

HASHY ANDROID APPLICATION

Submitted in partial fulfillment of the requirements
of the degree

**BACHELOR OF ENGINEERING IN
COMPUTER ENGINEERING**

By

**Danyl Fernandes
Mathew Philip
Gandharv More**

Supervisor

Prof. Kirti Motwani



**Department of Computer
Engineering Xavier Institute of
Engineering, Mahim, Mumbai - 400 016
University of Mumbai**

CERTIFICATE

(AY2020-21)

This is to certify that the Mini Project entitled “**Hashy**” is a bonafide work of **Danyl Fernandes - 2020012004(72)** **Gandharv More - 201901035(24)** **Philip Mathew - 201901030(44)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of “**Bachelor of Engineering**” in “**Computer Engineering**”.

(Prof.)
Supervisor

(Prof.)
Head of Department

(Prof.)
Principal

Mini Project Approval

This Mini Project entitled "Hashy"
by **Danyl Fernandes - 2020012004(72)** **Gandharv More - 201901035(24)**
Philip Mathew - 201901030(44) is approved for the degree of
Bachelor of Engineering in Computer Engineering.

Examiners

1.....
(Internal Examiner Name & Sign)

2.....
(External Examiner name & Sign)

Date:

Place:

Contents

Abstract	4
Acknowledgments	5
1 Introduction	6
1.1 Introduction	
1.2 Motivation	
1.3 Problem Statement & Objectives	
1.4 Organization of the Report	
2 Literature Survey	10
2.1 Survey of Existing System	
2.2 Limitations of Existing system or research gap	
2.3 Mini Project Contribution	
3 Proposed System (e.g. New Approach of Data Summarization)	11
3.1 Introduction	
3.2 Architecture/Framework	
3.3 Algorithm and Process Design	
3.4 Details of Hardware &Software	
3.4 Experiment and Results	
3.5 Conclusion and Future work.	
References	18

ABSTRACT

In cryptography, **MD5** (*Message Digest version 5*) and **SHA** (*Secure Hash Algorithm*) are two well-known message digest algorithms. They are also referred as cryptographic hash functions, which take arbitrary-sized data as input (message) and produce a fixed-length hash value. One of the most important properties of hash functions is, it's infeasible to generate a message that has a given hash (secure one-way). Hash functions are frequently used to check data integrity such as checking integrity of a downloaded file against its publicly known hash value. Another common usage is to encrypt user's password in database.

The Java platform provides two implementation of hashing functions: MD5 (produces 128-bit hash value), SHA-1 (160-bit) and SHA-2 (256-bit). This tutorial demonstrates how to generate MD5 and SHA hash values from String or file using Java.

Message Digest Algorithm 5 (MD5) is a cryptographic hash algorithm that can be used to create a 128-bit string value from an arbitrary length string. Although there has been insecurities identified with MD5, it is still widely used. MD5 is most commonly used to verify the integrity of files. However, it is also used in other security protocols and applications such as SSH, SSL, and IPsec. Some applications strengthen the MD5 algorithm by adding a salt value to the plaintext or by applying the hash function multiple times.

ACKNOWLEDGEMENTS

Apart from the efforts of the group, the success of our project depends largely on the encouragement and guidelines of many others. We would like to take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. We would like to show my greatest appreciation to Prof. Kirti Motwani for their tremendous support and help. We feel motivated and encouraged by our professors to select an innovative idea for the betterment of the society. Without their encouragement and guidance this project would not have materialized. We would also like to thank our H.O.D Prof. Dr. Saurabh Patil for his support and motivation. The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. I am grateful for their constant support and help.

INTRODUCTION:

1.1 INTRODUCTION

A cryptographic hash function is a hash function. It takes an arbitrary block of input string and returns a fixed-size bit of output string. The cryptographic hash values differ such that any accidental or intentional change to the data. The data to be encoded are often called the message and the hash value is called the message digest. The SHA Algorithm is used in digital certificate as well as in data integrity and message authentication. SHA is a fingerprint that specifies the data and was developed by N.I.S.T. as a U.S. Federal Information Processing Standard (FIPS), is intended for use with digital signature applications .

In order to secure a network or system, a basic understanding of the low-level aspects of systems must be achieved. Security Architecture and Design, which details low-level aspects of operating systems and hardware as they relate to security. Key design principles such as layering, abstraction, the concept of security domains, and the ring model architecture are discussed. Familiarity with the operation of a CPU and memory.

As a wide use of internet day by day it is needed that a proper file has been download from peer to peer (P2P) servers and network. Due to present of same name file it is quite difficult to find the original so message digest plays an important role in such type of downloads. These type of file may be bound with message authentication code which proves that the source is verified otherwise it shows the warning that verified source not found or vice versa

1.2 MOTIVATION

We were exploring various topics for building an application. As we know Hashing System has various applications in our day-to-day life. We wanted to build something using technology and explore various aspects of hashing algorithms. We explored different sources and decided to build up an application to highlight this aspect and gain the knowledge in this domain. We explored various research papers and technologies to find out the previously done research in this field. We chose this project to enhance and upgrade our knowledge.

1.3 PROBLEM STATEMENT:

Nowadays, with enhancing technology we do not have applications which can secure the data and not leak out the personal details hence we can use hashing to provide and message authentication and can also be used to check if the message has been modified or not. In addition to this digital key which use hashing algorithms could be used with private key.

Password storage can be done with such applications and hence lead to a more secure framework and could make it secure and away from the reach of hackers.

This application could be used in key generation meaning the key can be generated from digest of passphrase; can be made computationally expensive to prevent brute-force attacks. It generates Pseudorandom number and is iterated using the hashing of a seed value.

This application is used for Intrusion detection and virus detection, hence keeps a check on the hash of files on system.

OBJECTIVES OF THE SYSTEM:

- Message authentication: used to check if a message has been modified.
- Digital signatures: encrypt digest with private key.
- Password storage: digest of password is compared with that in the storage; hackers cannot get password from storage •
- Key generation: key can be generated from digest of pass-phrase; can be made computationally expensive to prevent brute-force attacks.
- Pseudorandom number generation: iterated hashing of a seed value. • Intrusion detection and virus detection: keep and check hash of files on system
- Message Digest
- Password Verification
- Data Structures (Programming Languages)
- Compiler Operation
- Rabin-Karp Algorithm
- Linking File name and path together

1.4 ORGANISATION OF THE REPORT

The purpose of the report is to give a detailed analysis of the project, its advantages, uses and details of implementation of the project.

The standard components of the paper are as follows:

1 Introduction	9
1.1 Introduction	
1.2 Motivation	
1.3 Problem Statement & Objectives	
1.4 Organization of the Report	
2 Literature Survey	10
2.1 Survey of Existing System	
2.2 Limitation Existing system or research gap	
2.3 Mini Project Contribution	
3 Proposed System	11
3.1 Introduction	
3.2 Architecture/ Framework	
3.3 Algorithm and Process Design	
3.4 Details of Hardware & Software	
3.4 Experiment and Results	
3.5 Conclusion and Future work.	
References	18

LITERATURE SURVEY:

2.1 SURVEY OF THE EXISTING SYSTEM:

Under literature survey we reviewed research work done by various professors, we checked a paper published on “A secured Cryptographic Hashing Algorithm” written by Prof. Rakesh Mohanty, Niharjyoti Sarangi, Sukant Kumar Bishi in which Cryptographic hash functions for calculating the message digest of a message has been in practical use as an effective measure to maintain message integrity since a few decades. This message digest is unique, irreversible and avoids all types of collisions for any given input string. The message digest calculated from this algorithm is propagated in the communication medium along with the original message from the sender side and on the receiver side integrity of the message can be verified by recalculating the message digest of the received message and comparing the two digest values.

So, in our project we implemented the basic idea from the research paper have modified their algorithm for calculating the message digest of any message and implemented it using a high level programming language. An experimental analysis and comparison with the existing MD5 hashing algorithm, which is predominantly being used as a cryptographic hashing tool, shows this algorithm to provide more randomness and greater strength from intrusion attacks. In this algorithm the plaintext message string is converted into binary string and fragmented into blocks of 128 bits after being padded with user defined padding bits. Then using a pseudo random number generator, a key is generated for each block and operated with the respective block by a bitwise operator. This process is iterated for the whole message and finally a fixed length message digest is obtained.

Another similar paper reviewed by us was, “Cryptographic Hash Functions: A Review” published by Rajeev Sobti, G.Geetha which has an extensive research on Cryptographic Hash functions are used to achieve a number of security objectives. In this paper, we bring out the importance of hash functions, its various structures, design techniques, attacks and the progressive recent development in this field. We used the same concept and modified it as per our requirements.

2.2 Limitations of Existing system or research gap

Some of the limitations that have been identified with the designs of existing hashing applications are listed below:

1. Most of the hashing applications that have been developed don't produce very accurate results.
2. Generating a hash for a given string on-the-fly has been a challenge.
3. Web applications aren't as fast a solution and mobile applications for the same have not existed before
4. SHA was not seen in as a hashing option in the surveyed applications.

2.3 Mini Project Contribution

When we were exploring various ideas over the internet, an innovative idea caught our attention that could not only work as a project but could also be an amazing source of knowledge.

We referred various android application project sites and the website caught our attention. We surveyed the various existing hashing algorithms and came across the multiple advantages and a few disadvantages which could be overcome using different hashing algorithms i.e. SHA1 and SHA256 and MD5 and found out the system that could be used as an extension various messaging applications and attack prone games or other applications for a wide range of audience and acts as a great utility. There were a few noticeable features we pointed out and chose to work on them to create a system that could be a a very useful source and can be used in variety of ways.

Proposed System

(e.g. New Approach of Data Summarization):

3.1 Introduction

Cryptographic hash functions for calculating the message digest of a message has been in practical use as an effective measure to maintain message integrity since a few decades. This message digest is unique, irreversible and avoids all types of collisions for any given input string. The message digest calculated from this algorithm is propagated in the communication medium along with the original message from the sender side and on the receiver side integrity of the message can be verified by recalculating the message digest of the received message and comparing the two digest values. In this paper we have designed and developed a new algorithm for calculating the message digest of any message and implemented it using a high level programming language. An experimental analysis and comparison with the existing MD5 hashing algorithm, which is predominantly being used as a cryptographic hashing tool, shows this algorithm to provide more randomness and greater strength from intrusion attacks. In this algorithm the plaintext message string is converted into binary string and fragmented into blocks of 128 bits after being padded with user defined padding bits. Then using a pseudo random number generator a key is generated for each block and operated with the respective block by a bitwise operator. This process is iterated for the whole message and finally a fixed length message digest is obtained.

3.2 Architecture/ Framework

We used a ConstraintLayout as the root layout to house the UI Components of the main screen of the application. We used EditTexts inside TextInputLayouts to receive user input. The TextInputLayouts are constrained to the side of the root view also known as the parent view.

There is one TextInputEditText to receive the input string from the user. There is a dropdown list designed in XML using a custom adapter that has predefined options of hashing algorithms.

We designed a dropdown list to handle different hashing options available within the app. We used a TextView to display the generated hash and added a click listener on the TextView such that upon clicking the text the hash is copied to the clipboard.

3.3 Algorithm and Process Design:

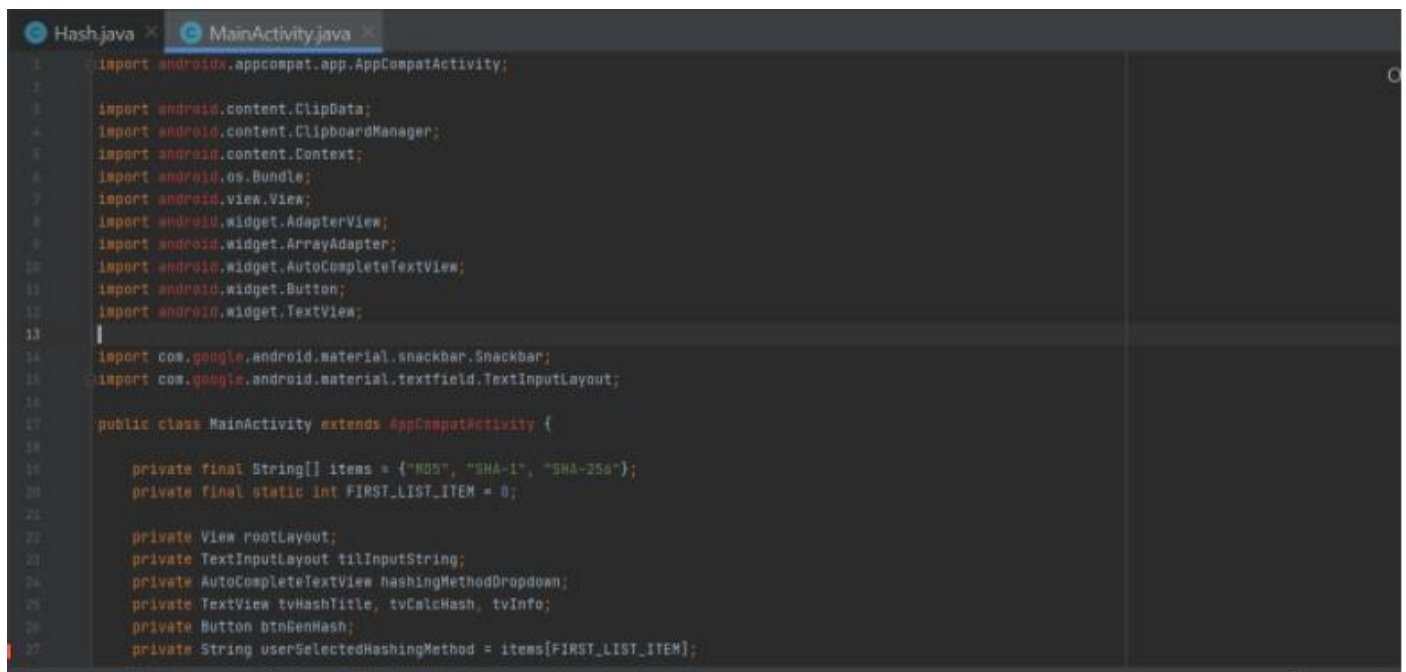
Source Code:

MainActivity.java is the entry point to the Android Application.

- The callback onCreate() is fired when the Activity (screen) is first created
- We call setupUi() from within onCreate() , that instantiates variables that store a reference to their respective UI elements
- After Initialisation, a clickListener is setup on the generate button that listens to clicks on the button
- When a click is registered, a callback to onClick() is fired which in turn calls HashMessage.getEncryptedMessage()
- A string representation of the hash is returned which is then displayed in the TextView on screen

Hash.java

- This is the utility class that provides the `getEncryptedMessage()` method that returns a string representation of the generated hash.
- The `HashMessage` class uses Java's in-built `MessageDigest` class to generate the required hashes.
- The `MessageDigest`'s factory method `instance()` is called and a string of the Hashing Algorithm to use is passed to it.
- A switch case switches between the three Hashing Algorithm options that the app provides
- This value of the hash is parsed as a s



```
1 import androidx.appcompat.app.AppCompatActivity;
2
3 import android.content.ClipData;
4 import android.content.ClipboardManager;
5 import android.content.Context;
6 import android.os.Bundle;
7 import android.view.View;
8 import android.widget.AdapterView;
9 import android.widget.AdapterView.OnItemClickListener;
10 import android.widget.AutoCompleteTextView;
11 import android.widget.Button;
12 import android.widget.TextView;
13
14 import com.google.android.material.snackbar.Snackbar;
15 import com.google.android.material.textfield.TextInputLayout;
16
17 public class MainActivity extends AppCompatActivity {
18
19     private final String[] items = {"MD5", "SHA-1", "SHA-256"};
20     private final static int FIRST_LIST_ITEM = 0;
21
22     private View rootLayout;
23     private TextInputLayout tilInputString;
24     private AutoCompleteTextView hashingMethodDropdown;
25     private TextView tvHashTitle, tvCalcHash, tvInfo;
26     private Button btnGenHash;
27     private String userSelectedHashingMethod = items[FIRST_LIST_ITEM];
```

```
Hash.java x MainActivity.java x
28
29 @Override
30 protected void onCreate(Bundle savedInstanceState) {
31     super.onCreate(savedInstanceState);
32     setContentView(R.layout.activity_main);
33
34     setupUI();
35 }
36
37 private void setupUI() {
38     rootLayout = findViewById(R.id.rootLayout);
39     tilInputString = findViewById(R.id.til_input_string);
40     tvCalcHash = findViewById(R.id.tv_calc_hash);
41     btnGenHash = findViewById(R.id.btn_gen_hash);
42     tvHashTitle = findViewById(R.id.tv_hash_title);
43     tvInfo = findViewById(R.id.tv_info);
44
45     setupHashingMethodDropdown();
46     handleGenButtonClick();
47
48     hideHashTextViews();
49     setupGenHashTextView();
50 }
51
52 private void setupHashingMethodDropdown() {
53     ArrayAdapter arrayAdapter = new ArrayAdapter<>(getApplicationContext(), R.layout.list_item, items);
54 }
```

```
Hash.java x MainActivity.java x
54
55 private void setupHashingMethodDropdown() {
56     ArrayAdapter arrayAdapter = new ArrayAdapter<>(getApplicationContext(), R.layout.list_item, items);
57
58     hashingMethodDropdown = findViewById(R.id.actv_hashing_method);
59     hashingMethodDropdown.setAdapter(arrayAdapter);
60     hashingMethodDropdown.setText(arrayAdapter.getItem(FIRST_LIST_ITEM).toString(), false);
61
62     hashingMethodDropdown.setOnItemClickListener(new AdapterView.OnItemClickListener() {
63         @Override
64         public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
65             userSelectedHashingMethod = items[position];
66         }
67     });
68 }
69
70 private String retrieveUserTypedString() {
71     return tilInputString.getEditText().getText().toString();
72 }
73
74 private void validateInput(String userTypedInputString) {
75     if (userTypedInputString.isEmpty()) {
76         handleEmptyInput();
77     } else {
78         generateAndDisplayMash(userTypedInputString);
79     }
80 }
```



```

Hash.java x MainActivity.java x
80 private void generateAndDisplayHash(String userTypedInputString) {
81     String generatedHash = Hash.getEncryptedMessage(userSelectedHashingMethod, userTypedInputString);
82
83     showHashTextViews();
84     tvHashTitle.setText(userSelectedHashingMethod);
85     tvCalcHash.setText(generatedHash);
86 }
87
88 private void handleGenButtonClick() {
89     btnGenHash.setOnClickListener(new View.OnClickListener() {
90         @Override
91         public void onClick(View v) {
92
93             if (tilInputString.isErrorEnabled()) {
94                 tilInputString.setErrorEnabled(false);
95                 validateInput(retrieveUserTypedString());
96             } else {
97                 validateInput(retrieveUserTypedString());
98             }
99         }
100     });
101 }
102
103 private void handleEmptyInput() {
104     tilInputString.setError(getString(R.string.til_empty_input_error));
105 }
106

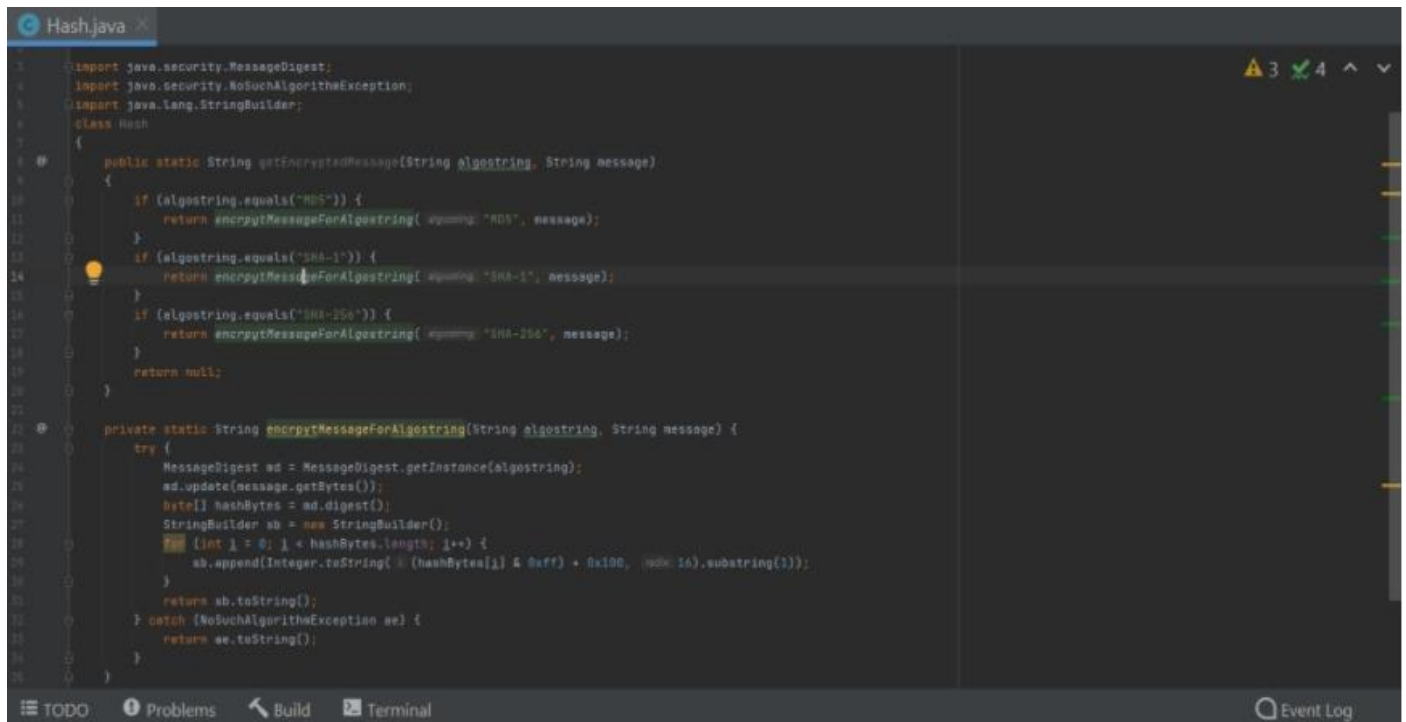
```

```

Hash.java x MainActivity.java x
106 private void showHashTextViews() {
107     tvInfo.setVisibility(View.VISIBLE);
108     tvHashTitle.setVisibility(View.VISIBLE);
109     tvCalcHash.setVisibility(View.VISIBLE);
110 }
111
112 private void hideHashTextViews() {
113     tvInfo.setVisibility(View.GONE);
114     tvHashTitle.setVisibility(View.GONE);
115     tvCalcHash.setVisibility(View.GONE);
116 }
117
118 private void setupGenHashTextView() {
119     tvCalcHash.setOnClickListener(new View.OnClickListener() {
120         @Override
121         public void onClick(View v) {
122             copyTextToClipboard(tvCalcHash.getText().toString());
123             displayCopySuccess();
124         }
125     });
126 }
127
128 private void copyTextToClipboard(String textToCopy) {
129     ClipboardManager clipboard = (ClipboardManager) getSystemService(Context.CLIPBOARD_SERVICE);
130     ClipData clip = ClipData.newPlainText("Hashed Message", textToCopy);
131     clipboard.setPrimaryClip(clip);
132 }

```

Hash.java



```
1 import java.security.MessageDigest;
2 import java.security.NoSuchAlgorithmException;
3 import java.lang.StringBuilder;
4 class Hash
5 {
6     public static String getEncryptedMessage(String algostring, String message)
7     {
8         if (algostring.equals("MD5")) {
9             return encryptMessageForAlgostring("MD5", message);
10        }
11        if (algostring.equals("SHA-1")) {
12            return encryptMessageForAlgostring("SHA-1", message);
13        }
14        if (algostring.equals("SHA-256")) {
15            return encryptMessageForAlgostring("SHA-256", message);
16        }
17        return null;
18    }
19
20    private static String encryptMessageForAlgostring(String algostring, String message) {
21        try {
22            MessageDigest md = MessageDigest.getInstance(algostring);
23            md.update(message.getBytes());
24            byte[] hashBytes = md.digest();
25            StringBuilder sb = new StringBuilder();
26            for (int i = 0; i < hashBytes.length; i++) {
27                sb.append(Integer.toString((hashBytes[i] & 0xff) + 0x100, 16).substring(1));
28            }
29            return sb.toString();
30        } catch (NoSuchAlgorithmException ae) {
31            return ae.toString();
32        }
33    }
34 }
```

3.4 Hardware and Software requirements:

Software Requirements:

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools.

To support application development within the Android operating system, Android Studio uses a Gradle-based build system, emulator, code templates, and Github integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules.

Technology Used:

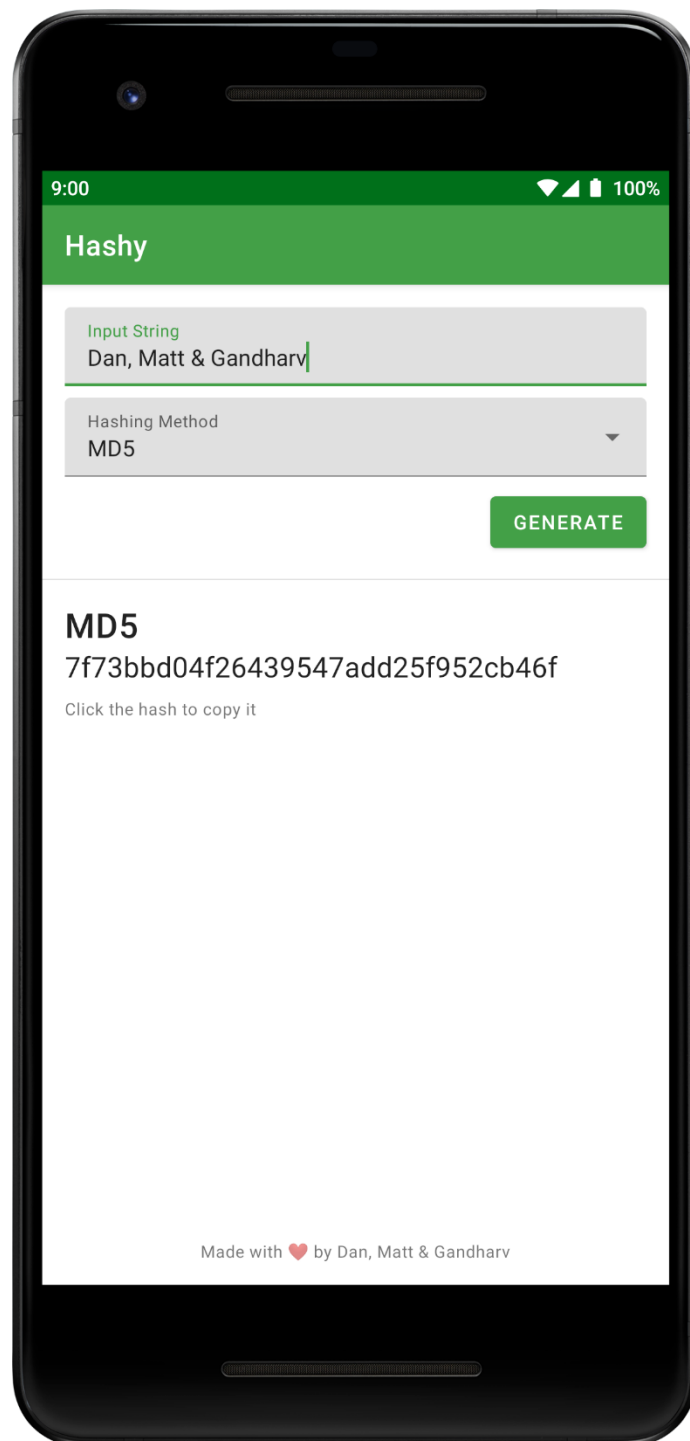
Java: Java is an entire programming language resembling C or C++. It takes a sophisticated programmer to create Java code. And it requires a sophisticated programmer to maintain it. With Java, you can create complete applications. Or you can attach a small group of instructions, a Java "applet" that improves your basic HTML. A Java Applet can also cause text to change color when you roll over it. A game, a calendar, a scrolling text banner can all be created with Java Applets. There are sometimes compatibility problems between Java and various browsers, operating systems or computers, and if not written correctly, it can be slow to load. Java is a powerful programming language with excellent security, but you need to be aware of the tradeoffs.

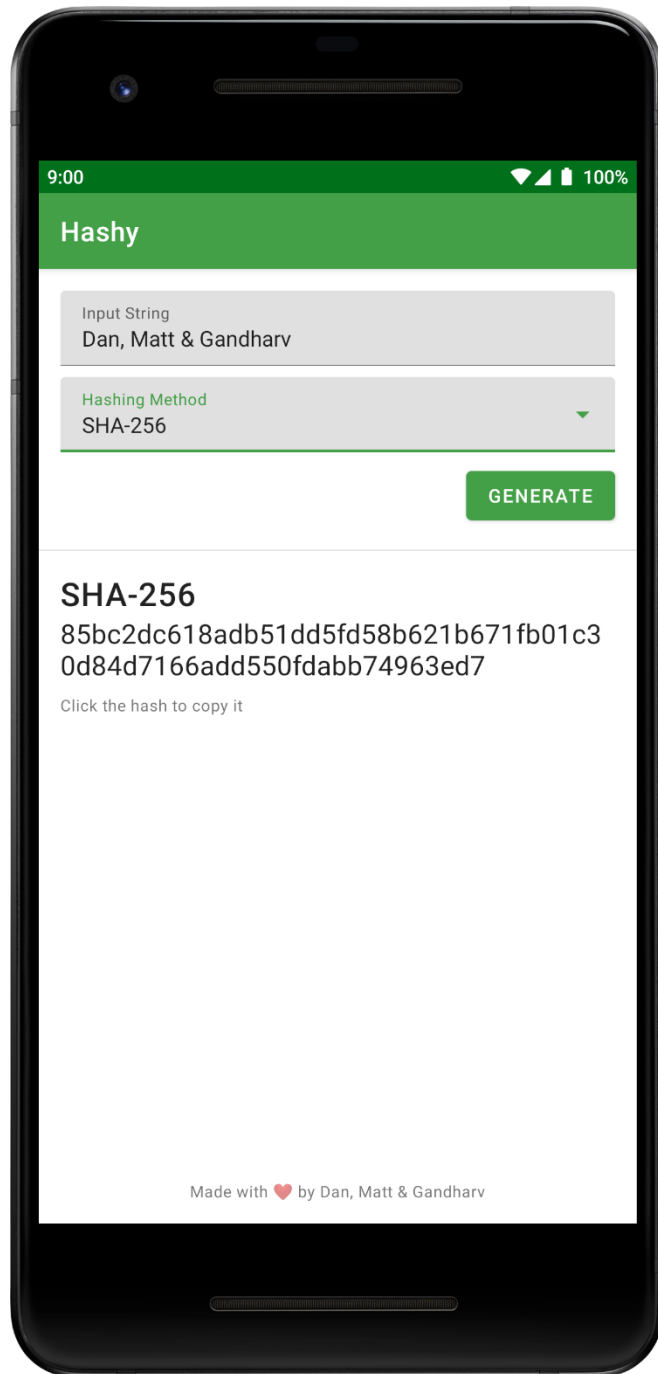
Hardware Requirements

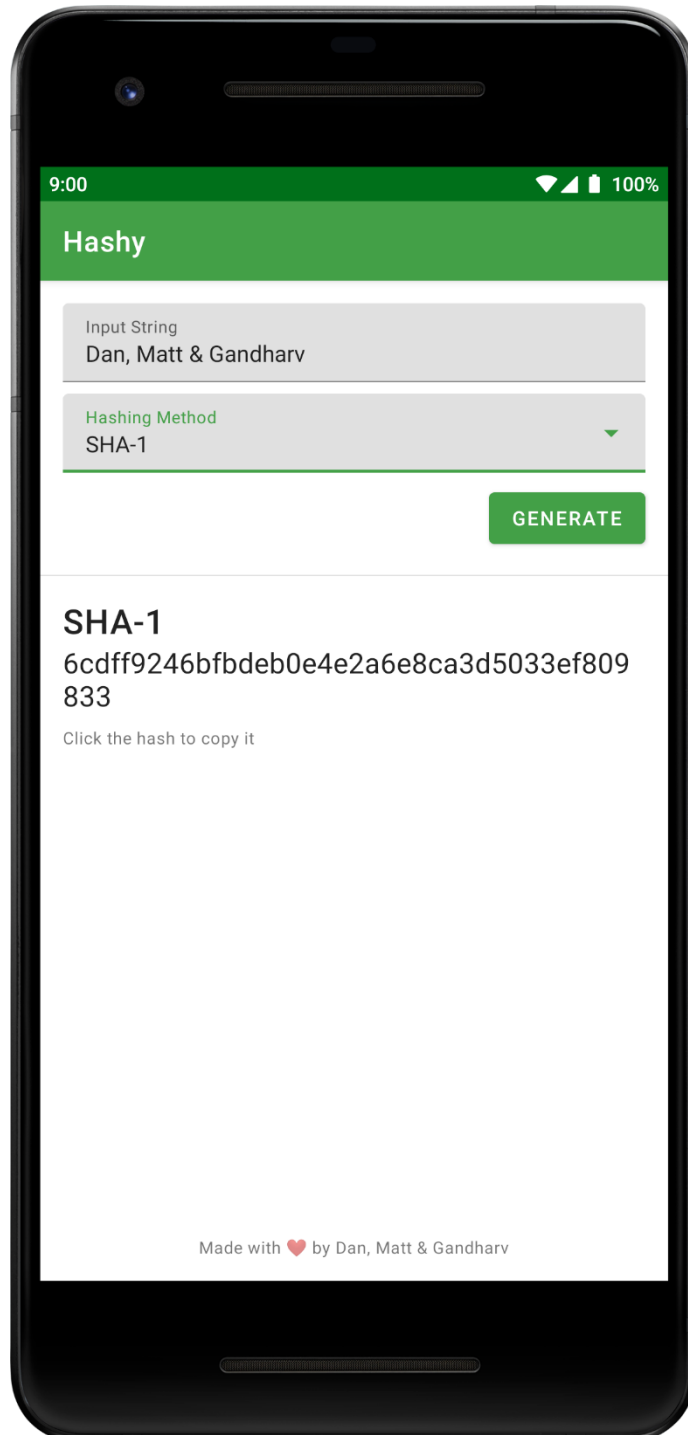
- Microsoft Windows 7/8/10 (32-bit or 64-bit)
- 3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
- 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

3.4 Experiment and Results:

The Application:







3.5 Conclusion and Future scope:

Applications of hashing include: -

Password storage: Hashing protects how passwords are stored and saved. Instead of keeping a password, in the form of a plaintext. It is stored as a hash value or a digest. The hash values are stored in a hash table. An intruder can only see the hash values and cannot log into a system using the hash value.

Password verification: Hashing is used for password verification every time you login into an application, account, or system. A password verifies if you are the actual user of that account. If the password you enter matches the hash value on the server-side, you get authorization.

Checking of data integrity: Hashing checks for data integrity. It gives the user assurance that no data has been modified and the data is correct. It also assures the user that the data is original.

In conclusion, hashing and hash functions are essential tools in computer security. We have learned the objectives of hashing which include data integrity and authentication. We have learned what hashing is, how it works, and went over hash functions in cryptography. We have shown how cryptographic hash functions slowly gained its importance in the field of cryptology. We have made all attempts to give a complete picture of cryptographic hashes, its design techniques and vulnerabilities.

References:

- Eric Conrad, ... Joshua Feldman, in Eleventh Hour CISSP (Second Edition), 2014
- Cryptographic Hash Functions: A Review Rajeev Sobti¹, G.Geetha² ¹School of Computer Science, Lovely Professional University Phagwara, Punjab 144806, India ²School of Computer Applications, Lovely Professional University Phagwara, Punjab 144806, India
- A SECURED CRYPTOGRAPHIC HASHING ALGORITHM Prof. Rakesh Mohanty^{#1}, Niharjyoti Sarangi^{#2}, Sukant Kumar Bishi^{#3} [#]Department of Computer Science and Applications VSSUT, Burla, Orissa, India.