SCHOOL OF COMPUTER SCIENCE
UNIVERSITY OF PETROLEUM & ENERGY STUDIES
Bidholi Campus, Energy Acres, Dehradun – 248007.

January-May 2021

END SEMESTER REPORT

on

NotTy- Notes with added Security

Submitted by

Kashish Srivastava

(Enroll No. R134218079 & Sap id 500067405)

Yash Kumar

(Enroll. No. R134218192 & Sap id 500068736)

Vishal Patwal

(Enroll. No. R134218189 & Sap id 500068529)

Abhishek Gupta

(Enroll. No. R134218008 & Sap id 500067804)

Under the guidance of

Ms. Tripti Mishra

Department of Systemics

CONTENTS

School of Computer Science

University of Petroleum & Energy Studies, Dehradun

Project End Sem Report

Minor II

PROJECT TITLE

Protect your Notes through encryption and authentication via **NotTy- Notes with added Security**.

ABSTRACT

This project aims at taking a step to improve data privacy and maintain security in personal or confidential data of an individual. Failure of protection of confidential data may unlock the danger of information being used for illegal or explicit activities irrespective of the owner's will. Development of an application in this project would help in protecting as well as securing data via two-factor authentication. "NotTy - Notes Security" would be beneficial in securing the data, maintain its integrity and check through authentication and encryption algorithms. For the further implementation, it can be made beyond a local system requirement by connecting it to cloud.

KEYWORDS

Notes, Security, Protection, Two-Factor Authentication, Integrity, Confidential Data, Storage

INTRODUCTION

Data are qualities or information, generally numeric, that is gathered through observation. In a more specialized sense, information is a set of estimations of subjective or quantitative factors around at least one people or items. This data might be as text archives, pictures, sound bites, programming programs, or different kinds of information. Security and confidentiality of information are identified with one another. The principal contrast between security and confidentiality of information is that information security guards information from loss, damage and from unauthorized clients, while confidentiality permits only the authorized clients to get access to the data. Data Confidentiality is tied in with securing information against inadvertent, unlawful, or unapproved access, exposure, or robbery.

Confidentiality has to do with the protection of data, including authorization to view, edit, use, and share information. Data with low confidentiality concerns might be considered "public" or can

cause compromising access to unauthorized users. Data with high privacy concern is viewed as secret and should be kept private to forestall fraud, bargain of records and frameworks, lawful or reputational harm, and other severe consequences. These days clients store their personal information in notes for brisk access but they are quite prone to be exploited. Here comes NotTy (Notes Security) application with two-factor authentication for enhanced security. It generates an OTP everytime the user opens NotTy along with OTP authentication and hashing will be applied for additional security.

The synopsis is organized as follows: Section two is the problem statement for the project in reference to securing data and application. Section three is about Literature Review based on various data privacy and encryption insights. Section four is about the Objectives of the development of the application. Section five is dependent on the Methodology of the project. Section six initiates the System requirements and Schedule for the java application. Section seven illustrates the design and implementation of this project. Last section introduces References for the system.

PROBLEM STATEMENT

These days people generally tend to write any information or reminders in the form of notes. Sometimes sensitive information such as login credentials are also stored on such applications. Most of the notes applications present on the market offer little to no sense of security to the data stored in them.

This project aims to develop a text editor with simplistic features where the data stored can be verified for its integrity using SHA256 algorithm. It also aims to provide with two-factor user authentication viz. One Time Password received on the user's phone and a pin code so that only authorized users can access the application.

LITERATURE REVIEW

Title	Year/Link	Author	Remarks
Predicting user concerns about online privacy	https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.20530	Mike Z. Yao Ronald E. Rice Kier Wallis	Study aims at developing a model using generalized skills of a user, their psychological needs for privacy, internet usage experience, fluency in using the internet and their point of view on privacy rights. The results showed that beliefs in privacy rights and a psychological need for privacy were the main influences on online privacy concerns.

A comparative study of Message Digest 5(MD5) and SHA256 algorithm	https://iopscience.iop.org/article/10.1088/1742-6596/978/1/012116/meta	D Rachmawati, J T Tarigan and A B C Ginting	This study aims to do a comparaitive analysis of the two hashing algorithms, i.e.,MD5 and SHA256 mainly on the basis of their complexity and speed. These algorithms are used to verify the integrity of a document. To ensure the integrity of a document a digital signature. One of the type of methods to generate digital signatures is the use of hash functions. Message Digest 5 and SHA256 are two such algorithms of Hash Functions.
Encrypting Data at Rest	https://d1.awsstatic.com/whitepapers/AWS_Securing_Data_at_Rest_with_Encryption.c1b42b665c0693053ff13b581e673b13a2818816.pdf	Ken Beer Ryan Holland	An overview of different methods for encrypting of data at rest available today using AWS.
IEEE standards for encrypted storage	https://ieeexplore.ieee.org/abstract/document/1362602	J. Hughes	Mishandling of data at rest is a major factor in data being exposed. To address this the IEEE Computer Society's Security in Storage Working Group (http://siswg.org) is developing Project 1619, Standard Architecture for Encrypted Shared Storage Media. It's aim is to enable the interoperable encryption of storage devices, provide methods for information owners to transfer key material, and facilitate product certification. The standard currently includes encryption at the physical level of disk and tape drives, and future efforts will extend it to objects and file systems.

OBJECTIVES

Working on creating the java application which would keep the confidential data safe as well as integral in nature. Implementation of two-factor authentication along with encryption algorithm would be the most important aspect for the following.

Sub Objectives

- Text editor application in Java
- Implementation of two-factor authentication i.e. Password inquiry and One Time Password inquiry
- Implementing the concepts of hashing to maintain the data integrity and check for breaches if any.
- Working on the encryption algorithm to provide the user a secure environment.

METHODOLOGY

For this project, the major focus is on developing a secure application for the user for storing their confidential data which would be secured via two-factor authentication and hashing.

The entire implementation of this project can be explained in the following points:

1. The application has to be developed as in for opening, appending and reading the notes via file handling concept.
2. The application after basic development of text editor has to be followed by 2FA implementation.
3. The first implementation for authentication is to allow the user to set a password for the user.
4. The second implementation for authentication is of the OTP concept.
5. For the notes and confidential data, integrity has to be checked of the data stored with the help of hashing concepts.
6. At the end for a secure environment an encryption algorithm can be implemented.

SYSTEM REQUIREMENTS

Hardware:

- Laptop with i3/i5/i7 processor with 8Gb Ram, 1 Tb Hard-disk

Language:

- Java

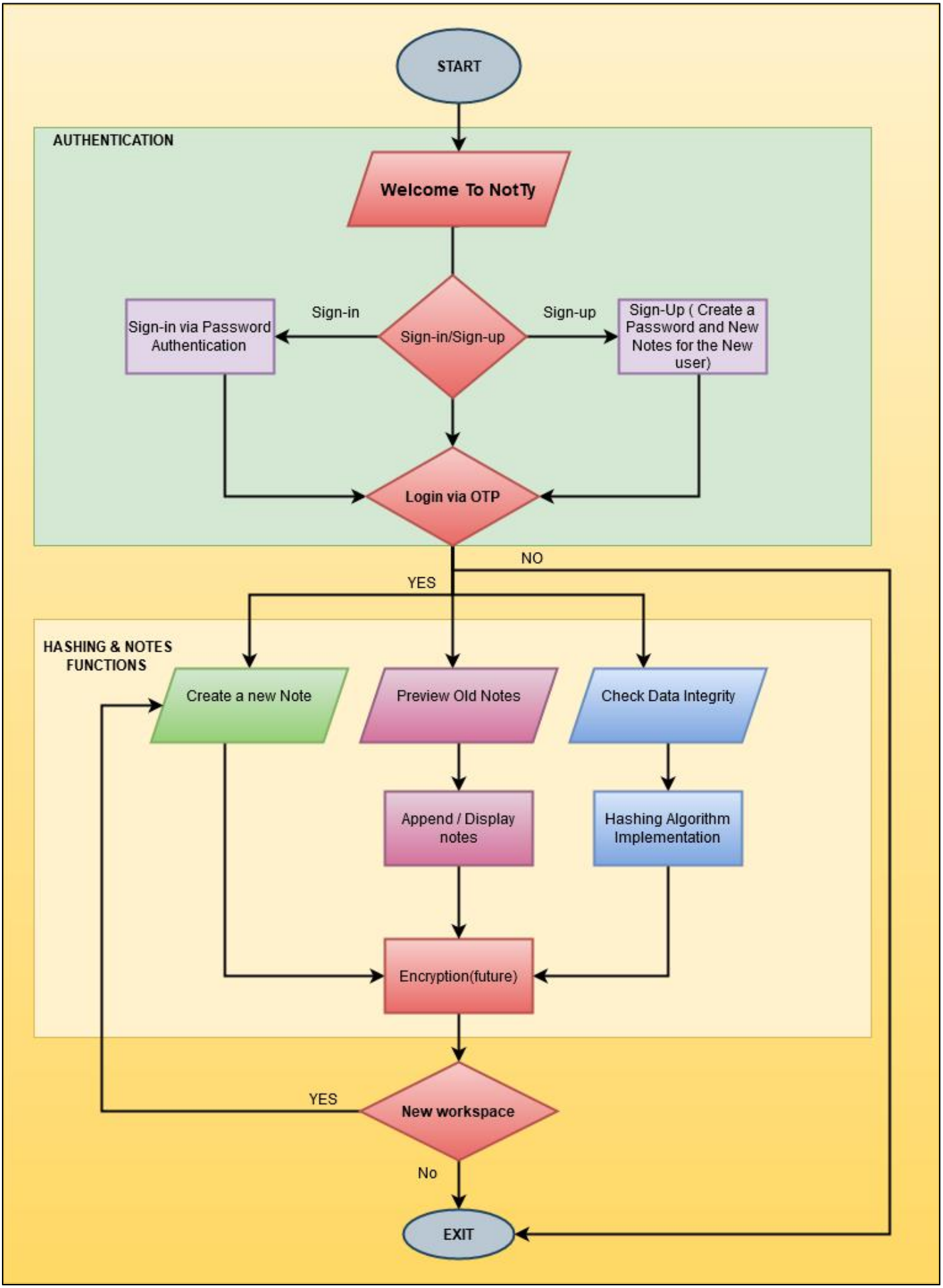
Operating System:

- Ubuntu/ Linux

Resources:

- Packages in Java

DESIGN AND ALGORITHM



1. Algorithm (Reading Notes)

- Step1- Asks user for input r(read), w(write), a(append)
- Step2- If ‘a’ input from user read file function is called.
- Step3- If ‘w’ or ‘a’ input from user write file function is called.
- Step4- Else it displays unexpected input.

```
   String sx=sc.nextLine();
if(sx.equalsIgnoreCase("r"))
{
    new FReading();
}
else if(sx.equalsIgnoreCase("w")||sx.equalsIgnoreCase("a"))
{
    writingToFile(sx);
}
else
{
    System.out.println("Sorry you try to do unexpected ,betterluck next time ");
}
sc.close();
```

2. Algorithm(Eriting Notes)

- Step1- User enter ‘w’ to write to a file
- Step2- If file exists, warns about overwriting
- Step3- If no, exit
- Step4- If yes, proceed
- Step5- User writes to the file
- Step6- User enters ‘Stop’ to finish writing

```
BufferedReader bf=new BufferedReader(new InputStreamReader(System.in));
//For writing new Content Everytime you run
FileWriter f0 =null;
if(s.equalsIgnoreCase("w"))
{
        f0 = new FileWriter(f,false);
        System.out.println("CAUTION >> Please understand it will overwrite the content of the file ");
        System.out.println("Type 'no' to exit");
        System.out.println("Do you want to proceed :type 'yes' ");
        in=new Scanner(System.in);
        String sl=in.nextLine();
        if(sl.equals("no"))
        System.exit(0);
        System.out.println("Write 'stop' when you finish writing file ");
        //f.delete(); //optional karna h
        f.createNewFile();
        while(!(source=bf.readLine()).equalsIgnoreCase("stop")){
            f0.write(source + System.getProperty("line.separator"));

   }

        in.close();
```

3. Algorithm(Appending Notes)

- Step1- User enter appending mode
- Step2- User enter data
- Step3- User enters ‘Stop’ to finish appending file

```
//For appending the content
    else
        {
    f0 = new FileWriter(f,true);
            System.out.println("Write 'stop' when you finish appending file ");
            while(!(source=bf.readLine()).equalsIgnoreCase("stop")){
            f0.append(source+ System.getProperty("line.separator"));
        }

   f0.close();
```

4. Algorithm

- Step1- User enter reading mode
- Step2- User enters file name which he wants to read
- Step3- If file exits then user reads file
- Step4- If file doesn't exist, error displayed.

```
public FReading() {

    try{
        Scanner s3=new Scanner(System.in);
        System.out.println("Enter file name u want to read");
        String fq=s3.nextLine();
        File f5=new File(fq);         //choose option which file to be read
        if(! f5.exists())
        f5.createNewFile();
        FileReader fl=new FileReader(f5);
        BufferedReader bf=new BufferedReader(fl);
        //For reading till end
        while((str=bf.readLine())!=null){
            System.out.println(str);
        }
        fl.close();
        }catch(Exception e){
        System.out.println("Error : " );
        e.printStackTrace();
    }
}
```

5. Algorithm(MD5 Hashing for Data Integrity)
    - Step1- All the contents of the file are passed on to the string
    - Step2- The hash that has to be generated will be of lowecase 32 characters
    - Step3- BigInteger helps in mathematical operation which involves very big integer calculations that are outside the limit
    - Step4- Message Digest for MD5

```
switch(ch)
        {
            case 1:try{
                        String password = bf.readLine();
                         String hash= String.format("%032x", // produces lower case 32 char wide hexa left-padded with 0
                         new BigInteger(1, // handles large POSITIVE numbers
                         MessageDigest.getInstance("MD5").digest(password.getBytes())));
                         System.out.println("Here is the hash for your file for integrity check  : " + hash);
                         }
                catch (NoSuchAlgorithmException e) {
                e.printStackTrace();// do whatever seems relevant
                }
            case 2: break;
            default: System.out.println("Thannkyou, you will exit from the editor" );
```

RESULTS

- Reading Mode- A user can view any pre-created file in this mode but cannot edit its components.
- Writing Mode- A user can create or edit the contents of the file in this mode but the changes will be overwritten means the previous data will not be saved.
- Append Mode- A user can create or edit the contents of the file in this mode and previous data will not be erased from the memory.
- It will show an error every time user requests an operation other than read, write or append.
- Password Creation- A new user is able to set a password on the application to secure their data. Every time user opens the application they have to prove their authenticity by entering correct password.
- An existing user can directly select a mode of operation they want to perform with their file. They don’t need to create a new password every time they enter.
- OTP based Two Factor Authentication- A user after entering password has to enter the code generated by the authenticator two successfully login.
- Data Integrity – MD5 hashing implementation to maintain the hashing of the algorithm.

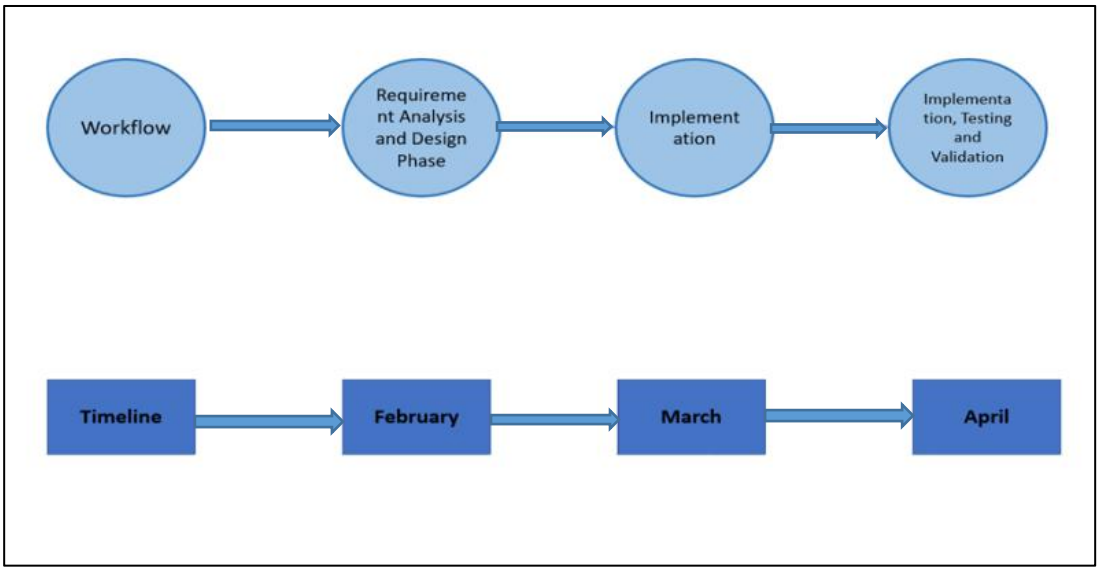
CLI based notes editor with 2-factor-authentication and Data Integrity

```
|@@@|||||||||||||||||||||||||||||||||||||||||||||@@@|
!!! Welcome to NotTy: Notes with Added Security !!!
|@@@|||||||||||||||||||||||||||||||||||||||||||||@@@|

If you are a new user press 1 else if you are an existing user press 2
2
Enter your password
K@sh

*****Authenticate via One-Time-Password*****
Enter the OTP provided by our authenticator
0057
!** Successfullly Logged-In your account! **!

Please select one of the below operations
 w for write mode
 r for read mode
 a for append mode
r
Enter file name u want to read
Kashish.txt
Part 2
Do you also want to check the integrity of the file you are reading? If yes press 1 else press 2
1
Enter file name u want to read
Kashish.txt
Here is the hash for your file for integrity check  : 6417c68d02736b9d6bbccd60d0e3bd39
```

CONCLUSION

Through this application, end-users will get a platform where no intruders can access their data which is updated on NotTy without their consent as it is secured with two-factor authentication. For the future scope we would prefer shifting the prototype to azure cloud for better scalability and provide encryption implementation for passwords and maintaining high security standards.

SCHEDULE (pert chart)



REFERENCES

[1] https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.20530

[2] https://iopscience.iop.org/article/10.1088/1742-6596/978/1/012116/meta

[3] https://d1.awsstatic.com/whitepapers/AWS_Securing_Data_at_Rest_with_Encryption.c1b42b665c0693053ff13b581e673b13a2818816.pdf

[4] https://ieeexplore.ieee.org/abstract/document/1362602

[5] https://medium.com/swlh/simple-notes-app-in-android-java-9062d7bb3bc0

[6] https://www.researchgate.net/publication/235788474_Java_technology_in_the_design_and_implementation_of_web_applications

Synopsis Draft verified by

Project Guide
(Name & Sign)

HOD
(Dept. of Systemics)