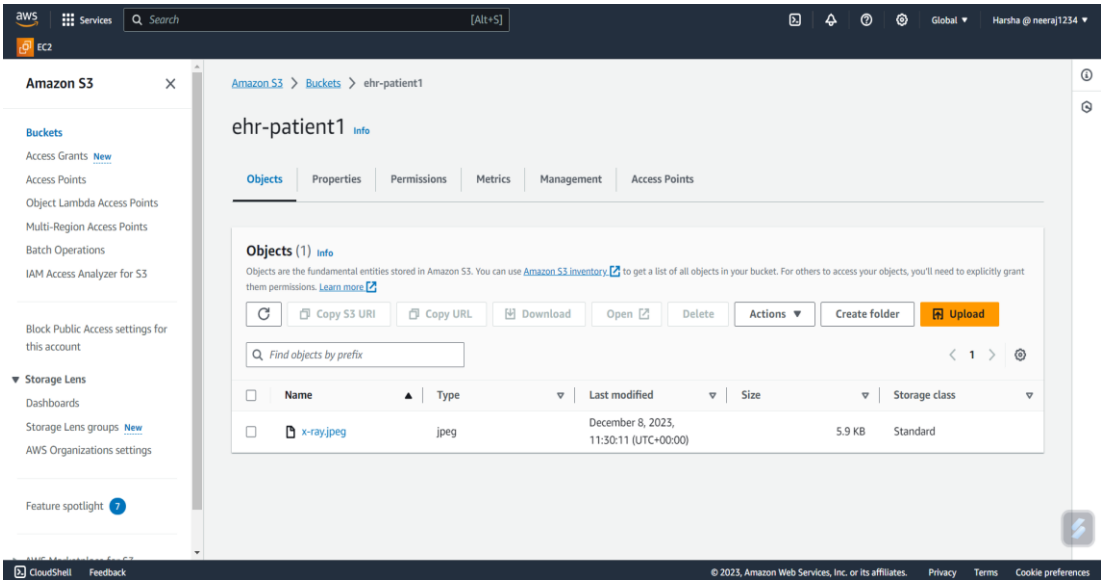

Fig31. S3 Bucket Objects.

6 Evaluation

Below Fig32 shows the importing libraries for the Fernet (Symmetric) Encryption in order to compare the performance with the CPABE.

```
import boto3
from cryptography.fernet import Fernet
import zlib
import time
```

Fig32. Importing libraries for Fernet

Below Fig33. shows the logic for the time taken for the fernet to encrypt/decrypt and upload the compressed data into S3 bucket.

```
# Measure encryption time
start_time_encryption = time.time()
encrypted_data = cipher_suite.encrypt(plaintext_data)
encryption_time = time.time() - start_time_encryption

# Measure compression size
compressed_data = zlib.compress(encrypted_data)
compression_ratio = len(compressed_data) / len(encrypted_data)  # Compression ratio
compression_size = len(compressed_data)

# Upload compressed data to S3
s3_object_key = 'compressed_encrypted_patient_data.txt'  # Replace with your desired object key
start_time_upload = time.time()
s3_client.put_object(
    Bucket=s3_bucket_name,
    Key=s3_object_key,
    Body=compressed_data
)
upload_time = time.time() - start_time_upload
```

Fig33. Data Encryption/Decryption for Fernet

Below Fig34. shows the logic for the time taken for the fernet to encrypt/decrypt and upload the compressed data.

```
# Encrypt the message
start_time = time.time()
policy = ['Doctor', 'BloodType=A+']
ciphertext = cpabe.encrypt(patient_data, policy)
encryption_time = time.time() - start_time

# Simulate upload time
upload_time = time.time() - start_time

# Calculate compression ratio
original_data_size = len(str(patient_data))
encrypted_data_size = len(str(ciphertext))
compression_ratio = ((original_data_size - encrypted_data_size) / original_data_size)
compression_size = (original_data_size) - (encrypted_data_size)


# Alice decrypts
start_time = time.time()
decrypted_message = cpabe.decrypt(ciphertext, alice_key)
decryption_time = time.time() - start_time
```

Fig34. Data Encryption/Decryption for CPABE

Below Fig35. shows

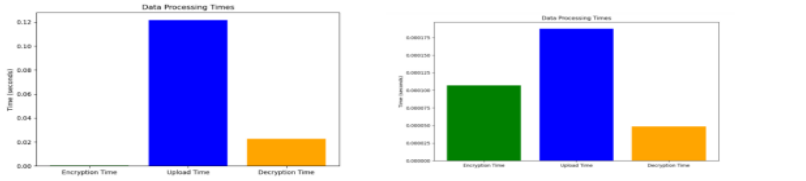

Figure 19. Comparison of encryption, decryption and upload times of Fernet (left) and CPABE (right).

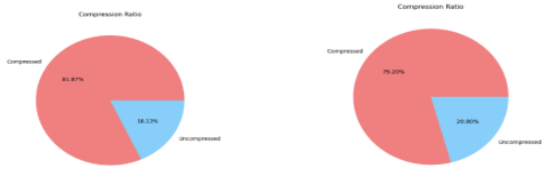

Fig35. Results

References

Alabdulatif, A., Khalil, I. and Mai, V., 2013, July. Protection of electronic health records (EHRs) in cloud. In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 4191-4194). IEEE.

Pandit, P., 2021. Case Study on AWS Identity and User Management.