A Project Report on

Enhancing Data Security in Cloud using Blockchain

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Information Technology

by

Dhananjay Yadav(17204015)
Aditi Shinde(16104022)
Akash Nair(16104051)

Under the Guidance of

Prof. Yamini Patil
Prof. Sneha Kanchan

Department of Information Technology
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic Year 2019-2020

Approval Sheet

This Project Report entitled ***"Enhancing Data Security in Cloud using Blockchain"*** Submitted by ***"Dhananjay Yadav"(17204015), "Aditi Shinde"(16104022), "Akash Nair"(16104051)*** is approved for the partial fulfillment of the requirenment for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

(Name)
Prof. Sneha Kanchan

(Name)
Prof. Yamini Patil

Prof. Kiran Deshpande
Head Department of Information Technology

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***"Enhancing Data Security in Cloud using Blockchain"*** submitted by ***"Dhananjay Yadav" (17204015),"Aditi Shinde" (16104022),"Akash Nair" (16104051)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Information Technology***, to the University of Mumbai,is a bonafide work carried out during academic year 2019-2020.

(Name)	(Name)
Prof. Sneha Kanchan	Prof. Yamini Patil

Prof. Kiran Deshpande	Dr. Uttam D.Kolekar
Head Department of Information Technology	Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane

Date:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

(Dhananjay Yadav 17204015)
(Aditi Shinde 16104022)
(Akash Nair 16104051)

Date:

Abstract

Blockchain is a distributed network mechanism invented to secure data in more advance method. Daily lots of data is exchanged and loaded on cloud into different sectors one of which is health sector. Data exchanged between the patient and doctors need to be secured to gain patients trust. Blockchain store data into chunks that make it hard to decode, which will help provide extra layer of security. Hash chain is the most reliable part of blockchain that will help keep the data unreadable. This data can be secured by using an blockchain mechanism at the backend of any hospital website to store the reports of the patients, and maintain an two-way authentication for doctors access to the reports.

Contents

List of Figures

List of Tables

List of Abbreviations

AWS	:	Amazon Web Server
AES	:	Advanced Encryption Standard
OTP	:	One Time Password

Chapter 1

Introduction

In current world, the biggest concern for every major sector is about the amount of security they can achieve onto their data. Almost everyone uses cloud as their storage and demands for great level of security, and what's better than blockchain when it comes to security. Blockchain, is a growing list of records, called blocks, that are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. Blockchain has emerged in many different sectors from bitcoins. Any sector can be considered here, whereas the major focus in this project is about the healthcare data. Hospitals generate a large amount of confidential data and especially every healthcare center needs to maintain HIPAA rules. Here we are using blockchain as a security mechanism to the data that's been uploaded by the patients regarding their diseases and that data is then secured through blockchain mechanism. This security mechanism will be the backend processing of any hospital website which will be hosted on cloud where Doctors and Patients can communicate with each other and share reports.

1.1 Blockchain

- At first the data would be stored with the help of certain algorithm.

- After which the data would be divided into chunks and hash values would be generated

- These chunks with there hash values will be interconnected with the concept of blockchain.

1.2 User Interface

- Admin side to manage the whole website for granting and revoking privileges to the other entities of website.

- Doctors Portal that would deal with dealing with patients like adding new patients, prescribing medicines, requesting to view report

- Patients Portal that would be able to upload report, view there own report, grant access to authorized doctors for there report.

1.3 Storage

- Some minor fields like storing user credentials for authentication, storing prescribed medicines.

- Storing the reports into chunks in an cloud enviornment using S3 storage.

Chapter 2

Literature Review

[1] NakamotoS.Bitcoin: A peer-to-peer electronic cash system[J]. Consulted,2008.

Blockchain originally refers to the increasing list of records or data in a chain like manner. The blockchain was found in 1992 as a technology to cease the tampering of timestamp on files. The latest technology blockchain can be traced back to 2008 when Satoshi Nakamoto (along with his colleagues) revised blockchain into a ledger like continuous transaction list which later led to the founding of the first crypto currency called bitcoin. Bitcoin was the first currency to have blockchain backing it for its secure and tamper less transaction which led to crypto currency gaining a new advancement and boon to its use in today's date. Blockchain process involves breaking down transaction data into chunks or blocks of a sort which has three main parts that include data, previous hash and its own hash that has been generated by the hash of previous blocks hash and the blocks data. Any tampering to the block leads to the change in the hash values and which in turn leads the hash in the previous block to be untraceable. This makes the blockchain very useful to use in transaction and monetary related functions. The data in the block may include transaction details like sender, receiver, amount, ID etc. The blockchain technology seemed promising at first because of its high level of security that it provides but the cost and complexity installing it was very high. It wasn't possible to implement it everywhere because of its high perpetuity, complexity and network cost that comes with it. Hence the use of blockchain was limited to the crypto currency since it had a high risk while transactions because it's conducted through the network.

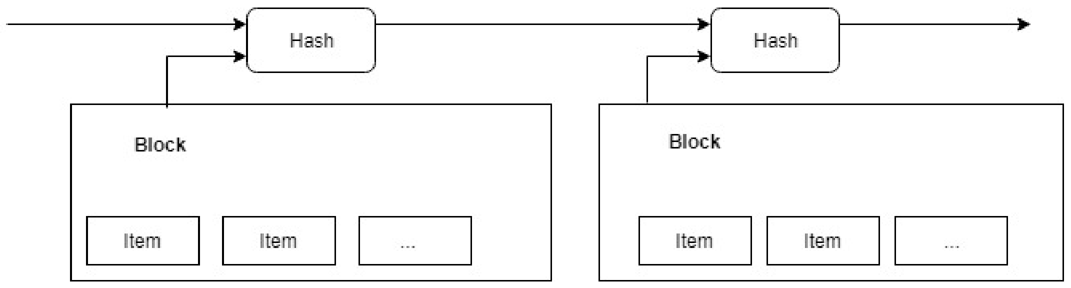


Figure 2.1: Connection Of blocks

Fig 1explains that the transaction related data was secured by having a timestamp server. The timestamp server would broadcast the hash of which the block was added or any changes were made, this proves that the block was made and it'd be relevant to get into the hash of the block. Each timestamp includes the timestamp of the last one in hash format which led to taking the data and the timestamp hash as the input for the new block that is to be generated. This leads to the re- enforcing the previous block by the new block that is created with the timestamp. They introduced a system for digital transactions without depending on trust and created a system of coins made from digital signature, which ensures the rightful ownership of the coins but doesn't provide any means to prevent double spending of the coins. To solve this problem, they introduced a peer to peer. Architecture of the network using proof of work to record the history of digital transactions that quickly becomes theoretically impossible for an attacker to change because of high computational cost, if real nodes control a major of CPU.

[2] 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) Amita Kashyap, G. Sravan Kumar, Sunita Jangir, Emmanuel S. Pilli, Preeti Mishra "IHIDS: Introspection-Based Hybrid Intrusion Detection System in Cloud Environment".

Cloud Computing provides any resources that the user needs as and bill it depending on the usage of those resources. This allows the user to use this resource and keep the cost to a minimal, but also increases the threat of cyber-attacks. Every year many companies report attempts of cyber-attacks on their resources. There are many kinds of attacks that causes users to worry about data confidentiality and integrity, these include DDoS attacks, backdoor attacks, flooding attacks, phishing attacks, port scanning and various attacks on virtual machines and hypervisors. The prevention of these attacks are done by including an IDS to maintain and check any kinds of irregularity in the system and network. detection of any insider or outsider attacks. Any kind of intrusion attempts led to the cloud owner and admin getting the notification related to it. This made the system secure and led to possibility of immediate counter against the attacker. The one major flaw with including an IDS at hypervisor layer was that the IDS only notified and didn't counter the attack itself, it wasn't capable of blocking the attack on its own. IHIDS contains mainly of these given components. The network packet capturing captures the live packets that are going through the virtual machine. The Wireshark packet sniffing tool is used for this purpose. The Wireshark is configured and embedded into the hypervisor layer to capture malicious data packets that are harmful to your system at the virtual bridge. The packet pre-processing includes holding off the .png files and extracting the header information from the data packets which is very useful for knowing the type of attack. After this, attack detection queries are fired which is used to identify the type of attack and generate a subsequent alert to the cloud admin and the cloud owner. System call tracing detects anomalies in the behavior by studying the logs. For this, an advance VMI tool named Drakvuf installed in the VMM layer. A modified version of hypervisor is installed which enables only certain permissible actions to be taken when dealing with the related resources like devices and memory. The trace pre- processing involves numbering each system call and storing them in a particular CSV file for further study of behavior and detection in the system. Finally, the alert generation and logging involve generation of logs based on anomalies in the normal procedures and also storing logs related to such in safe files. The relevant alert is also sent to the cloud admin and cloud owner.

[3] Watanabe, H., Fujimura, S., Nakadaira, A., Miyazaki, Y., Akutsu, A., Kishigami, J. J. (2015). Blockchain contract: A complete consensus using blockchain. 2015 IEEE 4th Global Conference on Consumer Electronics (GCCEP).

In this referred paper the actual concept was to provide a consensus mechanism to the data in which the publisher specifies that the mechanism was used to provide consensus mechanism to the data which is referred as contract data. A vast number of contractual documents, such as those relating to sales agreements, license papers, and other important documents are made all over the world. Current technology allows us to build record and manage these documents easily. However, the difficulty lies in protection of these documents for a major period of time as digital records make it easier to tamper the records and data. We believe that blockchain will help solve this issue. Because, blockchain itself is robust against attack, anyone can verify its records, and it's troublesome to vary its history. Hence the blockchain was implemented to have a contract transaction so secure that no data can be tampered or retrieved if attacker tries it so. The contract is divided in part of two where the mediator and contractors have a complete say in the part of contract. The contract is divided so that the data or address of the destination contractor, contract information, and the sender contract or information is all stored in hash values and sent through the network. The receiver contractor accepts and then the contract is sent back the first contractor to verify and which is then sent to the contract mediator who then confirms the contract which is then settled. This allows a complete trust and confidentiality between the contractors and mediator. It was one of the best ways of having an online transaction or contract which provided most security and anonymity to the contractor. The technology provided anonymity to the contractors which was itself a pro and a con. The anonymity implies that the data and information on the contractors remained unscathed and no information was leaked while the contract was in progress and no type of attack would breach the security of information, This was also a flaw in the system because the anonymity of the contractors means there is a possibility of fraudulent by the unknown contractors and it would be unable to trace them because the system itself maintains the anonymity of the contractors. The referred paper explains the consensus mechanism through an example of contract between 3 members dealing with a contract. In the consensus mechanism each individual creates a key pair. This key pair contains one private key for decryption and public key for encryption. The contractor sends the public key to the other contractors before handed. In the above diagram where Bob sends its public key to the Alice through which Alice confirms the contract and generates transaction data including the encrypted contract data. Once the transaction is noted and miner process is done, the block is generated related to the transaction on Alice's terminal. Since the blockchain is distributed over the network and Bob gets Alice's transaction data and can decrypt it using the private key that he holds. Once decrypted Bob can agree to the contract and present his consent to the contract. If the consent is given it again encrypts the contract data by Cory's public key and another block is added to the blockchain. This block can further be accessed by Cory who holds the private key to the encrypted contract data. Further, Cory can give his consent according to his will. Once it's done, Cory encrypts the data with the public key provided by Alice. And then another block is added confirming all contractors consent and views. Alice then once again decrypts the data which is to confirm the contract details and then sends the transaction details to the mediator which passes the contract through.They mentioned the use of blockchain for verifying contracts and proposed anew architecture to make this possible. This makes it possible to check the users consent and protection of data using blockchain. We will try to implement this protocol on data that will be stored onto the cloud.

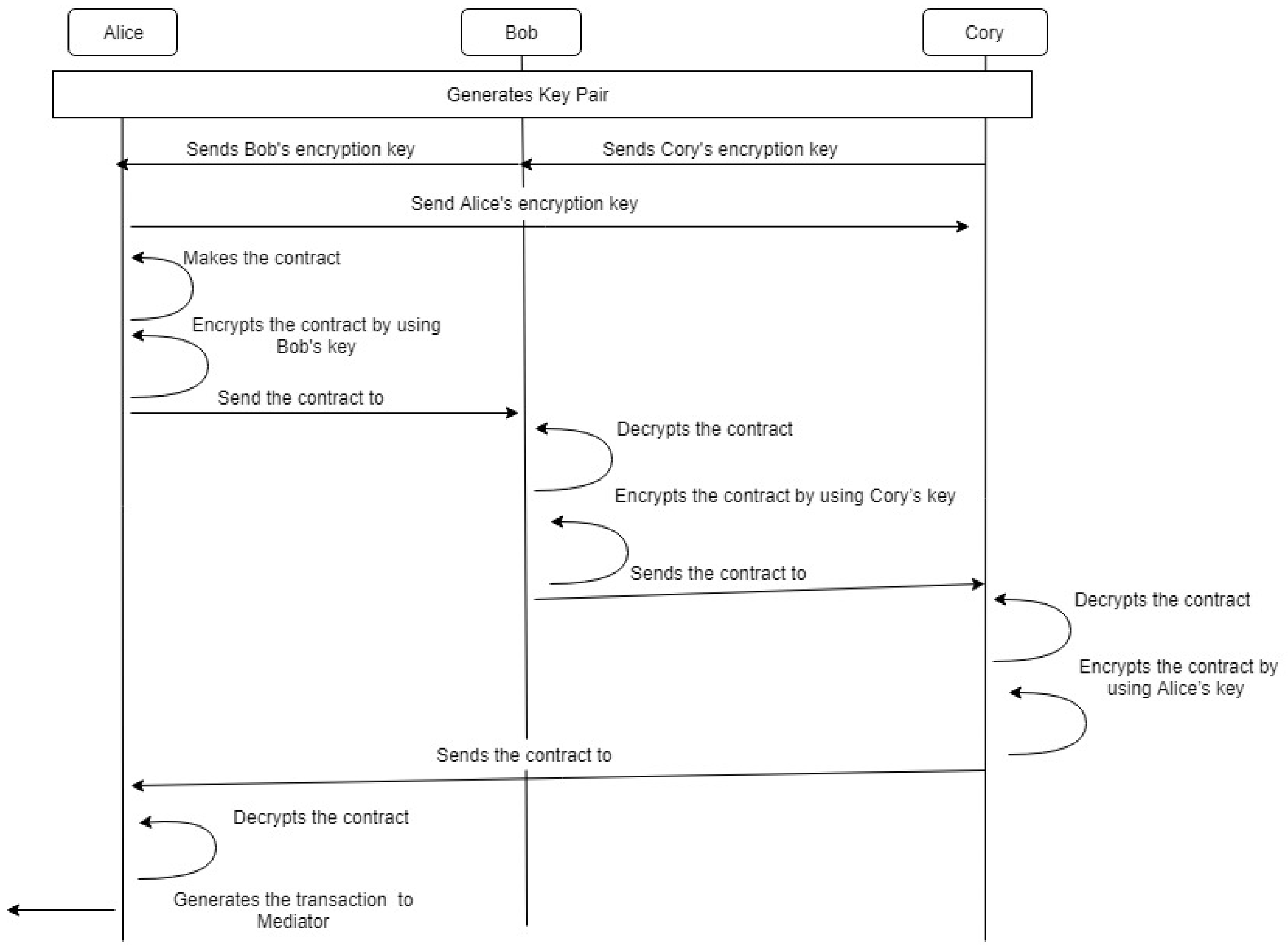

Alice
Bob
Cory
Generates Key Pair
Sends Bob's encryption key
Sends Cory's encryption key
Send Alice's encryption key
Makes the contract
Encrypts the contract by using Bob's key
Send the contract to
Decrypts the contract
Encrypts the contract by using Cory's key
Sends the contract to
Decrypts the contract
Encrypts the contract by using Alice's key
Sends the contract to
Decrypts the contract
Generates the transaction to Mediator

Figure 2.2: Consensus Mechanism

Chapter 3

Project Design

System Architecture

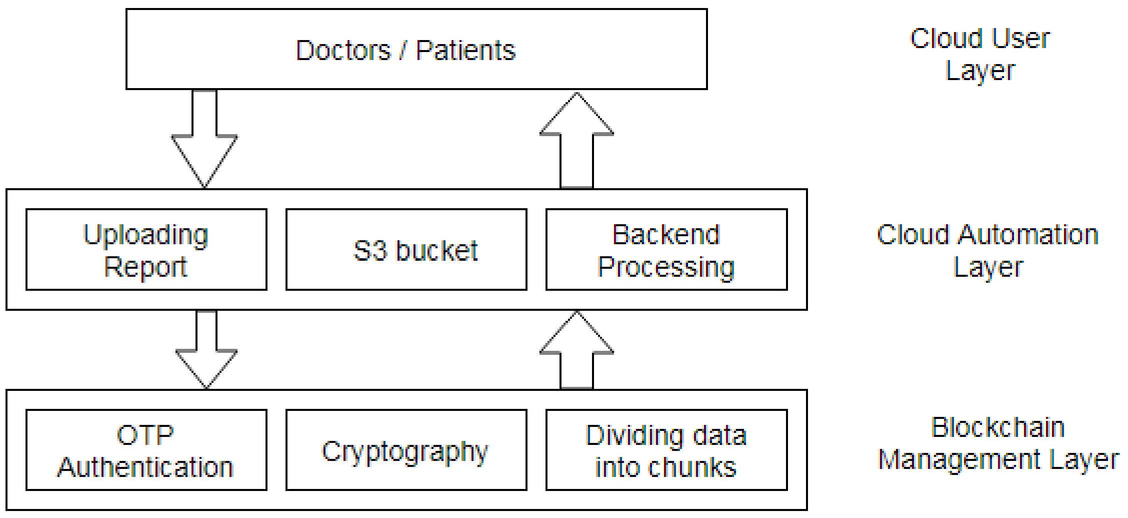


Figure 3.1: System Architecture

These are the primary objectives of our proposed system architecture:

- To divide the data into chunks.
- To maintain a proper encryption of the chunks.
- To have interlink between the chunks to form a chain of the data.
- Doctors must have a patient's side approval through OTP to access the reports.
- A proper GUI will be created which will help in the working for Doctors and Patients.
- Chunks will be stored into S3 buckets on AWS.

Data Chunks: Dividing data into chunks will increase the data integrity, by making the data more complicated at the backend as a single report will be broken into smaller files each containing some part of the report.

Cryptography: AES Encryption fill help secure the data and maintain confidentiality, each chunk of the report will be encrypted through AES individually

Inter-linking: These chunks of report need to have some sort of connection to maintain authenticity of the data this can be achieved through linking each chunk with the next chunk following the concept of chaining of blockchain.

Consensus: Approval from the patient side should be maintained to achieve a trust between doctor and patient; these can be done by maintaining OTP verification from patient's side for the doctors to download the report.

S3 buckets: Storage of these chunks of data needs to be in a secure environment and what's better than using a cloud environment, using S3 buckets to maintain the storage of file chunks. S3 makes to storage more reliable and durable.

The overall working of the proposed system will have an UI for any hospital through which the patients can upload their reports and doctors can view those reports. Doctors need to request the patients for downloading the report which requires an OTP that's been send on to the patient's email. As on for storage once the patient uploads the report the report is divided into chunks of file and these chunks are encrypted using AES, these chunks of data are distributed among 3 buckets that are formed using S3 an storage service by AWS. These buckets store the chunks of file randomly distributed among them and have an interlink to form a chain within the chunks.

Use Case Diagram

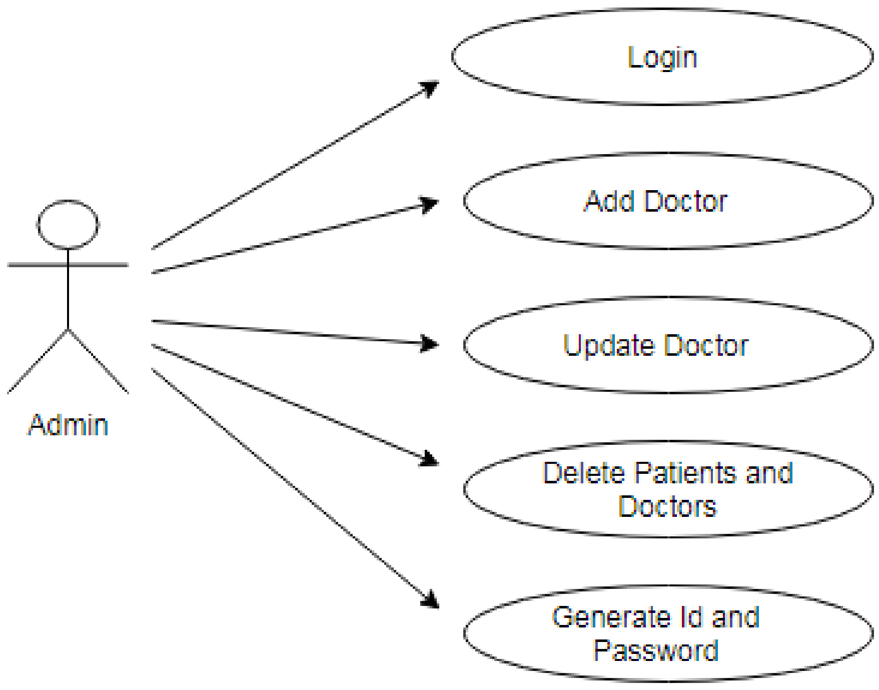


Figure 3.2: Use Case Diagram for Admin

The admin has the authority to add doctors, update doctors, delete patients and doctors. Once the admin add the doctor, Id and password for doctors login will be generated and will be available through Email.Doctor should use those credentials to login into his/her account.

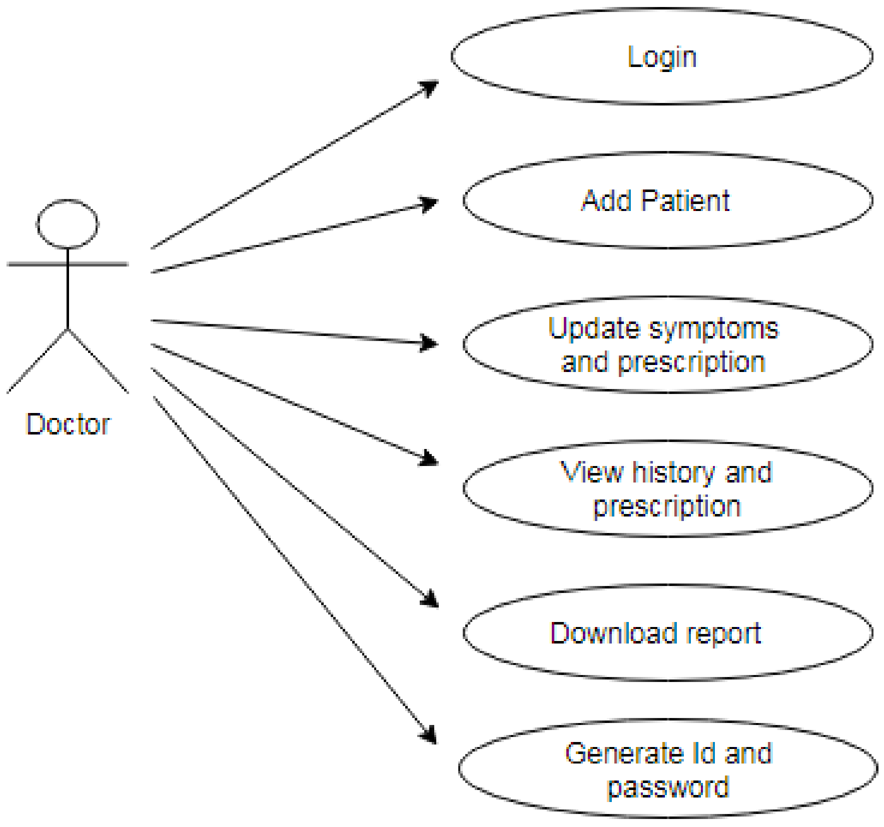

Doctor
Login
Add Patient
Update symptoms and prescription
View history and prescription
Download report
Generate Id and password

Figure 3.3: Use Case Diagram for Doctor

Once the doctor is logged in to his/her account, doctor can add the patients, view the history and prescription of the patients, update symptoms and prescription, download the patients report. When the doctor add the patient, patient receives the login credential through Email. The Email will contain the Id and password for the account.

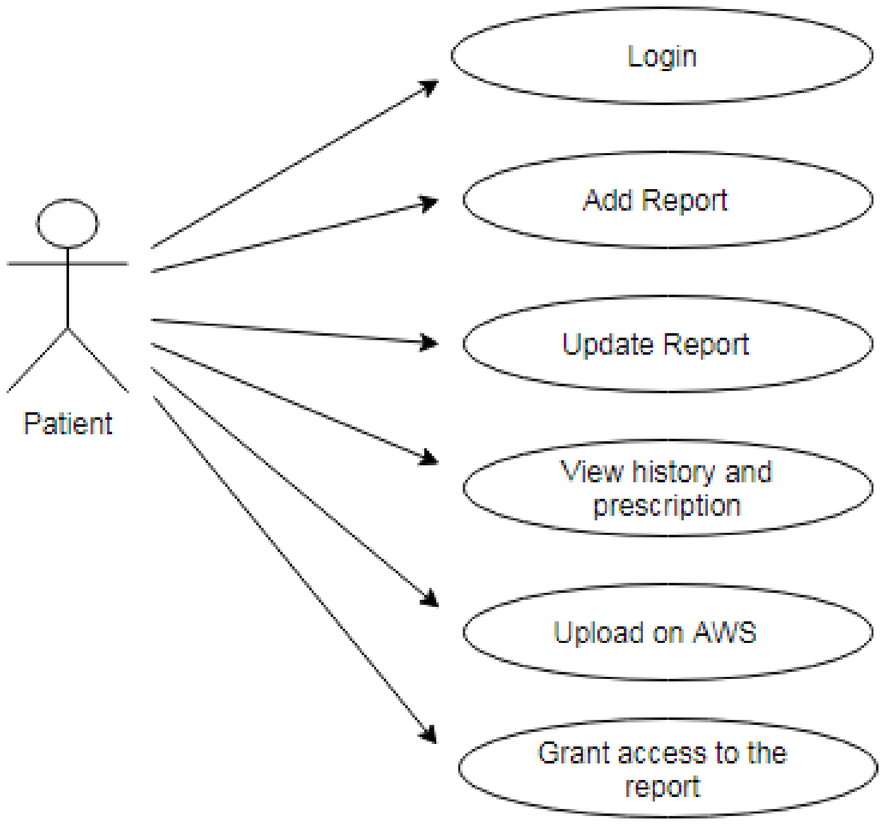


Figure 3.4: Use Case Diagram for Patient

With the help of the credentials patient can login into the account.Patient can upload the report and can also update the report.Patient can view his/her history and the prescription given by the doctor. The report uploaded by the patient is uploaded on AWS. Patient grands the access to their report by giving the OTP received on email when requested by doctor.

Activity Diagram

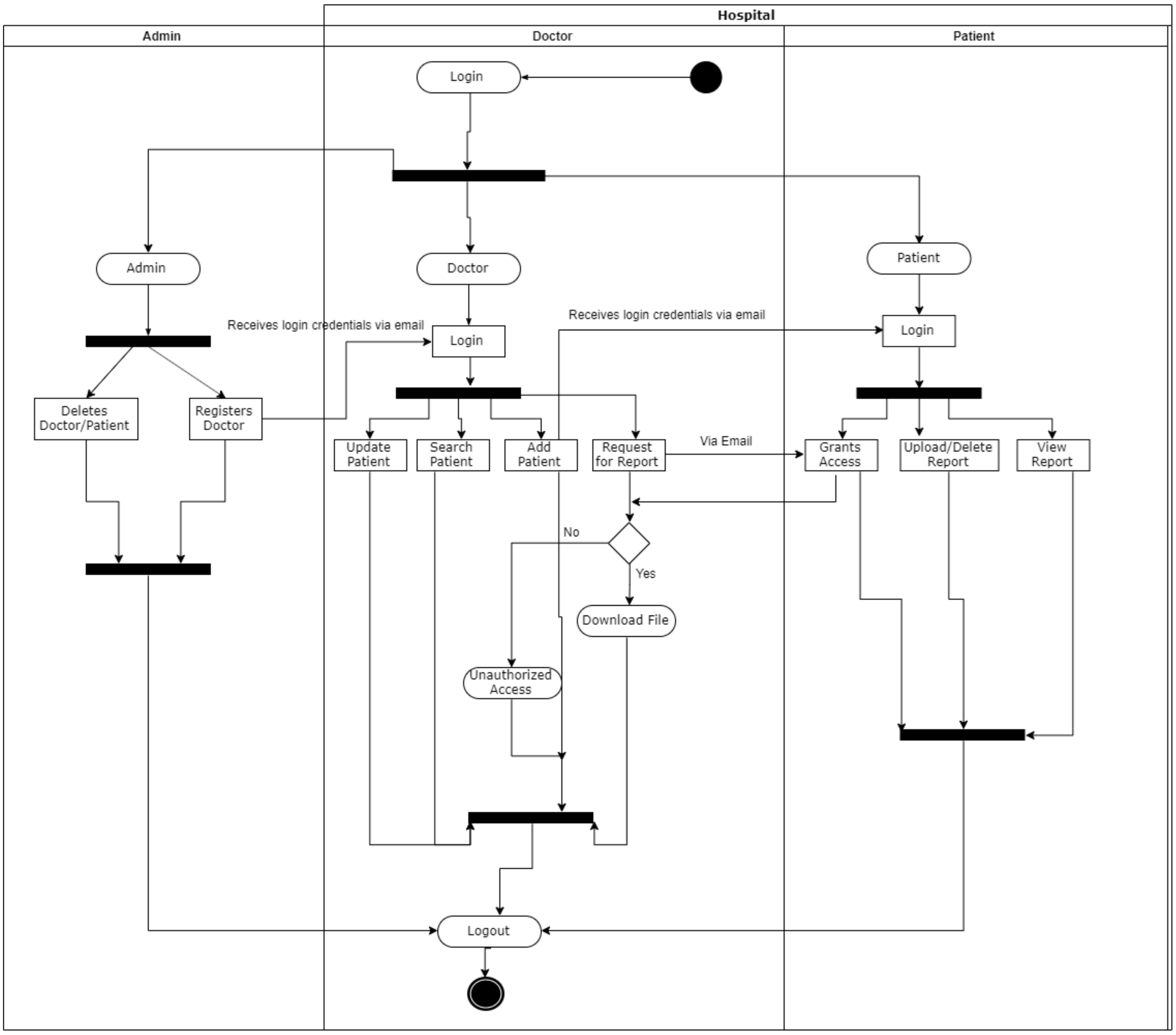


Figure 3.5: Activity Diagram

The above diagram shows all the activities and flow of it. The admin can add the doctors, delete the doctors as well as patients.with the help of the ID password given by admin, doctors can login into their respective accounts. They have the privileges to add the patients, view their history, update their symptoms, request for the report and can also download the report. The patients can view their history, report, upload the report and update the report. They can also grant the access to doctors for downloading the report.

Class Diagram

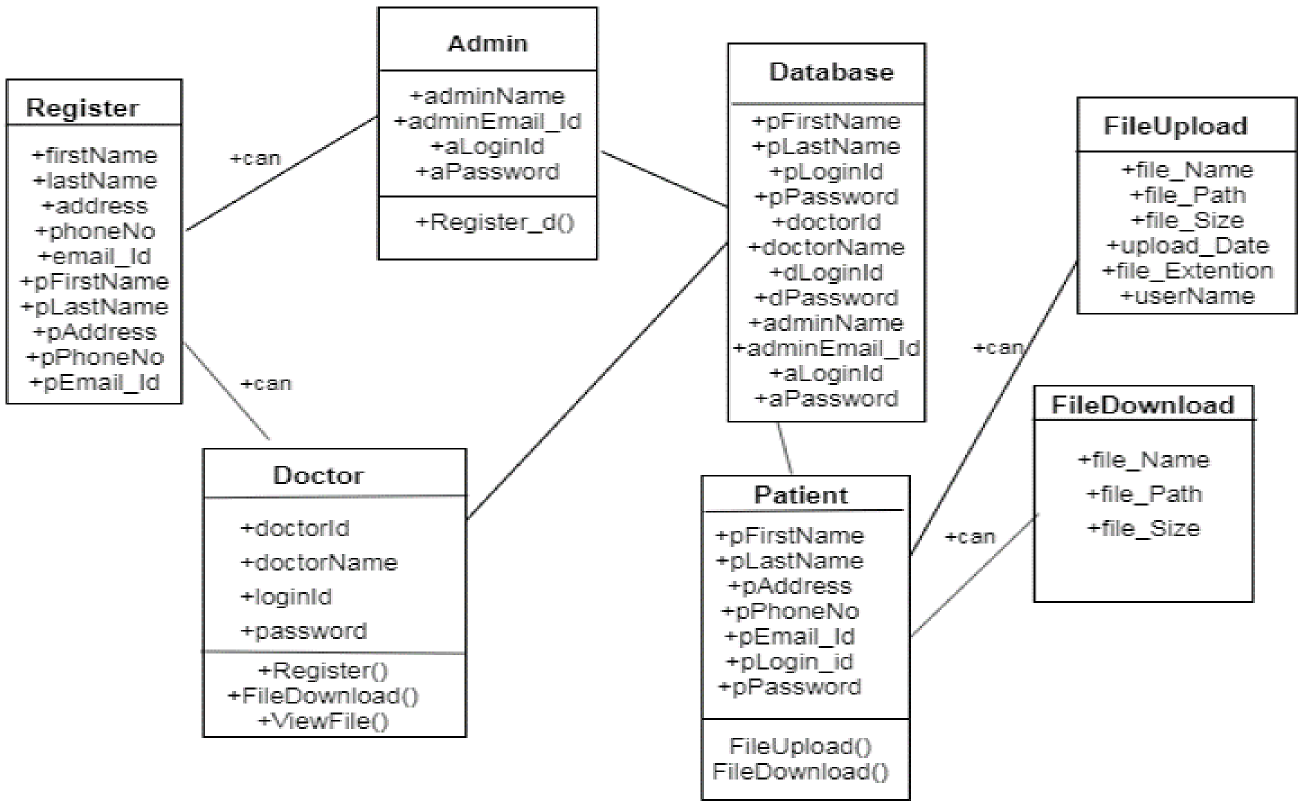


Figure 3.6: Class Diagram

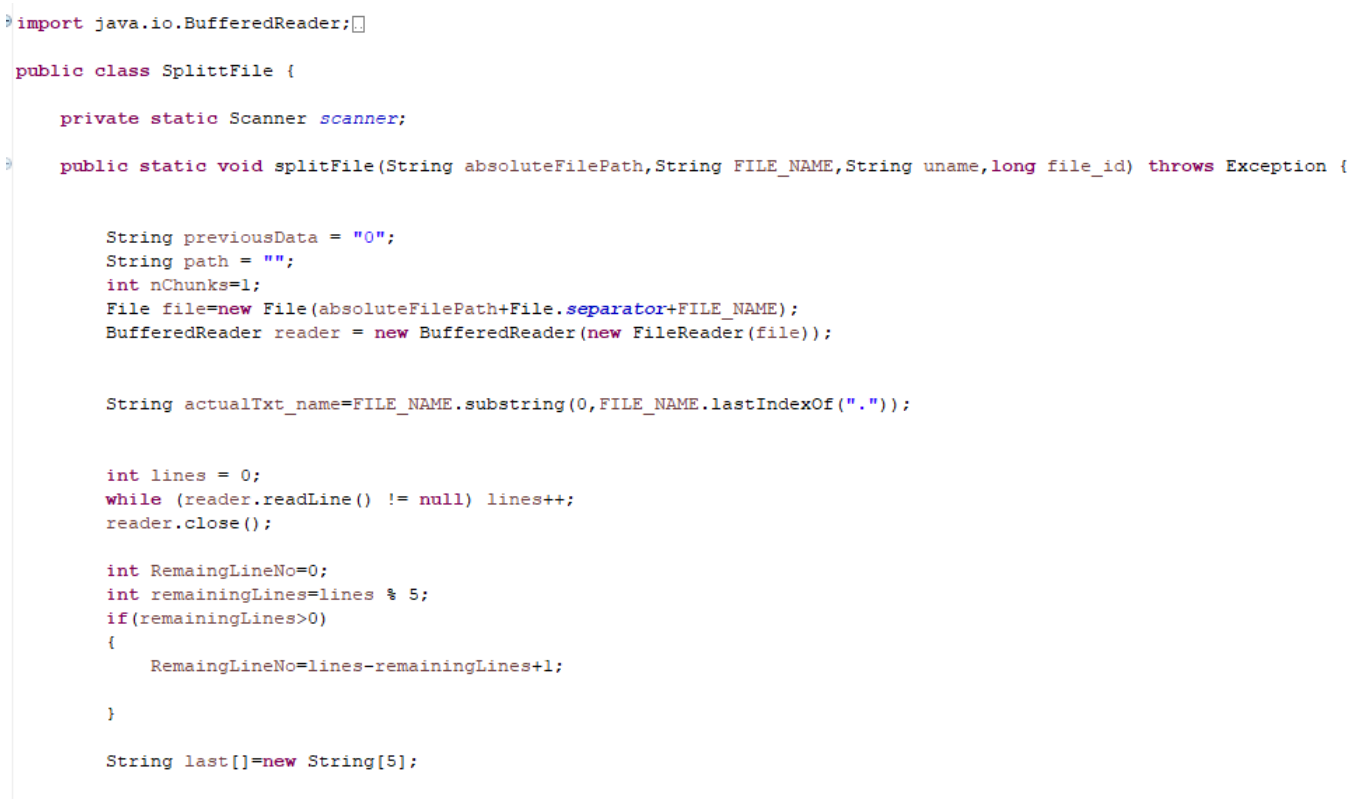
The Class Diagram represents the detail functions and the relation between the objects, the same is represented for our project showing how Admin,Doctor and Patients have an interconnection between them and what are the attributes for each object.

.

Chapter 4

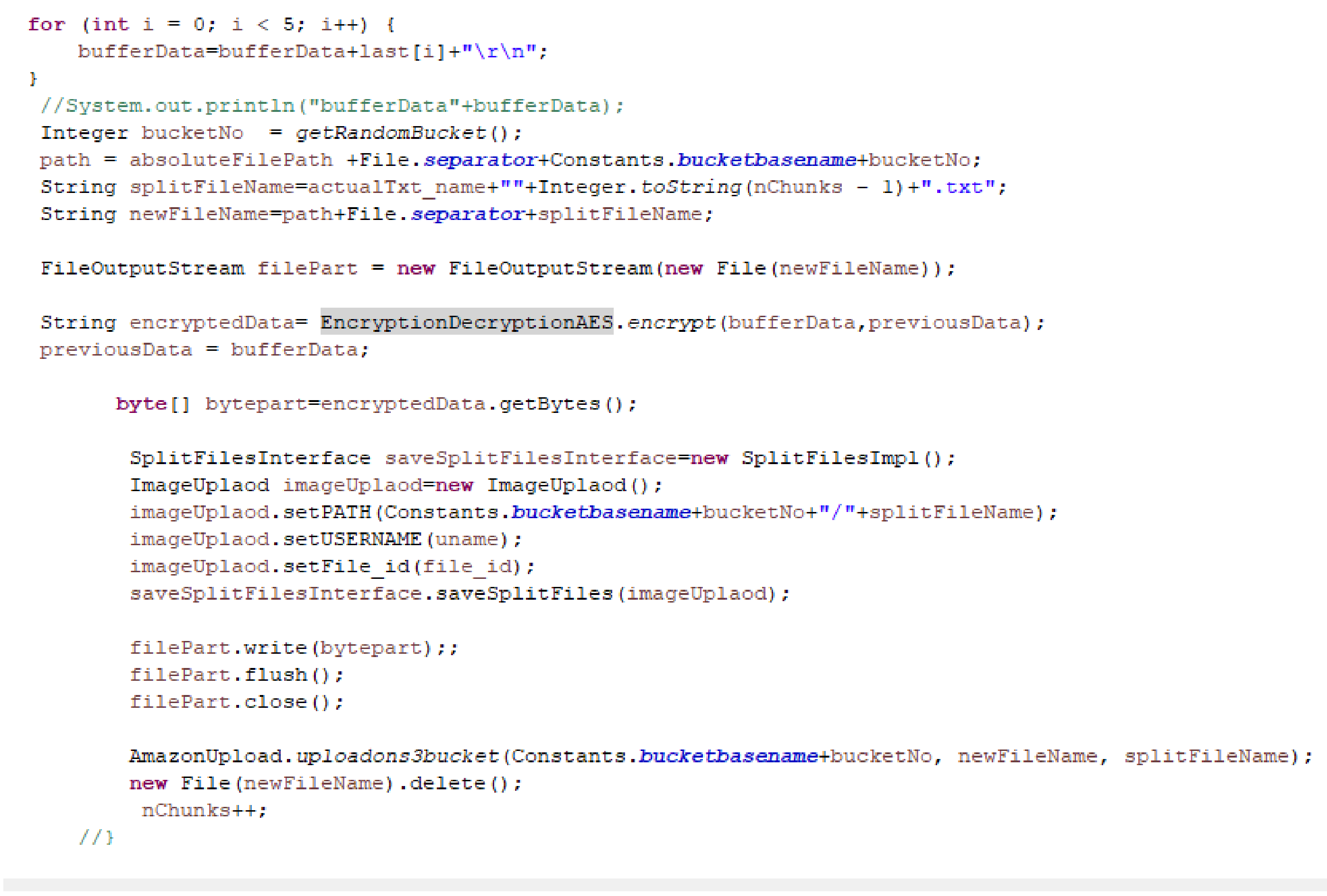
Project Implementation

CODE SNIPPETS



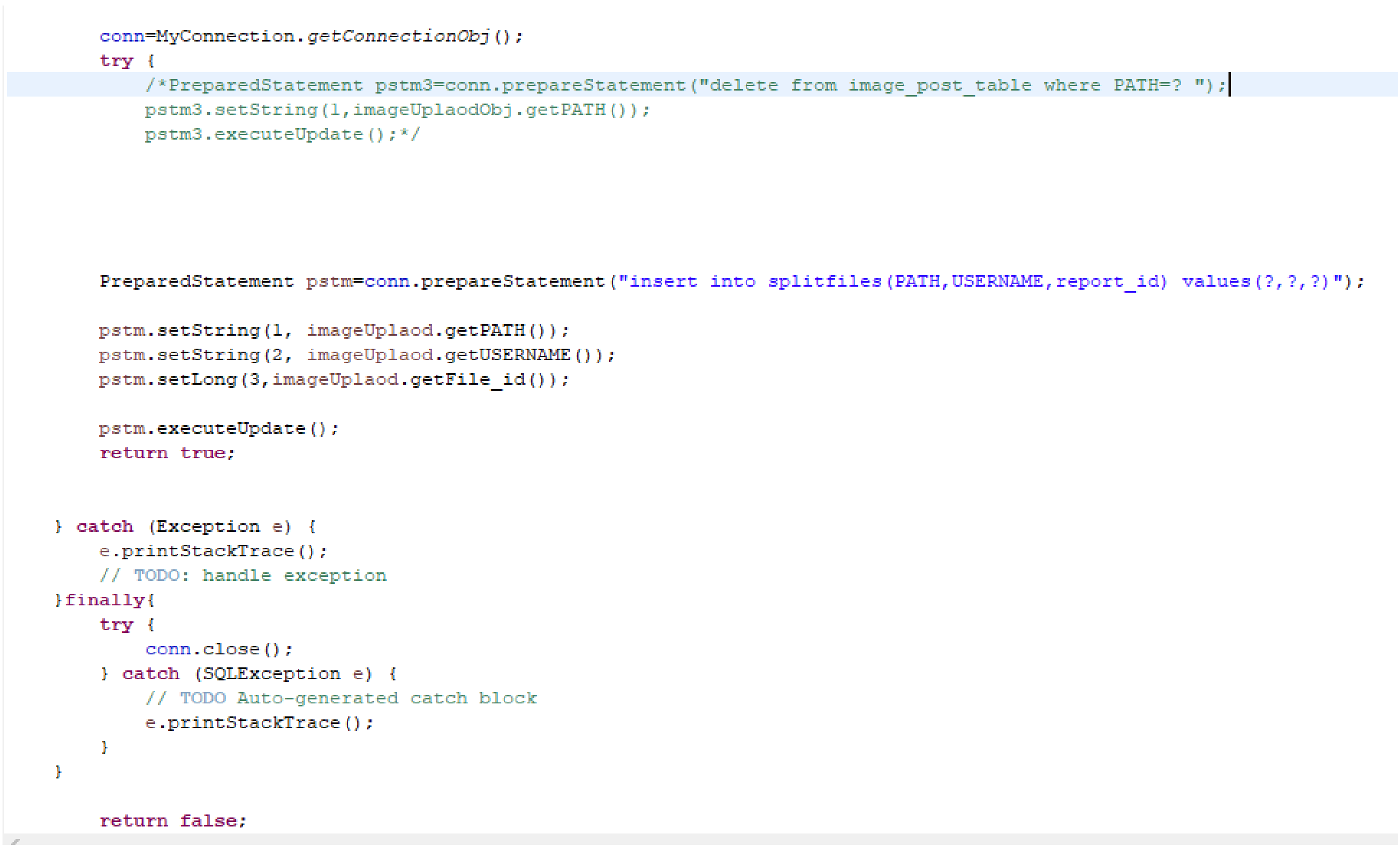
```java
import java.io.BufferedReader;

public class SplittFile {

    private static Scanner scanner;

    public static void splitFile(String absoluteFilePath,String FILE_NAME,String uname,long file_id) throws Exception {


        String previousData = "0";
        String path = "";
        int nChunks=1;
        File file=new File(absoluteFilePath+File.separator+FILE_NAME);
        BufferedReader reader = new BufferedReader(new FileReader(file));


        String actualTxt_name=FILE_NAME.substring(0,FILE_NAME.lastIndexOf("."));


        int lines = 0;
        while (reader.readLine() != null) lines++;
        reader.close();

        int RemaingLineNo=0;
        int remainingLines=lines % 5;
        if(remainingLines>0)
        {
            RemaingLineNo=lines-remainingLines+1;

        }

        String last[]=new String[5];
```

Figure 4.1: Split Function



```
for (int i = 0; i < 5; i++) {
    bufferData=bufferData+last[i]+"\r\n";
}
 //System.out.println("bufferData"+bufferData);
 Integer bucketNo  = getRandomBucket();
 path = absoluteFilePath +File.separator+Constants.bucketbasename+bucketNo;
 String splitFileName=actualTxt_name+""+Integer.toString(nChunks - 1)+".txt";
 String newFileName=path+File.separator+splitFileName;

 FileOutputStream filePart = new FileOutputStream(new File(newFileName));

 String encryptedData= EncryptionDecryptionAES.encrypt(bufferData,previousData);
 previousData = bufferData;

       byte[] bytepart=encryptedData.getBytes();

        SplitFilesInterface saveSplitFilesInterface=new SplitFilesImpl();
        ImageUplaod imageUplaod=new ImageUplaod();
        imageUplaod.setPATH(Constants.bucketbasename+bucketNo+"/"+splitFileName);
        imageUplaod.setUSERNAME(uname);
        imageUplaod.setFile_id(file_id);
        saveSplitFilesInterface.saveSplitFiles(imageUplaod);

        filePart.write(bytepart);;
        filePart.flush();
        filePart.close();

        AmazonUpload.uploadons3bucket(Constants.bucketbasename+bucketNo, newFileName, splitFileName);
        new File(newFileName).delete();
         nChunks++;
    //}
```

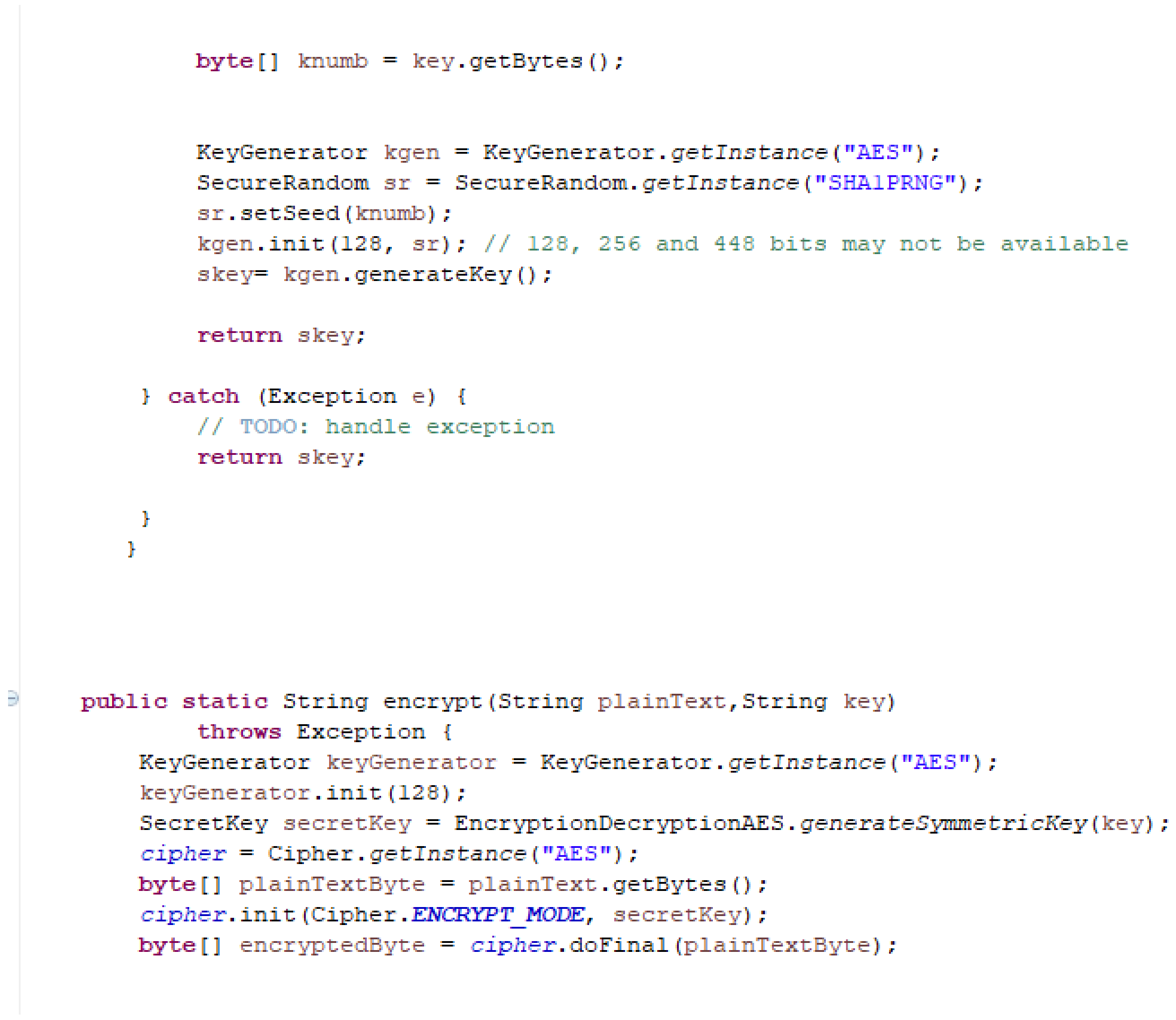
Figure 4.2: Encryption of Split Function



```
        conn=MyConnection.getConnectionObj();
        try {
            /*PreparedStatement pstm3=conn.prepareStatement("delete from image_post_table where PATH=? ");
            pstm3.setString(1,imageUplaodObj.getPATH());
            pstm3.executeUpdate();*/




        PreparedStatement pstm=conn.prepareStatement("insert into splitfiles(PATH,USERNAME,report_id) values(?,?,?)");

        pstm.setString(1, imageUplaod.getPATH());
        pstm.setString(2, imageUplaod.getUSERNAME());
        pstm.setLong(3,imageUplaod.getFile_id());

        pstm.executeUpdate();
        return true;


    } catch (Exception e) {
        e.printStackTrace();
        // TODO: handle exception
    }finally{
        try {
            conn.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

        return false;
```

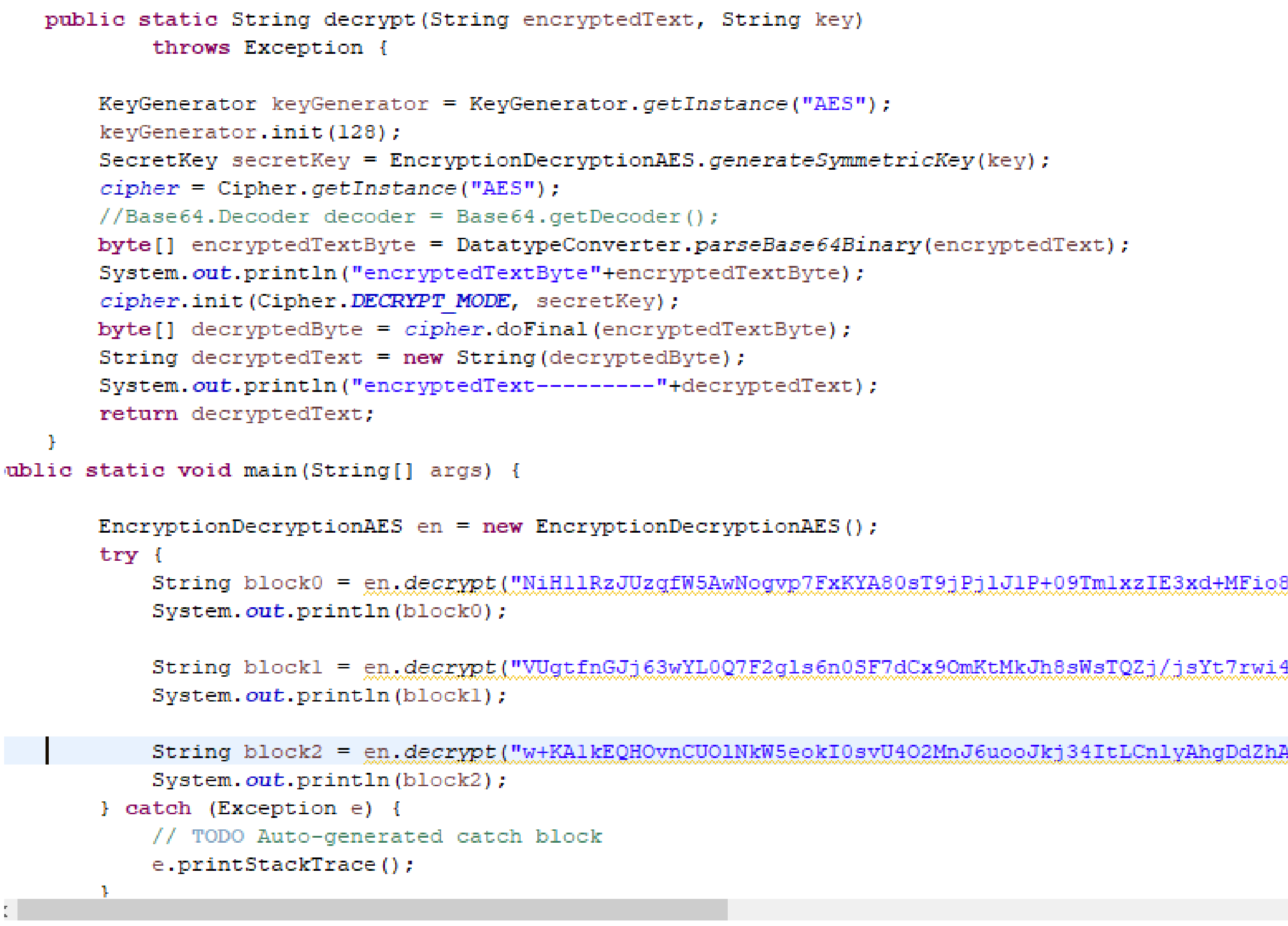
Figure 4.3: Data Connectivity of Split Function

The above code snippets represent how we implemented the function which splits the file and stores them. The 1st code shows how a certain file that is uploaded gets divided based on 5% of the total size of file. The 2nd snippet is about how the chunks of file are encrypted, calling the file EncryptionDecryptionAES invokes the process. The 3rd snippet represents the database schema for storing the chunks of files



```
            byte[] knumb = key.getBytes();


            KeyGenerator kgen = KeyGenerator.getInstance("AES");
            SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
            sr.setSeed(knumb);
            kgen.init(128, sr); // 128, 256 and 448 bits may not be available
            skey= kgen.generateKey();

            return skey;

        } catch (Exception e) {
            // TODO: handle exception
            return skey;

        }
       }



    public static String encrypt(String plainText,String key)
            throws Exception {
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(128);
        SecretKey secretKey = EncryptionDecryptionAES.generateSymmetricKey(key);
        cipher = Cipher.getInstance("AES");
        byte[] plainTextByte = plainText.getBytes();
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedByte = cipher.doFinal(plainTextByte);
```

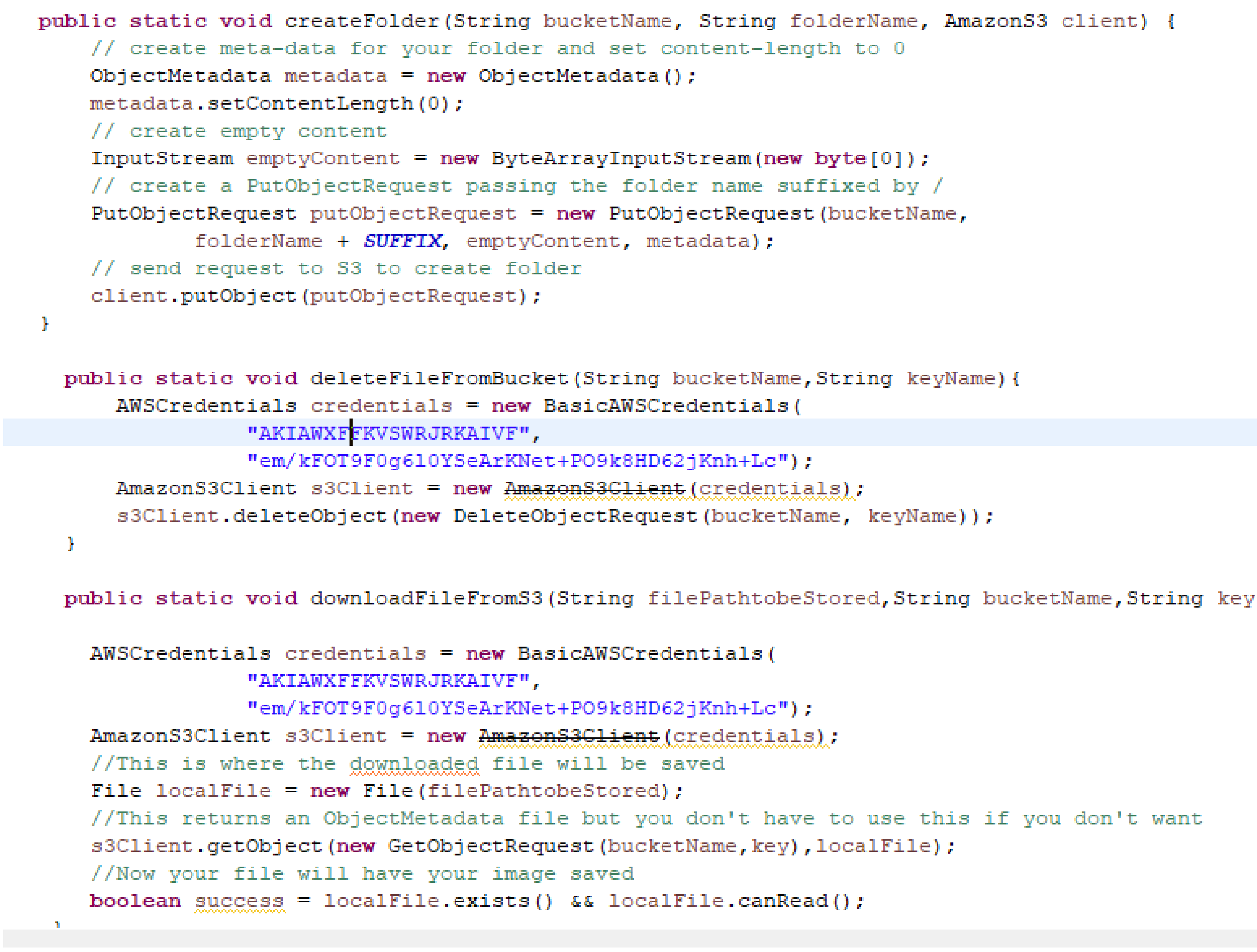
Figure 4.4: AES Function



```
    public static String decrypt(String encryptedText, String key)
            throws Exception {

        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(128);
        SecretKey secretKey = EncryptionDecryptionAES.generateSymmetricKey(key);
        cipher = Cipher.getInstance("AES");
        //Base64.Decoder decoder = Base64.getDecoder();
        byte[] encryptedTextByte = DatatypeConverter.parseBase64Binary(encryptedText);
        System.out.println("encryptedTextByte"+encryptedTextByte);
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decryptedByte = cipher.doFinal(encryptedTextByte);
        String decryptedText = new String(decryptedByte);
        System.out.println("encryptedText---------"+decryptedText);
        return decryptedText;
    }
ublic static void main(String[] args) {

        EncryptionDecryptionAES en = new EncryptionDecryptionAES();
        try {
            String block0 = en.decrypt("NiH11RzJUzqfW5AwNogvp7FxKYA80sT9jPj1J1P+09TmlxzIE3xd+MFio8
            System.out.println(block0);

            String block1 = en.decrypt("VUgtfnGJj63wYL0Q7F2gls6n0SF7dCx9OmKtMkJh8sWsTQZj/jsYt7rwi4
            System.out.println(block1);

            String block2 = en.decrypt("w+KA1kEQHOvnCUO1NkW5eokI0svU4O2MnJ6uooJkj34ItLCnlyAhgDdZhA
            System.out.println(block2);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

Figure 4.5: AES Function

The above code snippets represent the encryption and decryption of the files and also how each file is interlinked with each other to maintain security and follow blockchain mechanism. The block0, block1, block2 represents the encrypted text of each chunk of file in a sequence to form a chain, breaking the sequence of files restricts the data from encryption.



```
public static void createFolder(String bucketName, String folderName, AmazonS3 client) {
    // create meta-data for your folder and set content-length to 0
    ObjectMetadata metadata = new ObjectMetadata();
    metadata.setContentLength(0);
    // create empty content
    InputStream emptyContent = new ByteArrayInputStream(new byte[0]);
    // create a PutObjectRequest passing the folder name suffixed by /
    PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,
            folderName + SUFFIX, emptyContent, metadata);
    // send request to S3 to create folder
    client.putObject(putObjectRequest);
}

  public static void deleteFileFromBucket(String bucketName,String keyName){
      AWSCredentials credentials = new BasicAWSCredentials(
                "AKIAWXFFKVSWRJRKAIVF",
                "em/kFOT9F0g61OYSeArKNet+PO9k8HD62jKnh+Lc");
      AmazonS3Client s3Client = new ~~AmazonS3Client~~(credentials);
      s3Client.deleteObject(new DeleteObjectRequest(bucketName, keyName));
  }

  public static void downloadFileFromS3(String filePathtobeStored,String bucketName,String key

    AWSCredentials credentials = new BasicAWSCredentials(
                "AKIAWXFFKVSWRJRKAIVF",
                "em/kFOT9F0g61OYSeArKNet+PO9k8HD62jKnh+Lc");
    AmazonS3Client s3Client = new ~~AmazonS3Client~~(credentials);
    //This is where the downloaded file will be saved
    File localFile = new File(filePathtobeStored);
    //This returns an ObjectMetadata file but you don't have to use this if you don't want
    s3Client.getObject(new GetObjectRequest(bucketName,key),localFile);
    //Now your file will have your image saved
    boolean success = localFile.exists() && localFile.canRead();
```

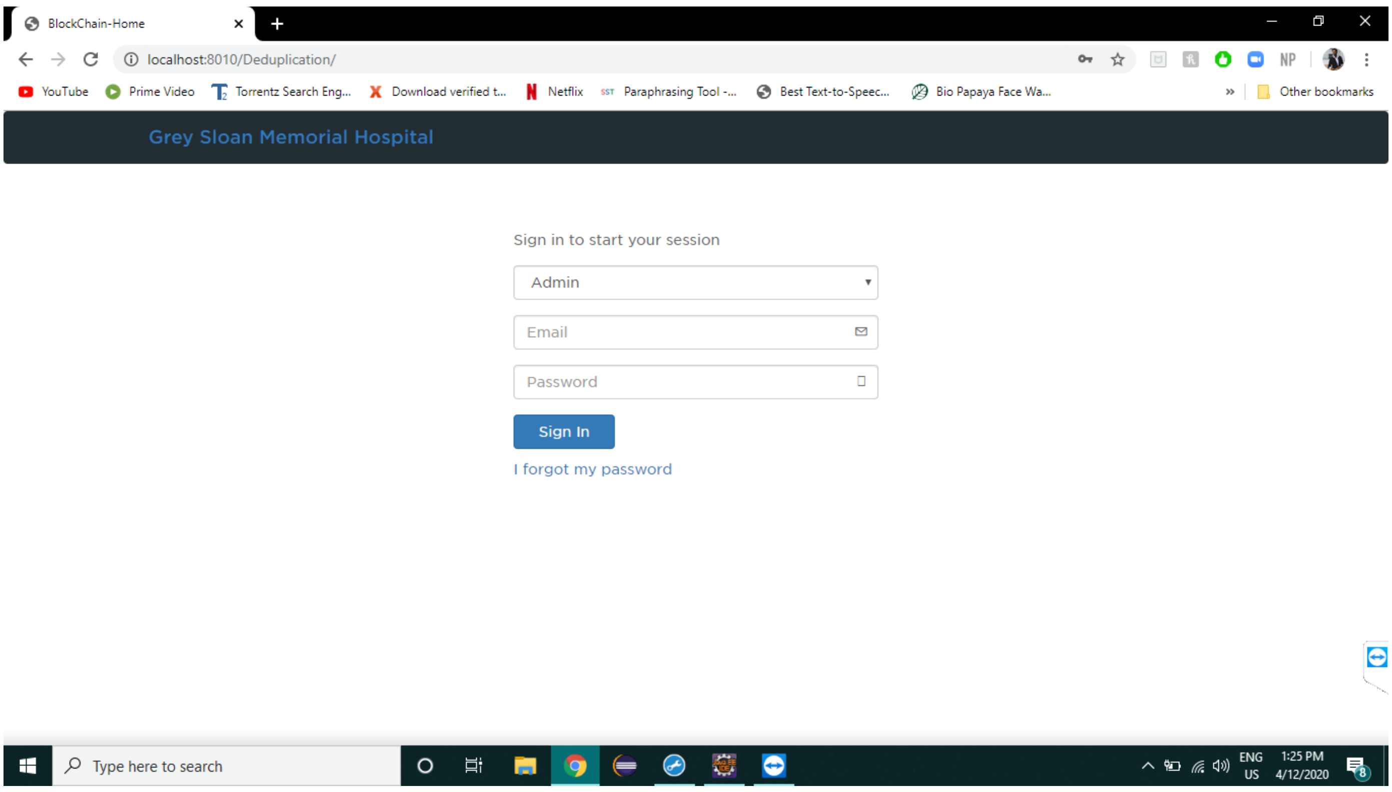
Figure 4.6: Amazon Function

The above code snippet is an java file that helps connect the storage with amazon S3 service. This .java file shifts or data storage on cloud with the help of AWS Credentials and Key that is generated while creating S3 buckets on AWS platform. The data are stored into various buckets randomly.

Chapter 5

Results

ADMIN LOGIN



Grey Sloan Memorial Hospital

Sign in to start your session

Admin

Email

Password

Sign In

I forgot my password

Figure 5.1: Admin login

ADMIN HOME PAGE AFTER LOGIN

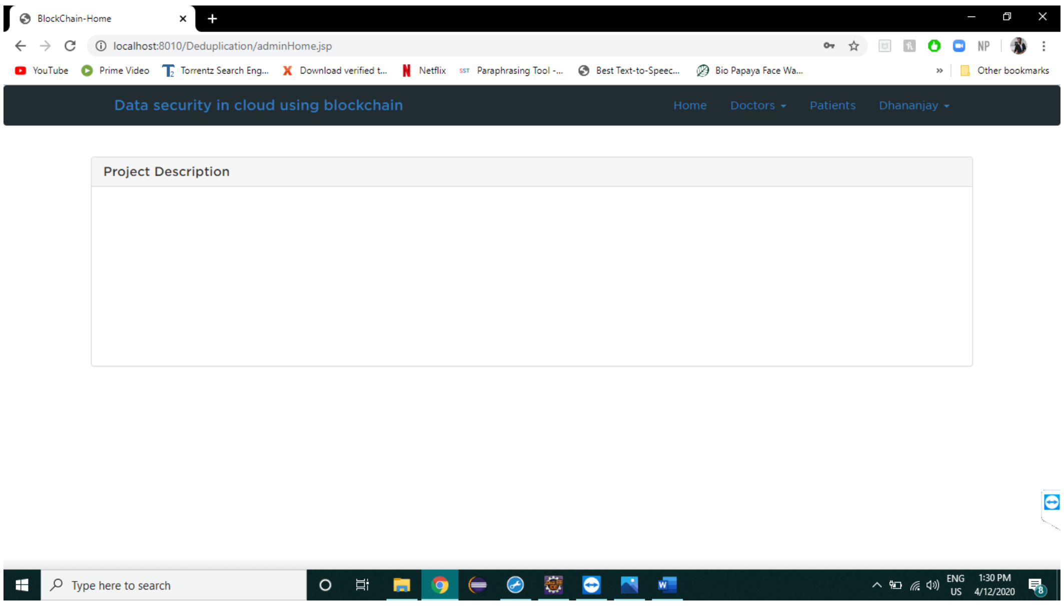

BlockChain-Home
localhost:8010/Deduplication/adminHome.jsp
Data security in cloud using blockchain
Home
Doctors
Patients
Dhananjay
Project Description

Figure 5.2: Admin home page after login

ADD DOCTOR

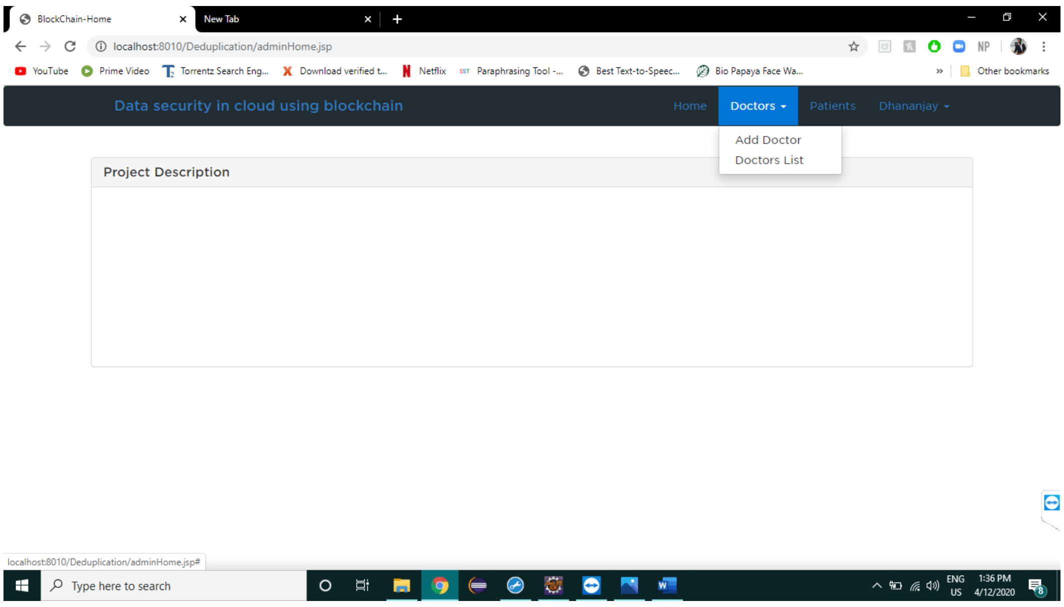
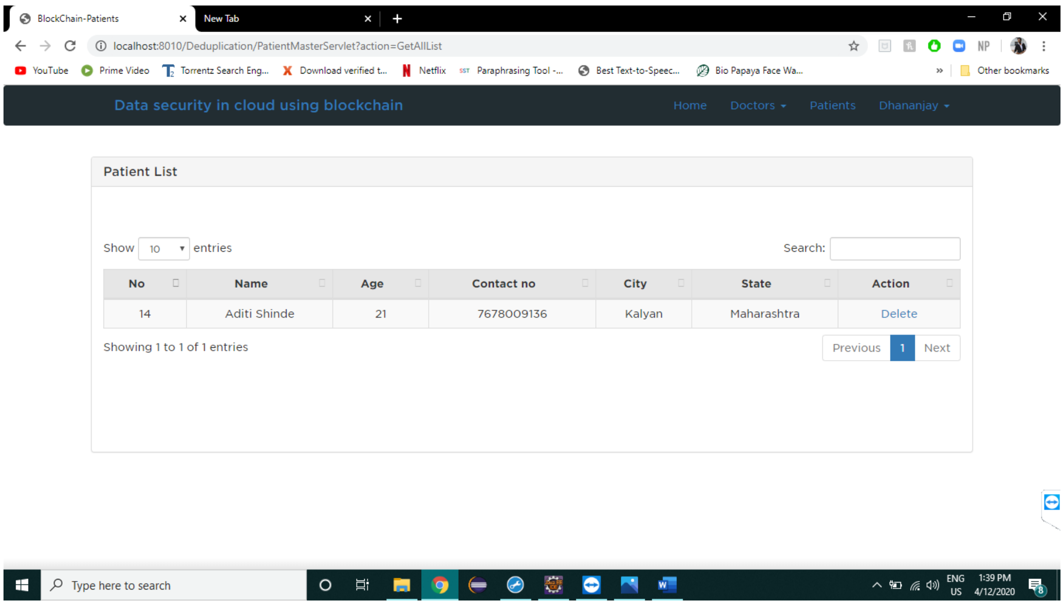


Figure 5.3: Add Doctor

PATIENTS LIST



Data security in cloud using blockchain

Home Doctors Patients Dhananjay

Patient List

Show 10 entries Search:

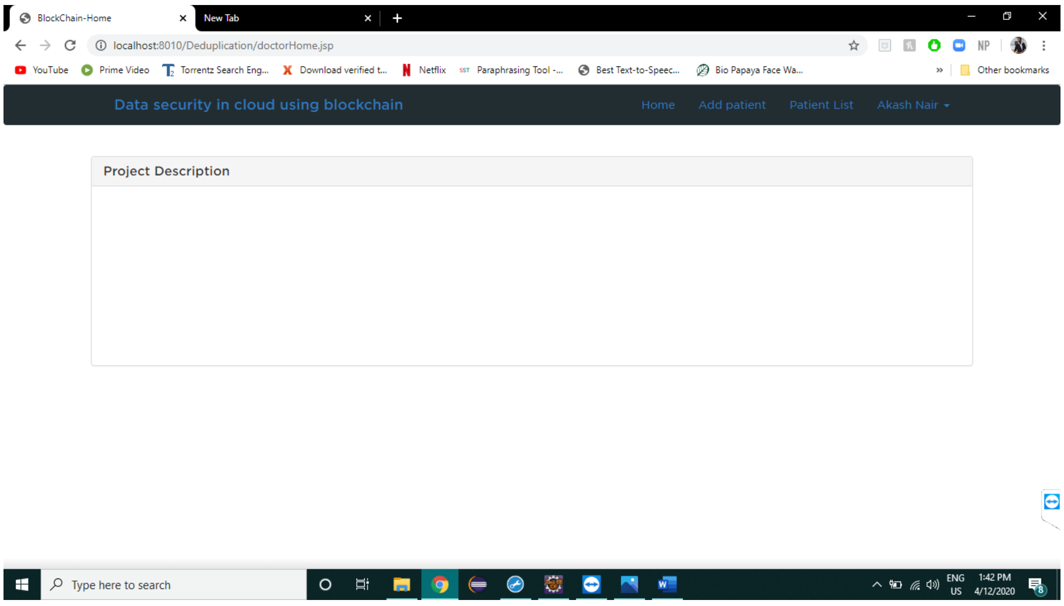
No	Name	Age	Contact no	City	State	Action
14	Aditi Shinde	21	7678009136	Kalyan	Maharashtra	Delete

Showing 1 to 1 of 1 entries

Previous 1 Next

Figure 5.4: Patient list

DOCTOR LOGIN



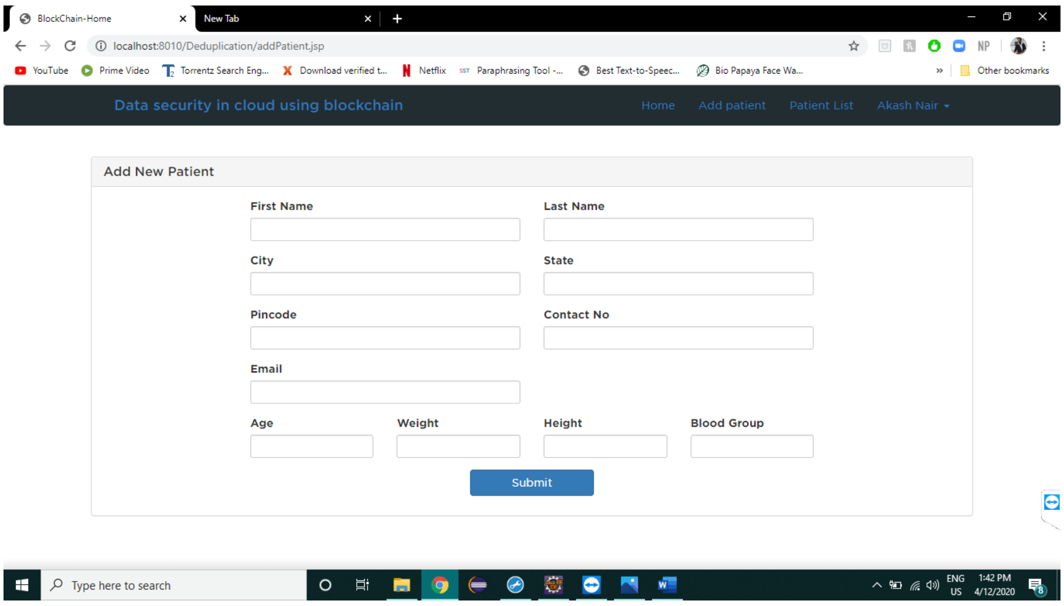
Data security in cloud using blockchain

Home Add patient Patient List Akash Nair

Project Description

Figure 5.5: Doctor Login

ADD PATIENT



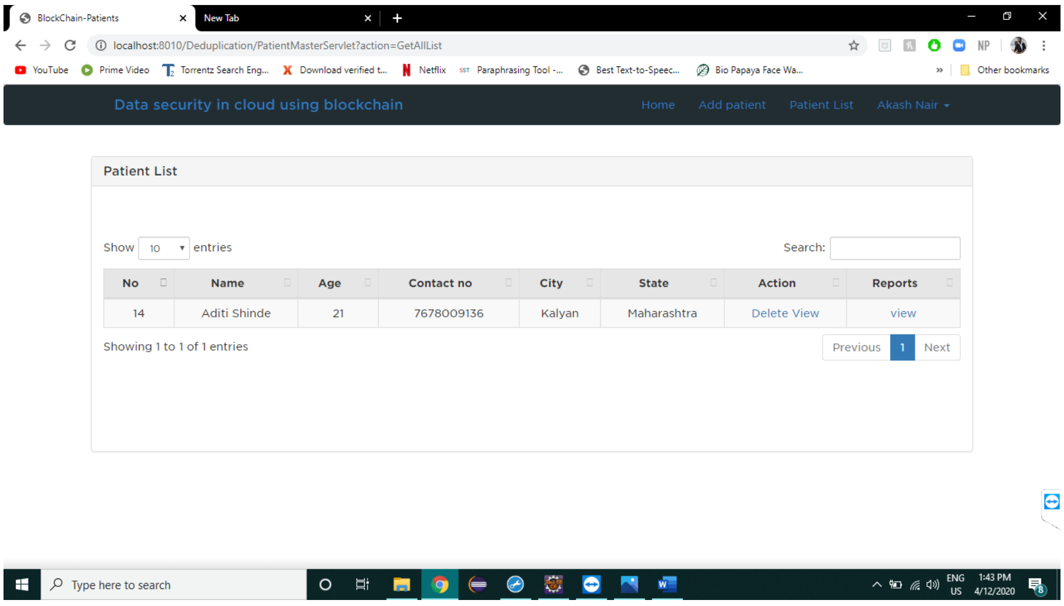
Add New Patient

First Name	Last Name
City	State
Pincode	Contact No
Email	

Age	Weight	Height	Blood Group

Figure 5.6: Add Patient

PATIENT LIST



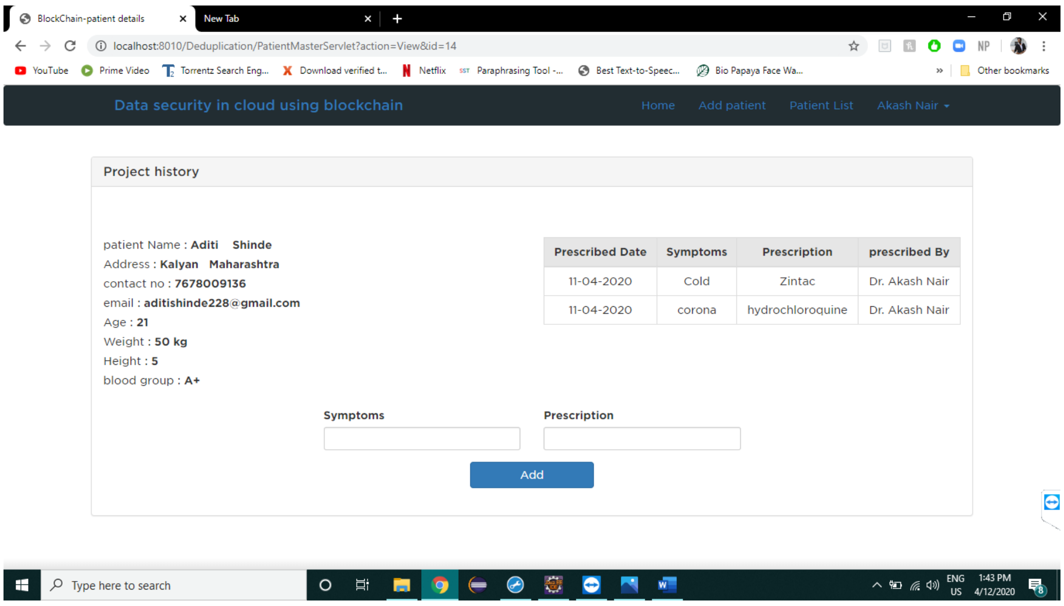
Patient List

No	Name	Age	Contact no	City	State	Action	Reports
14	Aditi Shinde	21	7678009136	Kalyan	Maharashtra	Delete View	view

Showing 1 to 1 of 1 entries

Figure 5.7: Patient list

ADD PRESCRIPTION AND VIEW PROFILE


BlockChain-patient details
New Tab
localhost:8010/Deduplication/PatientMasterServlet?action=View&id=14
Data security in cloud using blockchain
Home
Add patient
Patient List
Akash Nair
Add
Type here to search

Project history

patient Name : **Aditi Shinde**
Address : **Kalyan Maharashtra**
contact no : **7678009136**
email : **aditishinde228@gmail.com**
Age : **21**
Weight : **50 kg**
Height : **5**
blood group : **A+**

Prescribed Date	Symptoms	Prescription	prescribed By
11-04-2020	Cold	Zintac	Dr. Akash Nair
11-04-2020	corona	hydrochloroquine	Dr. Akash Nair

Symptoms
Prescription

Figure 5.8: Add Prescription and view profile

ENTER OTP TO DOWNLOAD REPORT

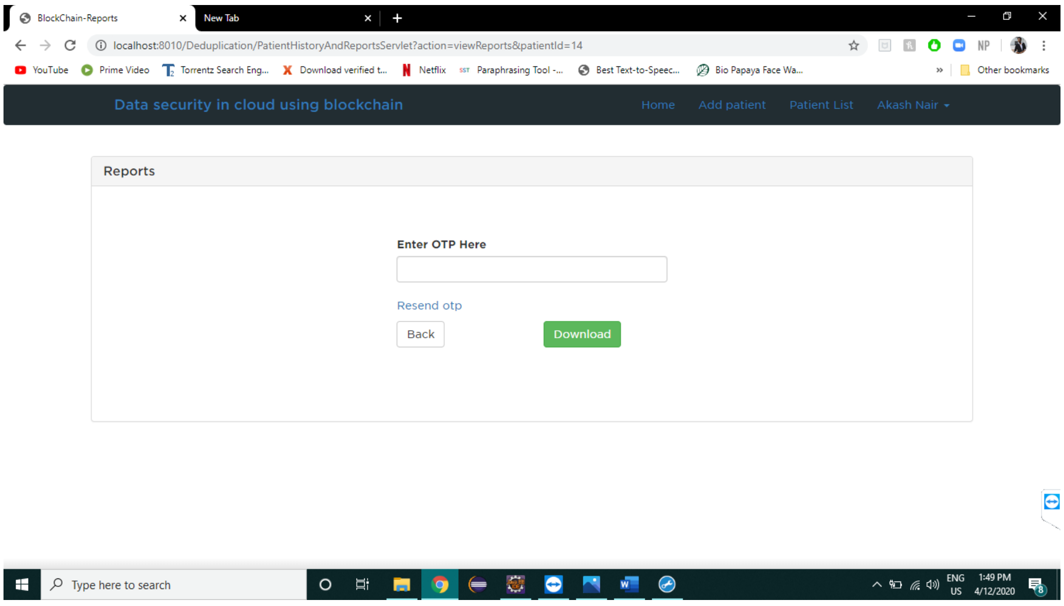

BlockChain-Reports
New Tab
localhost:8010/Deduplication/PatientHistoryAndReportsServlet?action=viewReports&patientId=14
Data security in cloud using blockchain
Home
Add patient
Patient List
Akash Nair
Reports
Enter OTP Here
Resend otp
Back
Download
Type here to search

Figure 5.9: Enter OTP to download report

PATIENT HOMEPAGE

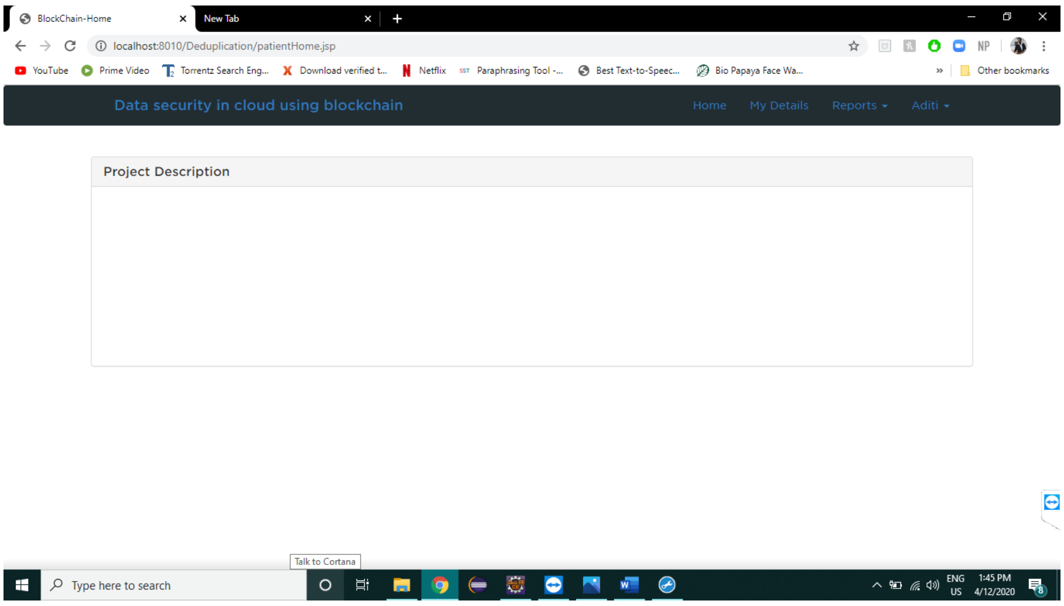
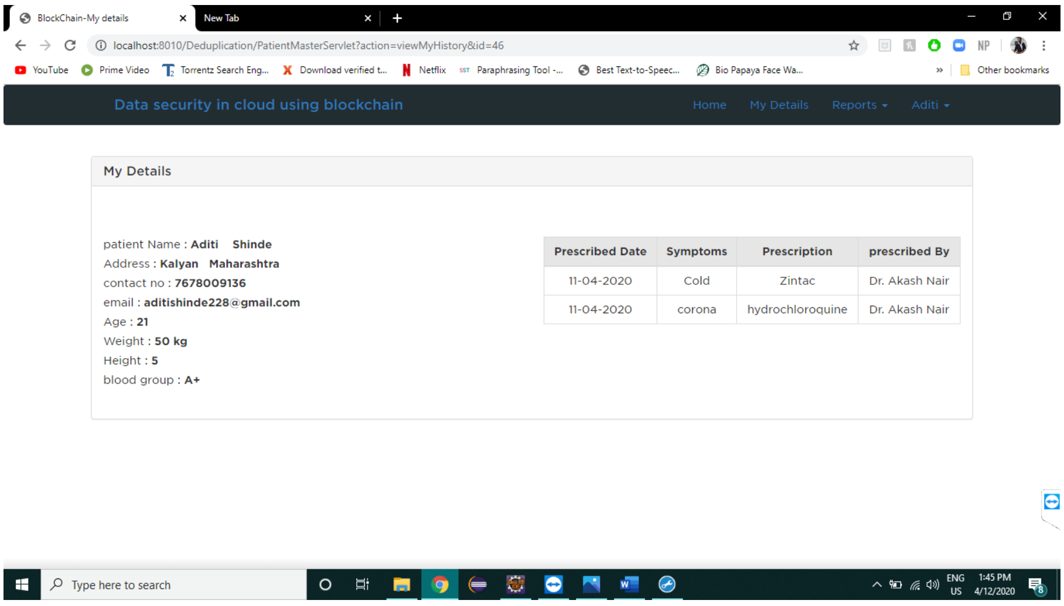


Figure 5.10: Patient homepage

PATIENTS MY PROFILE


BlockChain-My details
New Tab
localhost:8010/Deduplication/PatientMasterServlet?action=viewMyHistory&id=46
Data security in cloud using blockchain
Home
My Details
Reports
Aditi
My Details
Type here to search
1:45 PM
4/12/2020

patient Name : **Aditi Shinde**
Address : **Kalyan Maharashtra**
contact no : **7678009136**
email : **aditishinde228@gmail.com**
Age : **21**
Weight : **50 kg**
Height : **5**
blood group : **A+**

Prescribed Date	Symptoms	Prescription	prescribed By
11-04-2020	Cold	Zintac	Dr. Akash Nair
11-04-2020	corona	hydrochloroquine	Dr. Akash Nair

Figure 5.11: Patient my profile

ADD REPORT

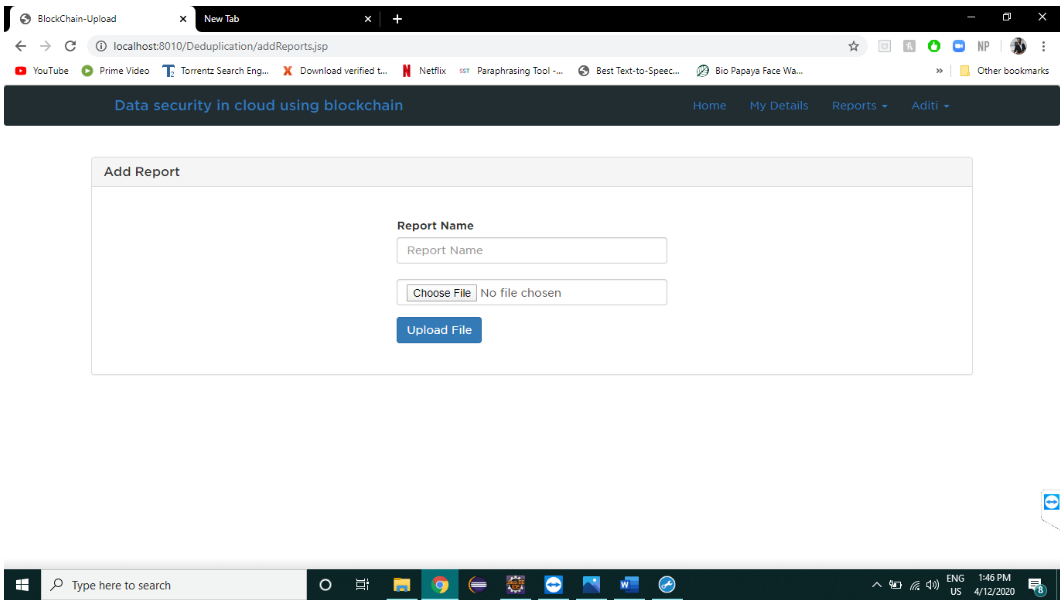
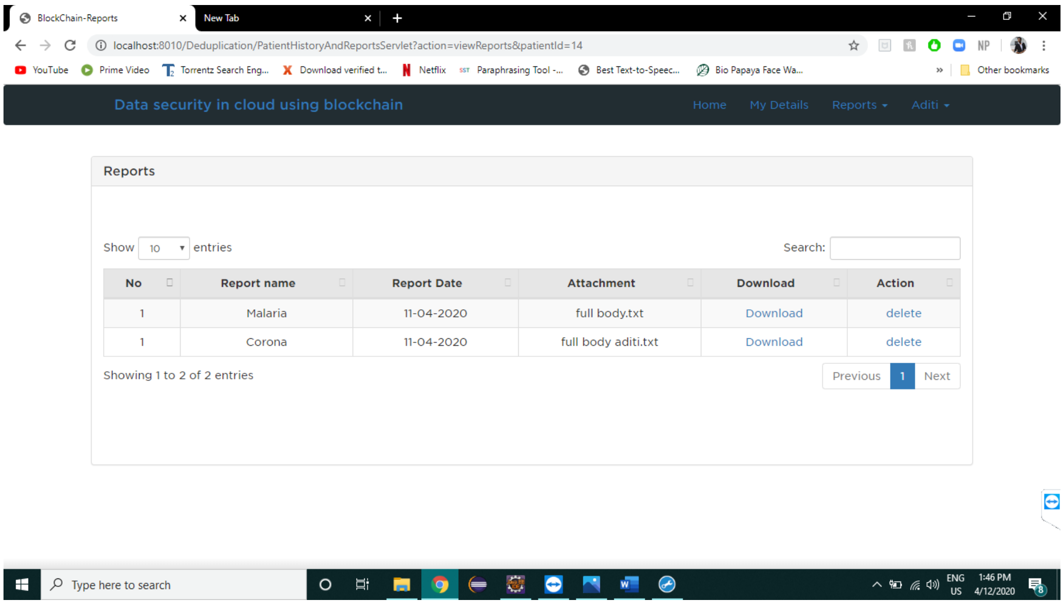


Figure 5.12: Add Report

VIEW REPORT


BlockChain-Reports
New Tab
localhost:8010/Deduplication/PatientHistoryAndReportsServlet?action=viewReports&patientId=14
Data security in cloud using blockchain
Home
My Details
Reports
Aditi
Reports
Show 10 entries
Search:
Previous
1
Next
Type here to search
1:46 PM
4/12/2020

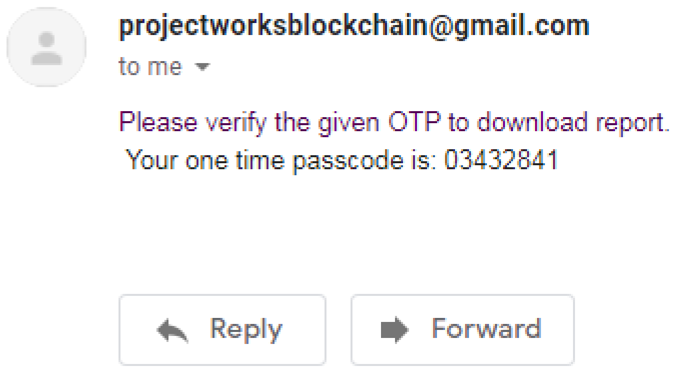
No	Report name	Report Date	Attachment	Download	Action
1	Malaria	11-04-2020	full body.txt	Download	delete
1	Corona	11-04-2020	full body aditi.txt	Download	delete

Showing 1 to 2 of 2 entries

Figure 5.13: View Report

EMAIL SCREENSHOTS:

Otp to download report

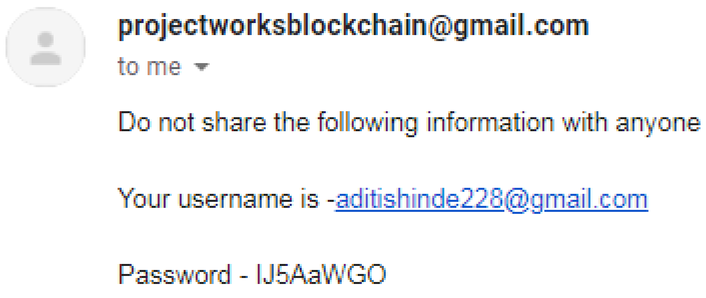


projectworksblockchain@gmail.com
to me

Please verify the given OTP to download report.
Your one time passcode is: 03432841

Figure 5.14: email to download report

Mail after registering a new doctor



projectworksblockchain@gmail.com
to me

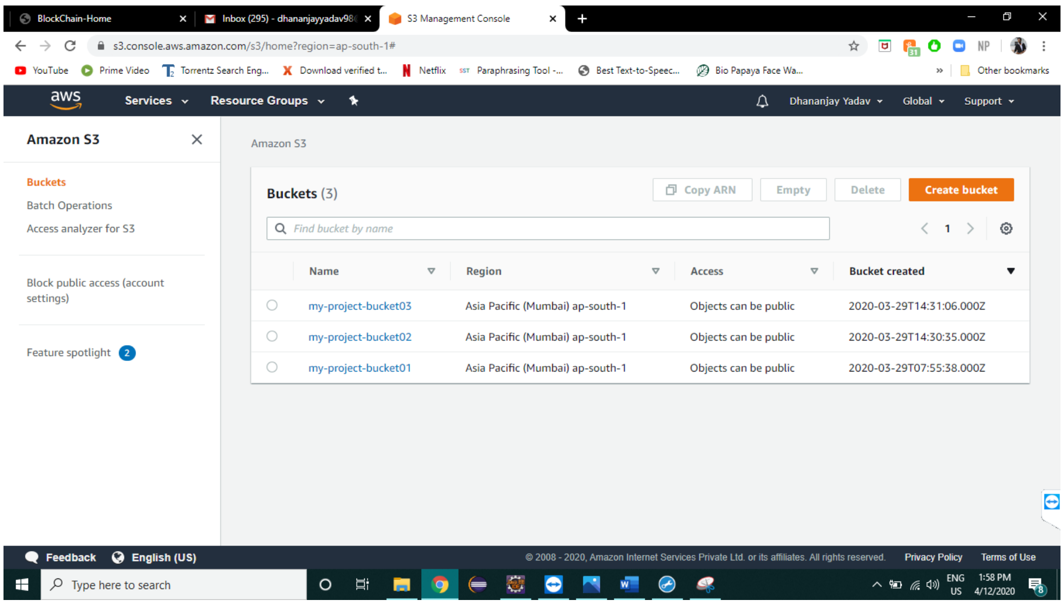
Do not share the following information with anyone

Your username is -aditishinde228@gmail.com

Password - IJ5AaWGO

Figure 5.15: Mail after registering a new doctor

S3 BUCKETS IN AWS



Amazon S3

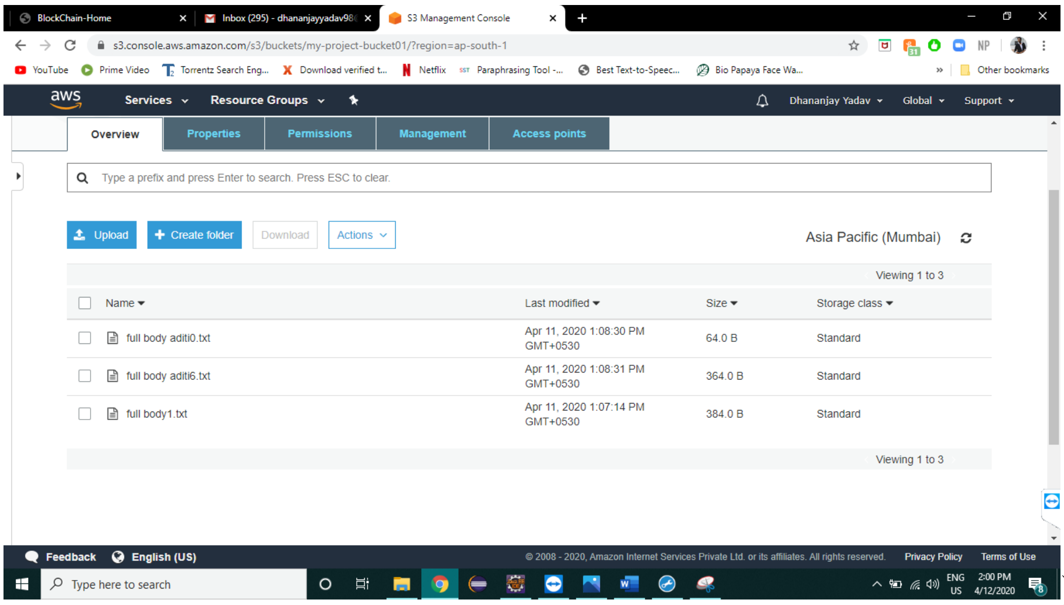
Buckets (3)

Name	Region	Access	Bucket created
my-project-bucket03	Asia Pacific (Mumbai) ap-south-1	Objects can be public	2020-03-29T14:31:06.000Z
my-project-bucket02	Asia Pacific (Mumbai) ap-south-1	Objects can be public	2020-03-29T14:30:35.000Z
my-project-bucket01	Asia Pacific (Mumbai) ap-south-1	Objects can be public	2020-03-29T07:55:38.000Z

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Figure 5.16: S3 buckets in AWS

DATA IN CHUNKS IN BUCKETS



Viewing 1 to 3

Name	Last modified	Size	Storage class
full body aditi0.txt	Apr 11, 2020 1:08:30 PM GMT+0530	64.0 B	Standard
full body aditi6.txt	Apr 11, 2020 1:08:31 PM GMT+0530	364.0 B	Standard
full body1.txt	Apr 11, 2020 1:07:14 PM GMT+0530	384.0 B	Standard

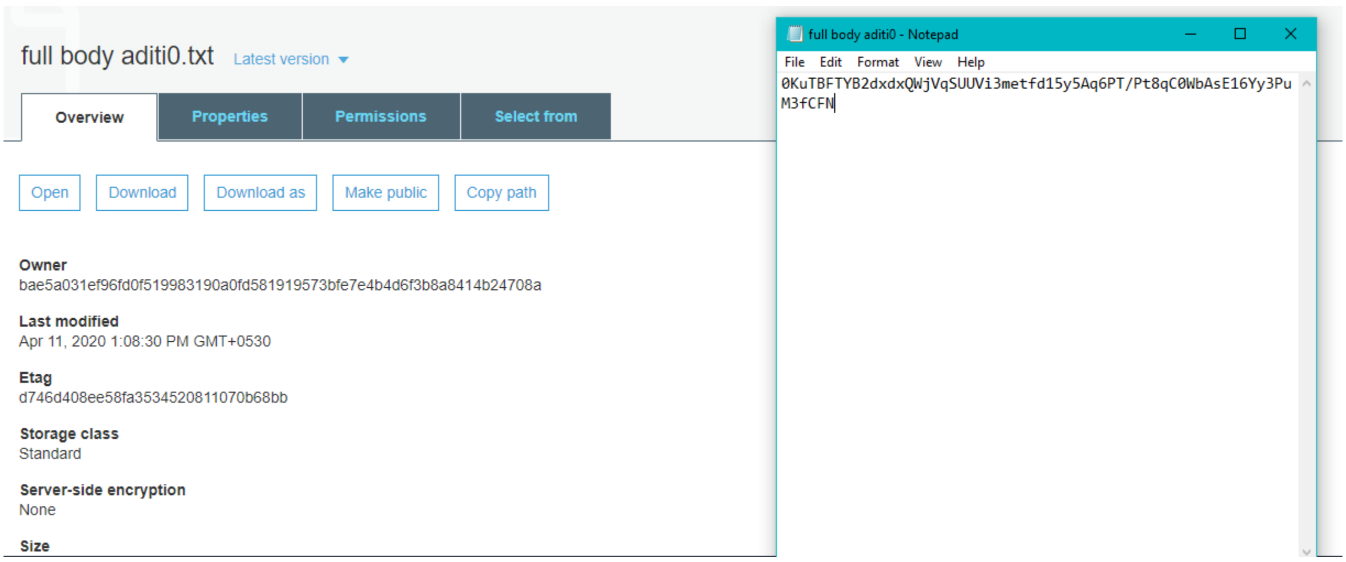
Viewing 1 to 3

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

)

Figure 5.17: Data in chunk in buckets

CHUNKS OF FILES ARE ENCRYPTED THROUGH AES



Owner
bae5a031ef96fd0f519983190a0fd581919573bfe7e4b4d6f3b8a8414b24708a

Last modified
Apr 11, 2020 1:08:30 PM GMT+0530

Etag
d746d408ee58fa3534520811070b68bb

Storage class
Standard

Server-side encryption
None

Size

```
0KuTBFTYB2dxdxQWjVqSUUVi3metfd15y5Aq6PT/Pt8qC0WbAsE16Yy3Pu
M3fCFN
```

Figure 5.18: Chunks of file are encrypted through AES

Chapter 6

Testing Results

6.1 Test Plan

- We perform functional testing to feed inputs are identify the output.It ensures that the requirements are fulfilled.

- Database testing was done to ensure proper schema,storage and retrieval of data.

- Performing tests until all test cases are passed.

Results for Functional Testing					
Test Case ID	**Test Cases**	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Redirection of Pages	Click on any button	Should redirect to the next page depending upon the action called	Gets redirected successfully	Passed
2	Orphan Pages	Check for all the action buttons	Every button must perform certain action	Every button is specified with a certain action	Passed
3	Proper data input	Enter incorrect data while registration	Should not register giving an error	User not registered	Passed

Table 6.1: Functional Testing

Results for Database Testing					
Test Case ID	**Test Cases**	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Correct Registration on credentials	Enter correct registration credentials	Successful registration	Registration Successful	Passed
2	Incorrect Registration credentials	Enter incorrect registration credentials	Unsuccessful registration	Registration unsuccessful	Passed
3	Correct login credentials	Enter correct login credentials	Successful login	Login successful	Passed
4	Incorrect login credentials	Enter incorrect login credentials	Unsuccessful login	Login unsuccessful	Passed
5	Uploading the file	Select the file to upload	File should be selected and uploaded	File selected and uploaded	Passed
6	Download files	Click on download button	File should be downloaded	File downloaded	Passed
7	Request for key	Click on request button	Message should be send to owner	Message sent to owner	Passed
8	Uploading Data into S3 buckets	Upload File through interface	File should be directly stored into S3 buckets	File gets stored into S3	Passed
9	Division of data into chunks	Uploading report	File should be stored into chunks	File is stored into chunks in random buckets	Passed
10	Encryption of chunks	Upload File	Divided chunks of file must be encrypted	Each chunk is encrypted through AES	Passed

11	Verification through OTP received in mail by the patient	Enter wrong OTP	Shouldn't be able to download file	Wrong OTP Error	Passed
12	Verification through OTP received in mail by the patient	Enter received OTP	Should be able to download file	File Downloaded	Passed

Table 6.2: Database Testing

Chapter 7

Conclusion and Future Scope

In the designed project we have implemented an Hospital Website which uses blockchain mechanism to store the reports uploaded by the patients, the UI is used to upload reports while as the storage is maintained using S3 bucket storages where the reports are stored into chunks and each chunk is encrypted using AES.

Future Scope for the system:

- Can be implemented on any kind of data that is stored on the cloud.
- Can be used to provide security to any confidential data. Can be implemented in sectors like IT, Medical, Banking , etc.
- Healthcare sector can improve by including many different kind of reports.
- Data collection can be maintained with better integrity if public blockchain is maintained to gather information from different countries during pandemics e.g:- Current situation of covid-19 is facing issues in collecting data to form pattern to overcome the pandemic.

Bibliography

[1] Nakamoto S. Bitcoin: A peer-to- peer electronic cash system[J]. Consulted, 2008.

[2] 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) Amita Kashyap, G. Sravan Kumar, Sunita Jangir, Emmanuel S. Pilli, Preeti Mishra "IHIDS: Introspection- Based Hybrid Intrusion Detection System in Cloud Environment".

[3] Watanabe, H., Fujimura, S., Nakadaira, A., Miyazaki, Y., Akutsu, A., Kishigami, J. J. (2015). Blockchain contract: A complete consensus using blockchain. 2015 IEEE 4th Global Conference on Consumer Electronics (GCCEP).

[4] Zhe, D., Qinghong, W., Naizheng, S., Yuhan, Z. (2017). Study on Data Security Policy Based on Cloud Storage. 2017 IEEE 3rd International Conference on Big Data Security on Cloud (Big Data Security), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS).

[5] Bharadwaj, D. R., Bhattacharya, A., Chakkaravarthy, M. (2018). Cloud Threat Defense – A Threat Protection and Security Compliance Solution.2018 IEEE International Conference on Cloud Computing in Emerging Markets.

Appendices

Appendix-A: Installation and Configuration of Eclipse

Step 1 : Installing Java
Before you start installing Eclipse, make sure that you have the Java Development Kit (the JDK) installed on your system.

Step 2 : Download
You can download Eclipse at eclipse.org/downloads. The latest version, as of time of writing, is Eclipse SimRel.

Step 3: Installation
Run the Eclipse installer. You should see a window like the one below; Select the first "Eclipse IDE for Java Developers" option. After that point, you can keep hitting "yes" and select all the default options (unless you want to change something). You should eventually see a screen like this. Click the "Launch" button.

Step 4: Configuration
I) When you run Eclipse, it'll ask you where you want your workspace to be.Your workspace will be the location where Eclipse will add any new projects you create.
You can change the location of the workspace if you want: just make sure you remember what you picked.
II) Once you're done, you should see a "Welcome" screen.Close the "welcome" tab to open the regular editor.
III) Next, select "Windows ¿ Preferences" (PC) or "Eclipse ¿ Preferences" (Mac) in the menu. Then, select "Java ¿ Installed JREs":
IV) Click the "Search" button and select the "Java" folder. This folder should contain your installed JRE and JDK. (If it contains only the installed JDK, that's also ok). You can probably find this folder located at:
Windows: C:Files
Mac: /Library/Java
V) After hitting "ok", you should see a screen with a line for either both the JRE and the JDK, or just the JDK. Select the line for the JDK:
VI) Click the "Apply and close" button.

Appendix-B: Installation and Configuration of Apache Tomcat

JDK or JRE will need to be installed on the Windows Server before you can configure Tomcat 9 on the server. **Installing Tomcat 9**

Step 1 :
Download Apache Tomcat from this link http://tomcat.apache.org/ click Download -¿ Tomcat 8.0 Choose Binary Distributions Core: 32-bit Windows zip / 64-bit Windows zip.

Step 2 :
Extract it to Document folder.

Step 3 :
Open Eclipse Environment Click on Servers Tab Click on No servers are available. Click this link to create a new server... Click Tomcat v8.0 Server and Next

Step 4 :
Select Apache installation Directory and click Finish.

Step 5 :
You should see Tomcat v8.0 Server at localhost [Stopped, Republish] under Servers tab.

Step 6 :
Now select the Server and click Start. Now it should be up and running on port 8080.

Appendix-C: Installation of MySQL

Step 1 : Download the MySQL Installer from dev.mysql.com The two download options are a web-community version and a full version.
i)web-community version: It will only download the server, by default, but you can select other applications (like Workbench) as desired.
ii)full installer: It will download the server and all the recommended additional applications.

Step 2 : Run the installer that you downloaded from its location on your server, generally by double-clicking.

Step 3 : Determine which setup type you would like to use for the installation:
i)Developer Default: This is the full installation of MySQL Server and the other tools needed for development. If you are building your database from the ground up or will be managing the data directly in the database, you'll want to use this setup type.
ii)Server Only: If you only need MySQL Server installed for use with a CMS or other application and will not be managing the database directly, you can install just the server.
iii)Custom: This setup type will allow you to customize every part of the installation from the server version to whichever additional tools you select.

Step 4 : Install the server instance and whichever additional products you selected.Then begin the configuration process by selecting the availability level (most users will use the default, standalone version).

Step 5 : Complete the configuration process by following the on-screen instructions. You'll want to make sure to install MySQL as a Service so that Windows can automatically start the service after a reboot or can restart the service if it fails.

Configuration of MySQL

Step 1 : Goto Database Perspective Window-¿ Perspective-¿Open Perspective-¿ Other

Step 2 : Select Database Development Perspective select the Database Development perspective and click on the Open button.

Step 3 : Create Database Connections **3.1 :** There you can see the Database Connections folder, right click and click on the New.
3.2 : Then you will see the New Connection Profile window as below; there you will find all available connections, filter the connection profile type with "MySQL" string and select the MySQL connection.Name this connection as MYSQLDBCONNECTION and click on Next button.
3.3 : This window allows you to attach the driver details to connect with the MySQL DB.Click on the below-highlighted icon to attach MySQL connector.
3.4 : Then you will see the following window;select the MySQL version which you wanted to

connect with and move to JAR List tab.Here we have to attach the MySQL connector.

Step 4 : Download and attach MySQL connector
4.1 : Download the MySQL connector from **https://downloads.mysql.com/archives/cj/**
4.2 : It will be downloaded as a .zip file. Extract the zip and there you will see the mysql-connector-java-xxx.jar. Add this jar file in Jar List tab by clicking on the Add JAR button and click on the Ok button. **4.3 :** Specify the MySQL connection details like username, password and database name and click on the Test Connection button.
4.4 : If everything configured well, you would see the below Ping succeeded! Message. Then Click on the Finish button.
Done.

Step 5 : Verify Again go to Data Source Explorer and refresh on Database Connections folder and expand the folders inside it.
You would see the MySQL database schema here.

Acknowledgement

We have great pleasure in presenting the report on **Enhancing Data Security in Cloud using Blockchain.** We take this opportunity to express our sincere thanks towards our guide **Prof. Yamini Patil** & Co-Guide **Prof. Sneha Kanchan** Department of IT, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Prof. Kiran B. Deshpande** Head of Department,IT, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Prof. Vishal S. Badgujar** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Dhananjay Yadav

17204015

Aditi Shinde

16104022

Akash Nair

16104051

Publication

Paper entitled **"Enhancing Data Security in Cloud using Blockchain"** is presented at **"International Conference of Intelligent Computing and Control System"** by **"Dhananjay Yadav, Aditi Shinde, Akash Nair".**