Final Year Project Report

Full Unit – Final Report

TigerCrypt

Inaam kabbara

Shiron newton

Teacher: MAJED JABER

Department Name – Cyber security

University Name - EPITA

May 19, 2023

Table of Contents

Abstract

The Android uses AES encryption, password-encryption application “TigerCrypt” to protect passwords. An intuitive user interface, registration and login forms, encryption and decryption tables, and password management tools are all provided by the app. “TigerCrypt” offers customers a handy way to retrieve and see passwords while securely storing and managing passwords for many domains in an encrypted format.

“TigerCrypt” distinguishes itself from other password encryption application by putting a special emphasis on password management for certain domains and by providing an easy-to-use user interface. The software has the potential to be a useful resource for Android users looking for a safe and simple way to manage their passwords, even though it is not yet accessible on any app stores.

Introduction

A decentralized and secure password management solution was required, which is why the “TigerCrypt” password encryption app was developed. “TigerCrypt” seeks to give consumers a more secure and autonomous solution to handle their passwords in a world where numerous apps are integrated and centralized.

“TigerCrypt” specifically tries to address the issue of passwords being susceptible to theft and hacking. Passwords are protected by AES encryption, which offers a high level of security and guarantees that user data is secure.

A user-friendly interface that enables users to examine both encrypted and decrypted passwords on tables is one of “TigerCrypt”'s primary features. As a result, users can easily manage their passwords and are guaranteed instant access to the data they require.

When compared to other password encryption programs, “TigerCrypt” stands out due to its emphasis on security and decentralization as well as its user-friendly layout. Without the need of centralized servers or laborious encryption procedures, the app offers users a safe and efficient way to save and manage their passwords.

In terms of advantages, “TigerCrypt” gives customers a simple and safe solution to manage their passwords, making sure that their information is kept secure from unauthorized access. Anyone looking for a trustworthy and user-friendly password management solution will find the app to be the perfect option because it is simple to use and has a variety of functions.

“TigerCrypt” is presently in the APK development stage and cannot be downloaded from any app stores. However, it is intended to release the app on the Android store following successful testing and presentation.

Literature Review:

With its decentralized password manager, the “TigerCrypt” project wants to give users greater security and privacy. We will look at the existing research on password encryption methods, decentralized password management, and user-focused attributes and features of password managers in this review of the literature. In conducting this literature evaluation, the following sources were used:

1. JumpCloud Expands Portfolio with Decentralized Password Manager

A decentralized password manager was just introduced to JumpCloud's suite of cloud-based identity and access management tools. Users can store their passwords locally on their device using the decentralized password manager rather than on a centralized server. Users benefit from increased security and privacy thanks to this strategy. The advantages of decentralized password management are discussed in this article, along with the expanding popularity of this strategy.

2. Data Encryption and Data Decryption Using the Advanced Encryption Standard (AES)

Data encryption and decryption commonly employ the Advanced Encryption Standard (AES) algorithm. The key size, number of rounds, and key scheduling are among the specifics of the AES algorithm that are examined in this work. A summary of the AES algorithm's functionality and security is also provided in the publication. Given that the “TigerCrypt” project uses the AES method for password encryption, this information is pertinent to it.

3. Password Manager Usability, Security, and Trust: A Search for User-Centric Features

This study examines the usability, security, and trust aspects of password managers from the user's point of view. The difficulties of establishing password managers that strike a balance between these characteristics are covered in the article, along with the value of user-centered design in producing efficient password managers. The study also offers suggestions for enhancing the password managers' usability, security, and reliability.

4. A Survey on Decentralized Password Management

The advantages of decentralized password management systems over centralized password management systems are discussed in this paper. The paper presents an overview of current decentralized password management systems and analyses the difficulties in creating decentralized password managers, such as key management and synchronization. The “TigerCrypt” project, which is creating a decentralized password manager, can learn several important lessons from this study.

Overall, the study of the literature emphasizes the value of decentralized password management for increased security and privacy as well as the difficulties in creating user-centered password managers that strike a balance between usability, security, and trust. The “TigerCrypt” project has robust security thanks to the use of the AES algorithm for password encryption, however key management and synchronization remain challenging issues. The literature analysis stresses the necessity for user-centered design in producing efficient password managers and offers insightful information for the development of the “TigerCrypt” project.

Methodology

Explain the methods, techniques, or approaches you employed to conduct your project. Describe the data collection process, any experiments performed, software or tools used, and any relevant assumptions or limitations.

Project goals and objectives:

we will analyze the project requirements and determine the features necessary for developing a decentralized password manager app with an AES encryption algorithm. We will also consider the importance of providing a user-friendly interface for the app and make sure that the design and development process aligns with the project objectives.

Development process:

will outline the various stages involved in the project and describe the tools and techniques used for each stage. This may include

planning- focus on current trends based on the length and strength and weakness of passwords

language selection – using java and java based libraries

platform design- android studio include with database helper for database store

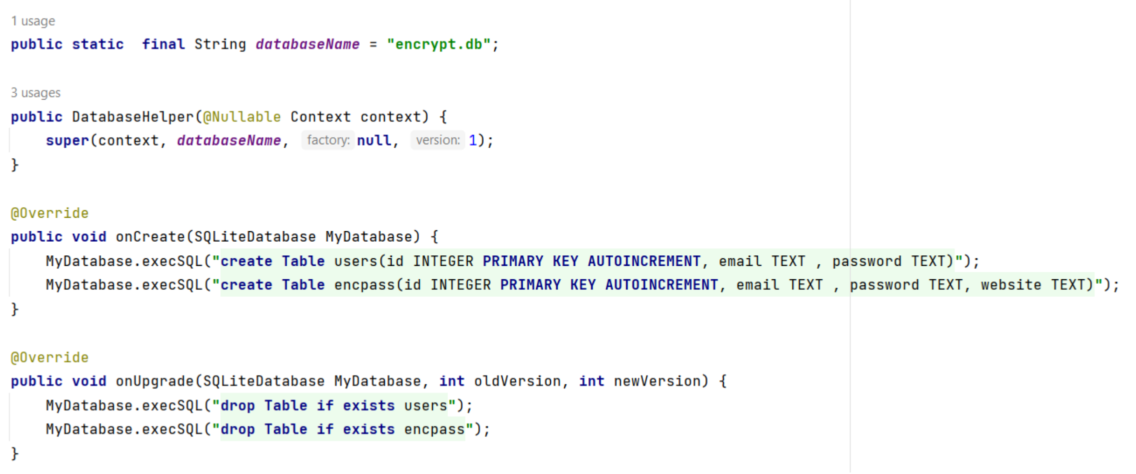
implementation using Android Studio, testing in the form of apk testing, and releasing the app in the app store. We will ensure that the development process is efficient, effective, and adheres to best practices.

Ethical considerations:

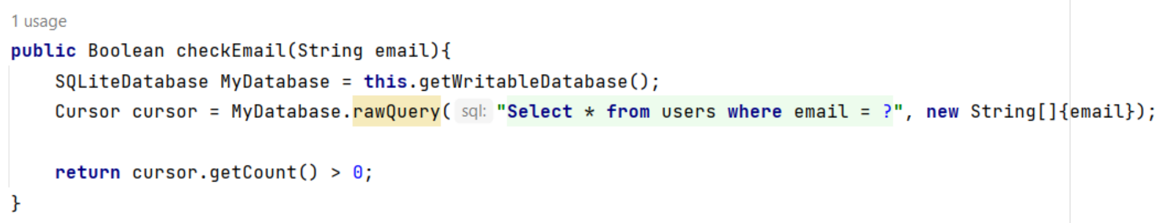
we will ensure that user privacy and data security are protected at all times. We will obtain user consent for data collection and usage and make sure that the decentralized nature of the app ensures that user data stays within their device.

Project timeline:

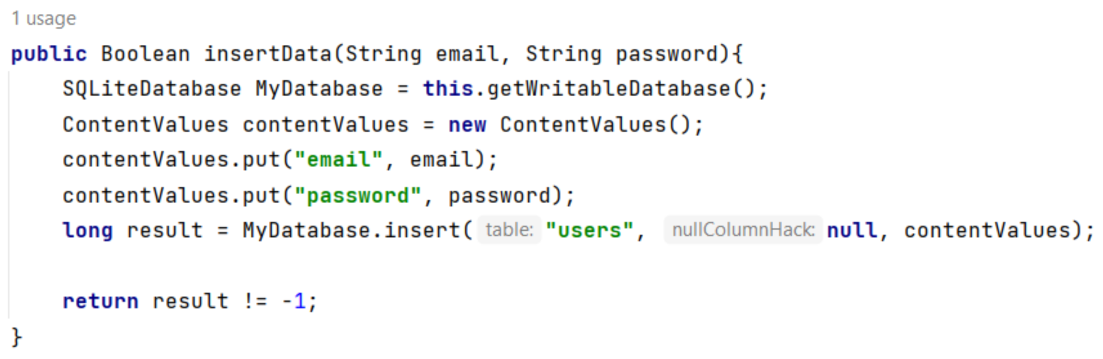
To complete the “TigerCrypt” project within the desired timeline, we will establish milestones and deadlines for each stage of the development process. Within 1 month half to implement the tool



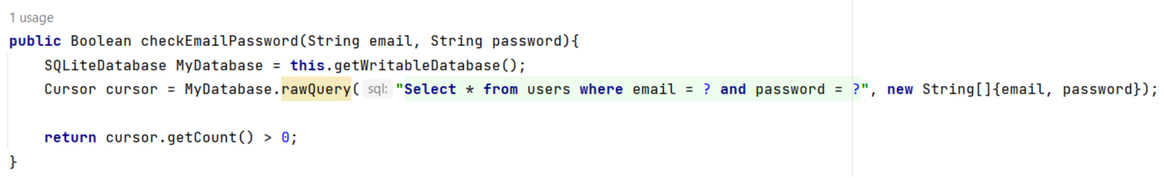
```
1 usage
public static  final String databaseName = "encrypt.db";

3 usages
public DatabaseHelper(@Nullable Context context) {
    super(context, databaseName,  factory: null,  version: 1);
}

@Override
public void onCreate(SQLiteDatabase MyDatabase) {
    MyDatabase.execSQL("create Table users(id INTEGER PRIMARY KEY AUTOINCREMENT, email TEXT , password TEXT)");
    MyDatabase.execSQL("create Table encpass(id INTEGER PRIMARY KEY AUTOINCREMENT, email TEXT , password TEXT, website TEXT)");
}

@Override
public void onUpgrade(SQLiteDatabase MyDatabase, int oldVersion, int newVersion) {
    MyDatabase.execSQL("drop Table if exists users");
    MyDatabase.execSQL("drop Table if exists encpass");
}
```



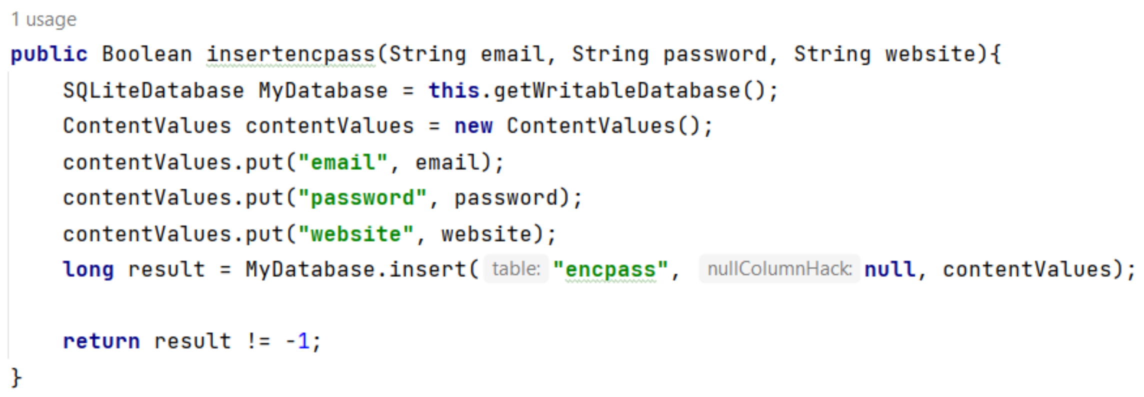
```
1 usage
public Boolean checkEmail(String email){
    SQLiteDatabase MyDatabase = this.getWritableDatabase();
    Cursor cursor = MyDatabase.rawQuery( sql: "Select * from users where email = ?", new String[]{email});

    return cursor.getCount() > 0;
}
```



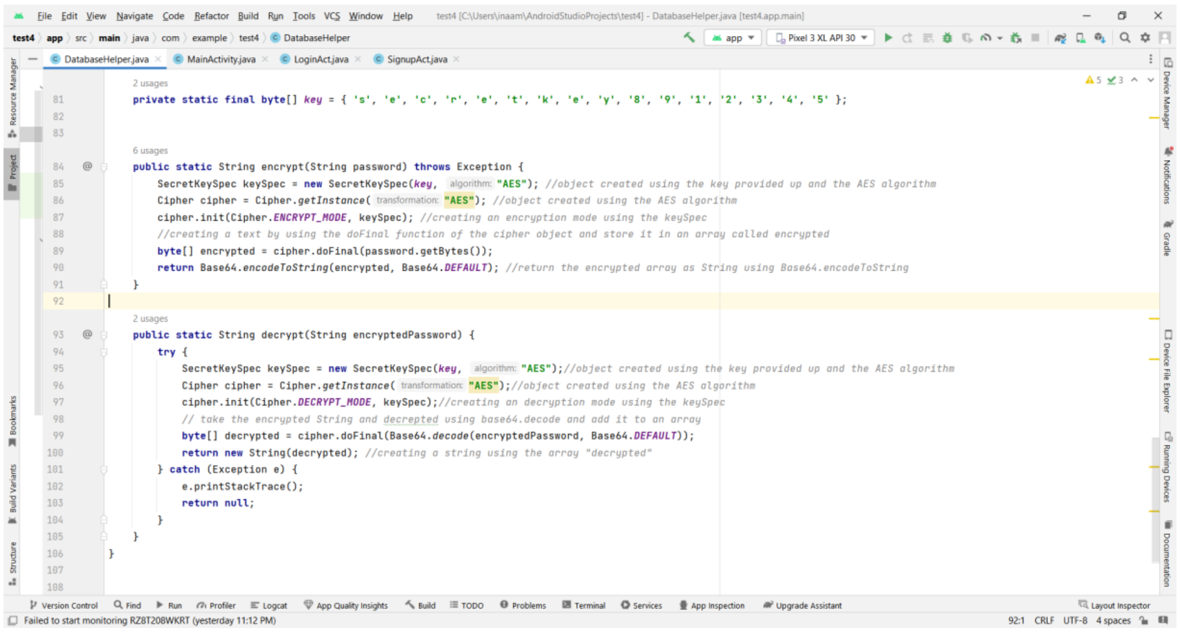
```
1 usage
public Boolean insertData(String email, String password){
    SQLiteDatabase MyDatabase = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("email", email);
    contentValues.put("password", password);
    long result = MyDatabase.insert( table: "users",  nullColumnHack: null, contentValues);

    return result != -1;
}
```



```
1 usage
public Boolean checkEmailPassword(String email, String password){
    SQLiteDatabase MyDatabase = this.getWritableDatabase();
    Cursor cursor = MyDatabase.rawQuery( sql: "Select * from users where email = ? and password = ?", new String[]{email, password});

    return cursor.getCount() > 0;
}
```

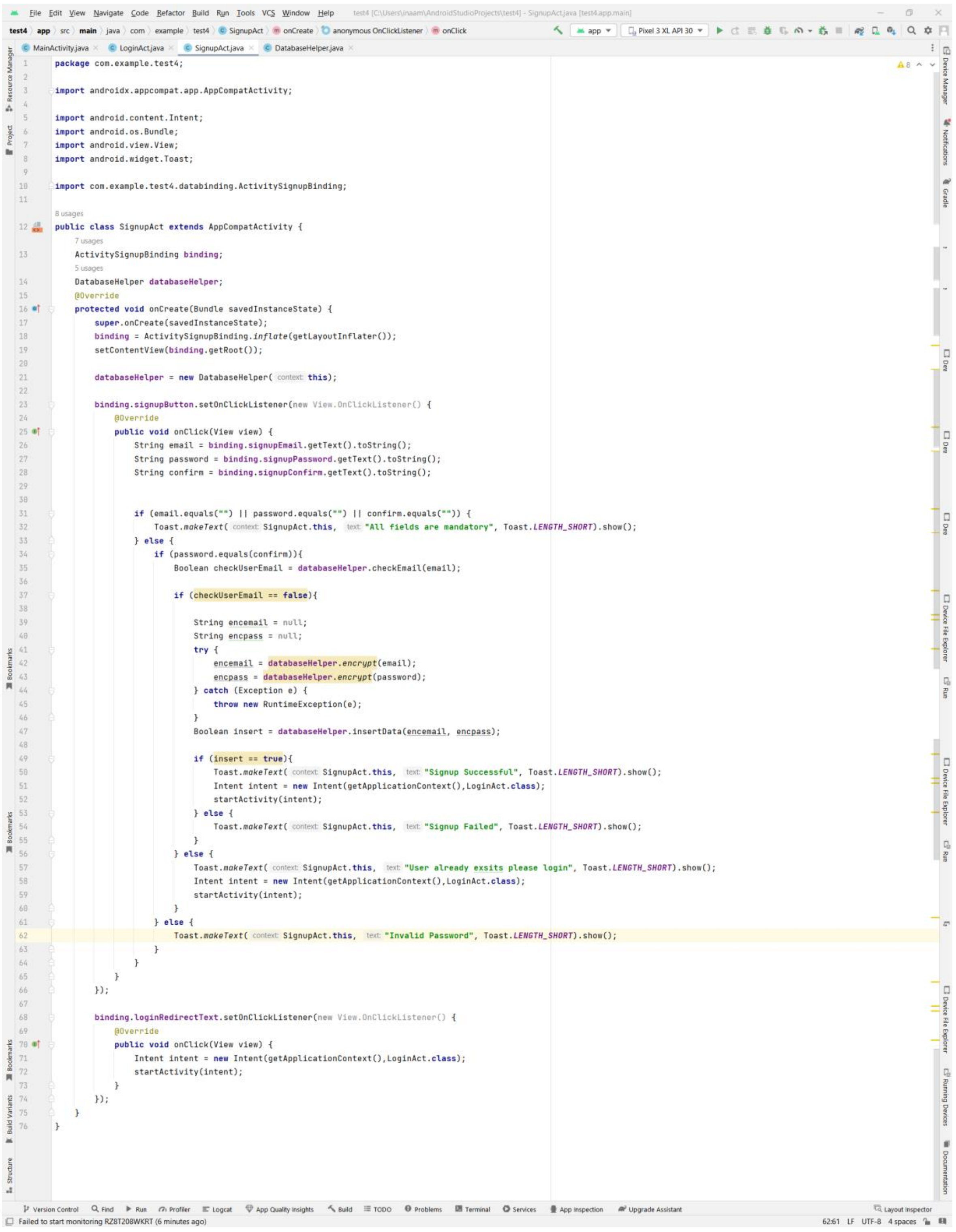


```
1 usage
public Boolean insertencpass(String email, String password, String website){
    SQLiteDatabase MyDatabase = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("email", email);
    contentValues.put("password", password);
    contentValues.put("website", website);
    long result = MyDatabase.insert( table: "encpass",  nullColumnHack: null, contentValues);

    return result != -1;
}
```



```
    2 usages
    private static final byte[] key = { 's', 'e', 'c', 'r', 'e', 't', 'k', 'e', 'y', '8', '9', '1', '2', '3', '4', '5' };


    6 usages
    public static String encrypt(String password) throws Exception {
        SecretKeySpec keySpec = new SecretKeySpec(key,  algorithm: "AES"); //object created using the key provided up and the AES algorithm
        Cipher cipher = Cipher.getInstance( transformation: "AES"); //object created using the AES algorithm
        cipher.init(Cipher.ENCRYPT_MODE, keySpec); //creating an encryption mode using the keySpec
        //creating a text by using the doFinal function of the cipher object and store it in an array called encrypted
        byte[] encrypted = cipher.doFinal(password.getBytes());
        return Base64.encodeToString(encrypted, Base64.DEFAULT); //return the encrypted array as String using Base64.encodeToString
    }

    2 usages
    public static String decrypt(String encryptedPassword) {
        try {
            SecretKeySpec keySpec = new SecretKeySpec(key,  algorithm: "AES");//object created using the key provided up and the AES algorithm
            Cipher cipher = Cipher.getInstance( transformation: "AES");//object created using the AES algorithm
            cipher.init(Cipher.DECRYPT_MODE, keySpec);//creating an decryption mode using the keySpec
            // take the encrypted String and decrepted using base64.decode and add it to an array
            byte[] decrypted = cipher.doFinal(Base64.decode(encryptedPassword, Base64.DEFAULT));
            return new String(decrypted); //creating a string using the array "decrypted"
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

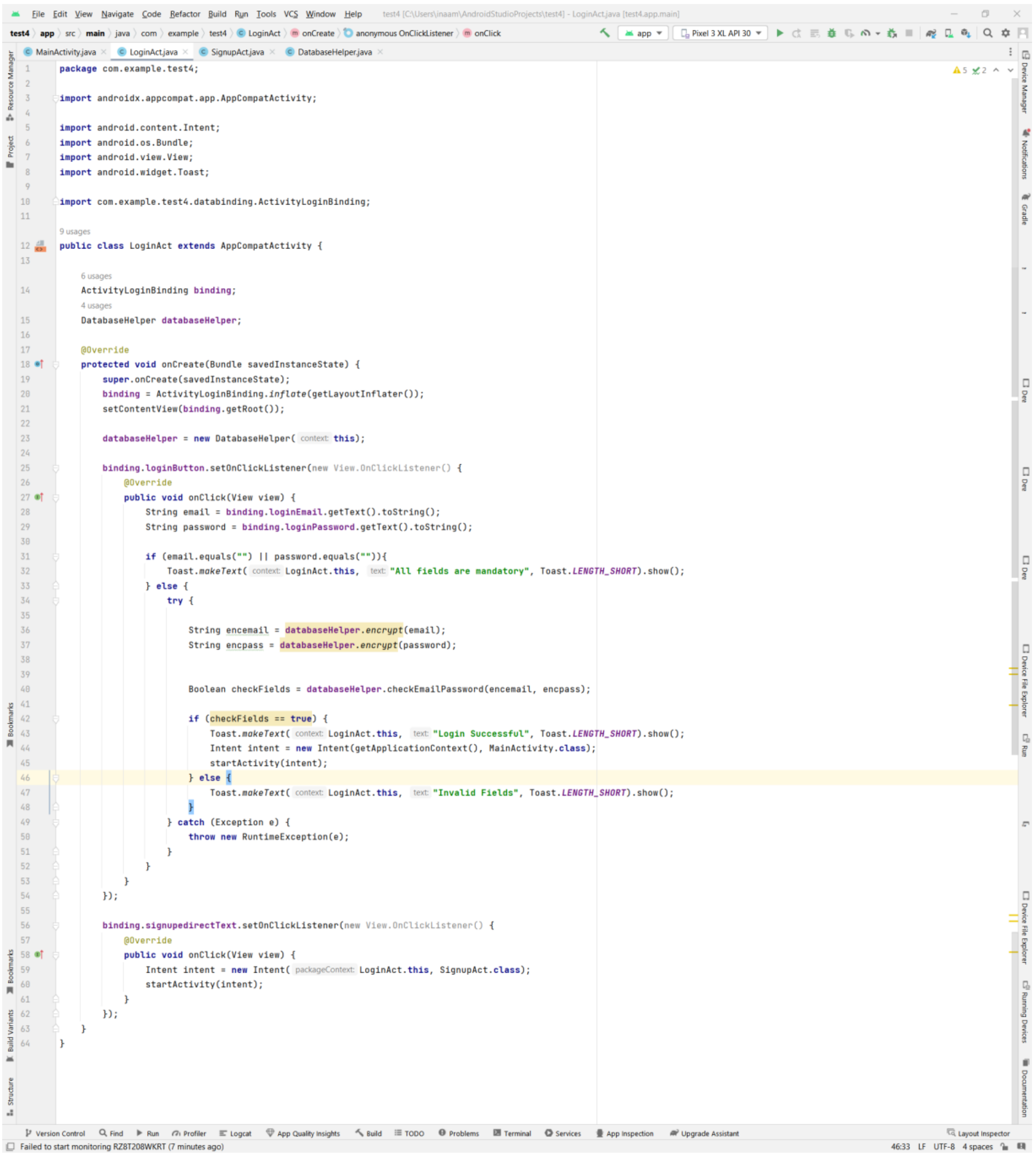
Failed to start monitoring RZ8T208WKRT (yesterday 11:12 PM)


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
test4 [C:\Users\inaam\AndroidStudioProjects\test4] - SignupAct.java [test4.app.main]
test4 › app › src › main › java › com › example › test4 › SignupAct › onCreate › anonymous OnClickListener › onClick
app
Pixel 3 XL API 30
MainActivity.java
LoginAct.java
SignupAct.java
DatabaseHelper.java

```java
package com.example.test4;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import com.example.test4.databinding.ActivitySignupBinding;

8 usages
public class SignupAct extends AppCompatActivity {
    7 usages
    ActivitySignupBinding binding;
    5 usages
    DatabaseHelper databaseHelper;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivitySignupBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        databaseHelper = new DatabaseHelper( context: this);

        binding.signupButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email = binding.signupEmail.getText().toString();
                String password = binding.signupPassword.getText().toString();
                String confirm = binding.signupConfirm.getText().toString();


                if (email.equals("") || password.equals("") || confirm.equals("")) {
                    Toast.makeText( context: SignupAct.this,  text: "All fields are mandatory", Toast.LENGTH_SHORT).show();
                } else {
                    if (password.equals(confirm)){
                        Boolean checkUserEmail = databaseHelper.checkEmail(email);

                        if (checkUserEmail == false){

                            String encemail = null;
                            String encpass = null;
                            try {
                                encemail = databaseHelper.encrypt(email);
                                encpass = databaseHelper.encrypt(password);
                            } catch (Exception e) {
                                throw new RuntimeException(e);
                            }
                            Boolean insert = databaseHelper.insertData(encemail, encpass);

                            if (insert == true){
                                Toast.makeText( context: SignupAct.this,  text: "Signup Successful", Toast.LENGTH_SHORT).show();
                                Intent intent = new Intent(getApplicationContext(),LoginAct.class);
                                startActivity(intent);
                            } else {
                                Toast.makeText( context: SignupAct.this,  text: "Signup Failed", Toast.LENGTH_SHORT).show();
                            }
                        } else {
                            Toast.makeText( context: SignupAct.this,  text: "User already exsits please login", Toast.LENGTH_SHORT).show();
                            Intent intent = new Intent(getApplicationContext(),LoginAct.class);
                            startActivity(intent);
                        }
                    } else {
                        Toast.makeText( context: SignupAct.this,  text: "Invalid Password", Toast.LENGTH_SHORT).show();
                    }
                }
            }
        });

        binding.loginRedirectText.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplicationContext(),LoginAct.class);
                startActivity(intent);
            }
        });
    }
}
```

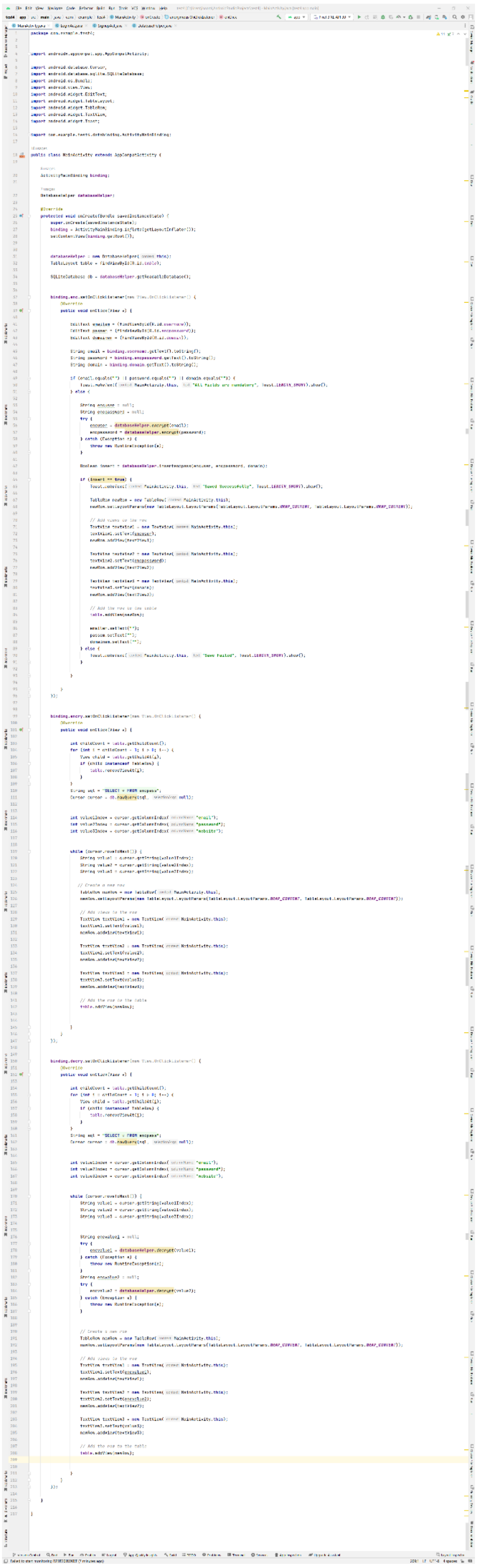
Failed to start monitoring RZ8T208WKRT (6 minutes ago)

62:61 LF UTF-8 4 spaces


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
test4 [C:\Users\inaam\AndroidStudioProjects\test4] - LoginAct.java [test4.app.main]
test4 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 test4 〉 LoginAct 〉 onCreate 〉 anonymous OnClickListener 〉 onClick
app
Pixel 3 XL API 30
MainActivity.java
LoginAct.java
SignupAct.java
DatabaseHelper.java
Version Control
Find
Run
Profiler
Logcat
App Quality Insights
Build
TODO
Problems
Terminal
Services
App Inspection
Upgrade Assistant
Layout Inspector

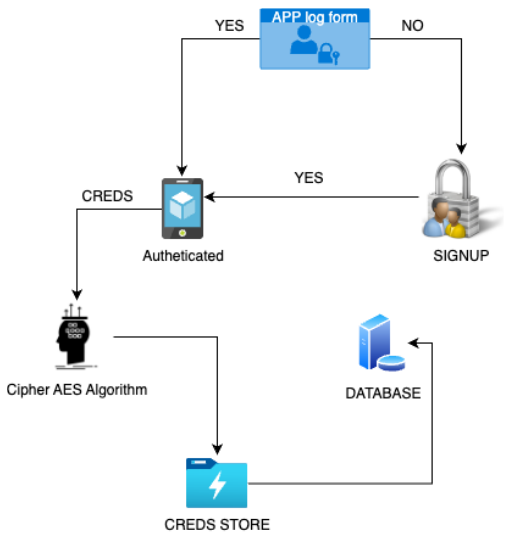
```java
package com.example.test4;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import com.example.test4.databinding.ActivityLoginBinding;

9 usages
public class LoginAct extends AppCompatActivity {

    6 usages
    ActivityLoginBinding binding;
    4 usages
    DatabaseHelper databaseHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityLoginBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        databaseHelper = new DatabaseHelper( context: this);

        binding.loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email = binding.loginEmail.getText().toString();
                String password = binding.loginPassword.getText().toString();

                if (email.equals("") || password.equals("")){
                    Toast.makeText( context: LoginAct.this,  text: "All fields are mandatory", Toast.LENGTH_SHORT).show();
                } else {
                    try {

                        String encemail = databaseHelper.encrypt(email);
                        String encpass = databaseHelper.encrypt(password);


                        Boolean checkFields = databaseHelper.checkEmailPassword(encemail, encpass);

                        if (checkFields == true) {
                            Toast.makeText( context: LoginAct.this,  text: "Login Successful", Toast.LENGTH_SHORT).show();
                            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                            startActivity(intent);
                        } else {
                            Toast.makeText( context: LoginAct.this,  text: "Invalid Fields", Toast.LENGTH_SHORT).show();
                        }
                    } catch (Exception e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        });

        binding.signupedirectText.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent( packageContext: LoginAct.this, SignupAct.class);
                startActivity(intent);
            }
        });
    }
}
```

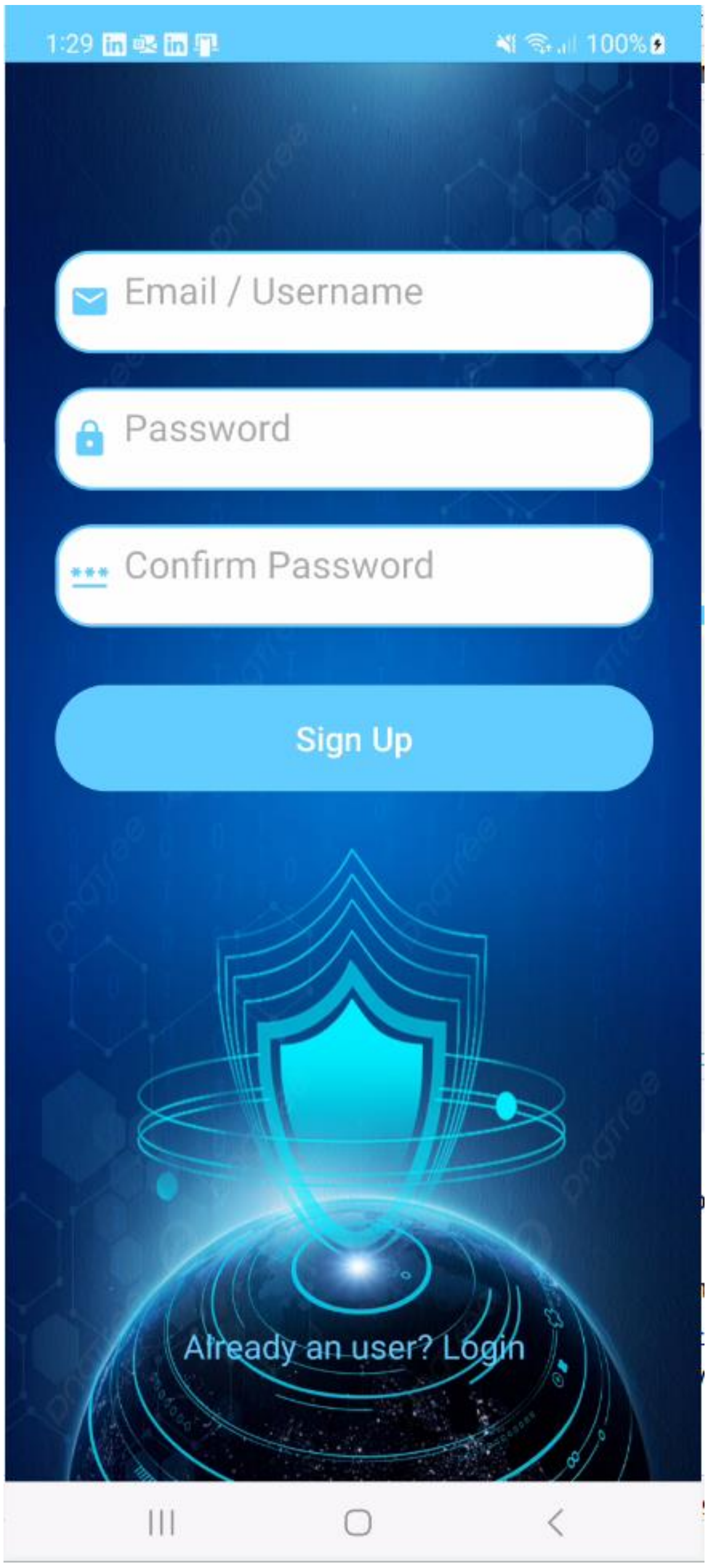
```
Failed to start monitoring RZ8T208WKRT (7 minutes ago)                46:33  LF  UTF-8  4 spaces
```

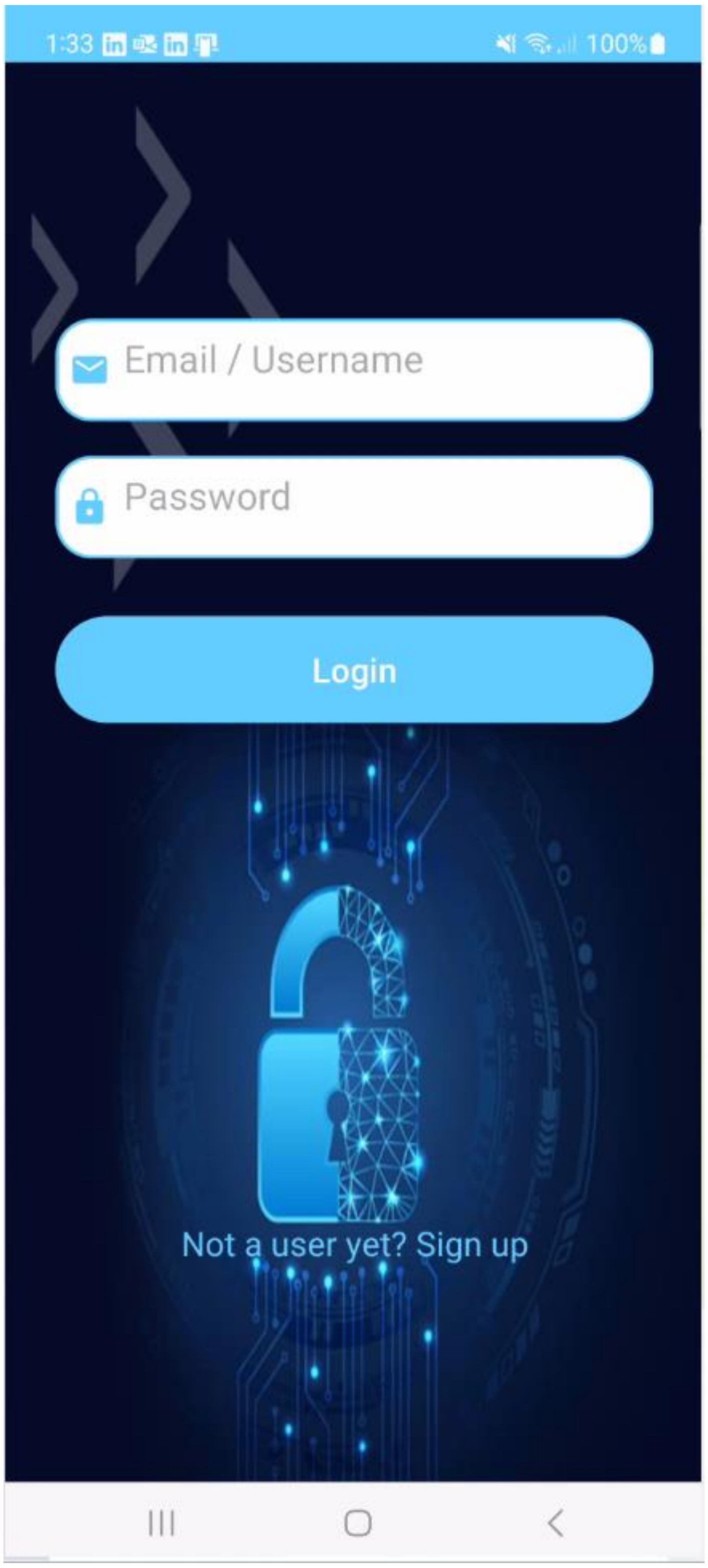


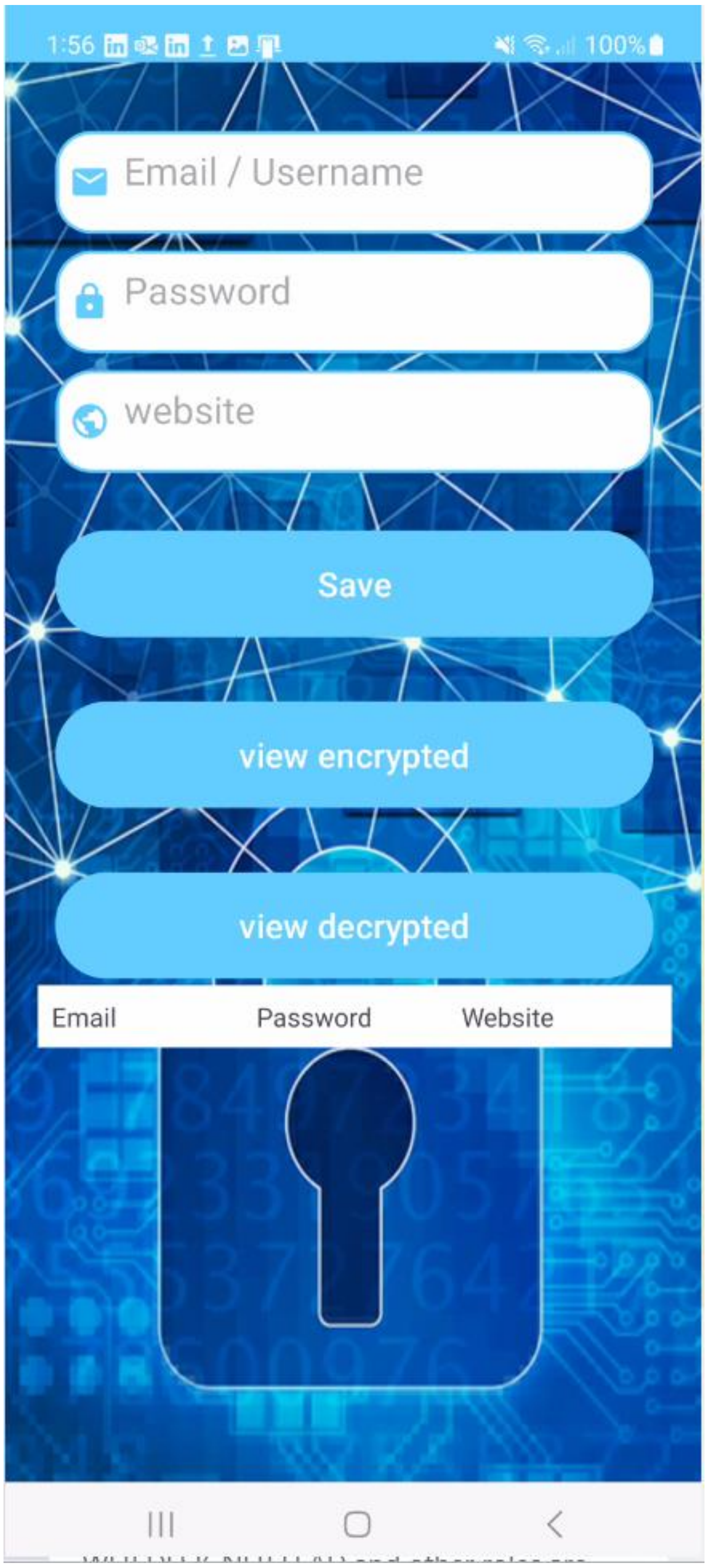
```java
package com.example.test4;


import androidx.appcompat.app.AppCompatActivity;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import android.widget.Toast;

import com.example.test4.databinding.ActivityMainBinding;

public class MainActivity extends AppCompatActivity {

    ActivityMainBinding binding;

    DatabaseHelper databaseHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());


        databaseHelper = new DatabaseHelper(this);
        TableLayout table = findViewById(R.id.table);

        SQLiteDatabase db = databaseHelper.getReadableDatabase();


        binding.enc.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                EditText emailen = (findViewById(R.id.username));
                EditText passen = (findViewById(R.id.encpassword));
                EditText domainen = (findViewById(R.id.domain));

                String email = binding.username.getText().toString();
                String password = binding.encpassword.getText().toString();
                String domain = binding.domain.getText().toString();

                if (email.equals("") || password.equals("") || domain.equals("")) {
                    Toast.makeText(MainActivity.this, "All fields are mandatory", Toast.LENGTH_SHORT).show();
                } else {

                    String encuser = null;
                    String encpassword = null;
                    try {
                        encuser = databaseHelper.encrypt(email);
                        encpassword = databaseHelper.encrypt(password);
                    } catch (Exception e) {
                        throw new RuntimeException(e);
                    }

                    Boolean insert = databaseHelper.insertencpass(encuser, encpassword, domain);

                    if (insert == true) {
                        Toast.makeText(MainActivity.this, "Saved Successfully", Toast.LENGTH_SHORT).show();

                        TableRow newRow = new TableRow(MainActivity.this);
                        newRow.setLayoutParams(new TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT, TableLayout.LayoutParams.WRAP_CONTENT));

                        // Add views to the row
                        TextView textView1 = new TextView(MainActivity.this);
                        textView1.setText(encuser);
                        newRow.addView(textView1);

                        TextView textView2 = new TextView(MainActivity.this);
                        textView2.setText(encpassword);
                        newRow.addView(textView2);

                        TextView textView3 = new TextView(MainActivity.this);
                        textView3.setText(domain);
                        newRow.addView(textView3);

                        // Add the row to the table
                        table.addView(newRow);

                        emailen.setText("");
                        passen.setText("");
                        domainen.setText("");
                    } else {
                        Toast.makeText(MainActivity.this, "Save Failed", Toast.LENGTH_SHORT).show();
                    }

                }

            }
        });


        binding.encry.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                int childCount = table.getChildCount();
                for (int i = childCount - 1; i > 0; i--) {
                    View child = table.getChildAt(i);
                    if (child instanceof TableRow) {
                        table.removeViewAt(i);
                    }
                }
                String sql = "SELECT * FROM encpass";
                Cursor cursor = db.rawQuery(sql, null);


                int value1Index = cursor.getColumnIndex("email");
                int value2Index = cursor.getColumnIndex("password");
                int value3Index = cursor.getColumnIndex("website");


                while (cursor.moveToNext()) {
                    String value1 = cursor.getString(value1Index);
                    String value2 = cursor.getString(value2Index);
                    String value3 = cursor.getString(value3Index);

                  // Create a new row
                    TableRow newRow = new TableRow(MainActivity.this);
                    newRow.setLayoutParams(new TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT, TableLayout.LayoutParams.WRAP_CONTENT));

                    // Add views to the row
                    TextView textView1 = new TextView(MainActivity.this);
                    textView1.setText(value1);
                    newRow.addView(textView1);

                    TextView textView2 = new TextView(MainActivity.this);
                    textView2.setText(value2);
                    newRow.addView(textView2);

                    TextView textView3 = new TextView(MainActivity.this);
                    textView3.setText(value3);
                    newRow.addView(textView3);

                    // Add the row to the table
                    table.addView(newRow);


                }
            }
        });


        binding.decry.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                int childCount = table.getChildCount();
                for (int i = childCount - 1; i > 0; i--) {
                    View child = table.getChildAt(i);
                    if (child instanceof TableRow) {
                        table.removeViewAt(i);
                    }
                }
                String sql = "SELECT * FROM encpass";
                Cursor cursor = db.rawQuery(sql, null);


                int value1Index = cursor.getColumnIndex("email");
                int value2Index = cursor.getColumnIndex("password");
                int value3Index = cursor.getColumnIndex("website");


                while (cursor.moveToNext()) {
                    String value1 = cursor.getString(value1Index);
                    String value2 = cursor.getString(value2Index);
                    String value3 = cursor.getString(value3Index);


                    String encvalue1 = null;
                    try {
                        encvalue1 = databaseHelper.decrypt(value1);
                    } catch (Exception e) {
                        throw new RuntimeException(e);
                    }
                    String encvalue2 = null;
                    try {
                        encvalue2 = databaseHelper.decrypt(value2);
                    } catch (Exception e) {
                        throw new RuntimeException(e);
                    }


                    // Create a new row
                    TableRow newRow = new TableRow(MainActivity.this);
                    newRow.setLayoutParams(new TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT, TableLayout.LayoutParams.WRAP_CONTENT));

                    // Add views to the row
                    TextView textView1 = new TextView(MainActivity.this);
                    textView1.setText(encvalue1);
                    newRow.addView(textView1);

                    TextView textView2 = new TextView(MainActivity.this);
                    textView2.setText(encvalue2);
                    newRow.addView(textView2);

                    TextView textView3 = new TextView(MainActivity.this);
                    textView3.setText(value3);
                    newRow.addView(textView3);

                    // Add the row to the table
                    table.addView(newRow);


                }
            }
        });

    }

}
```

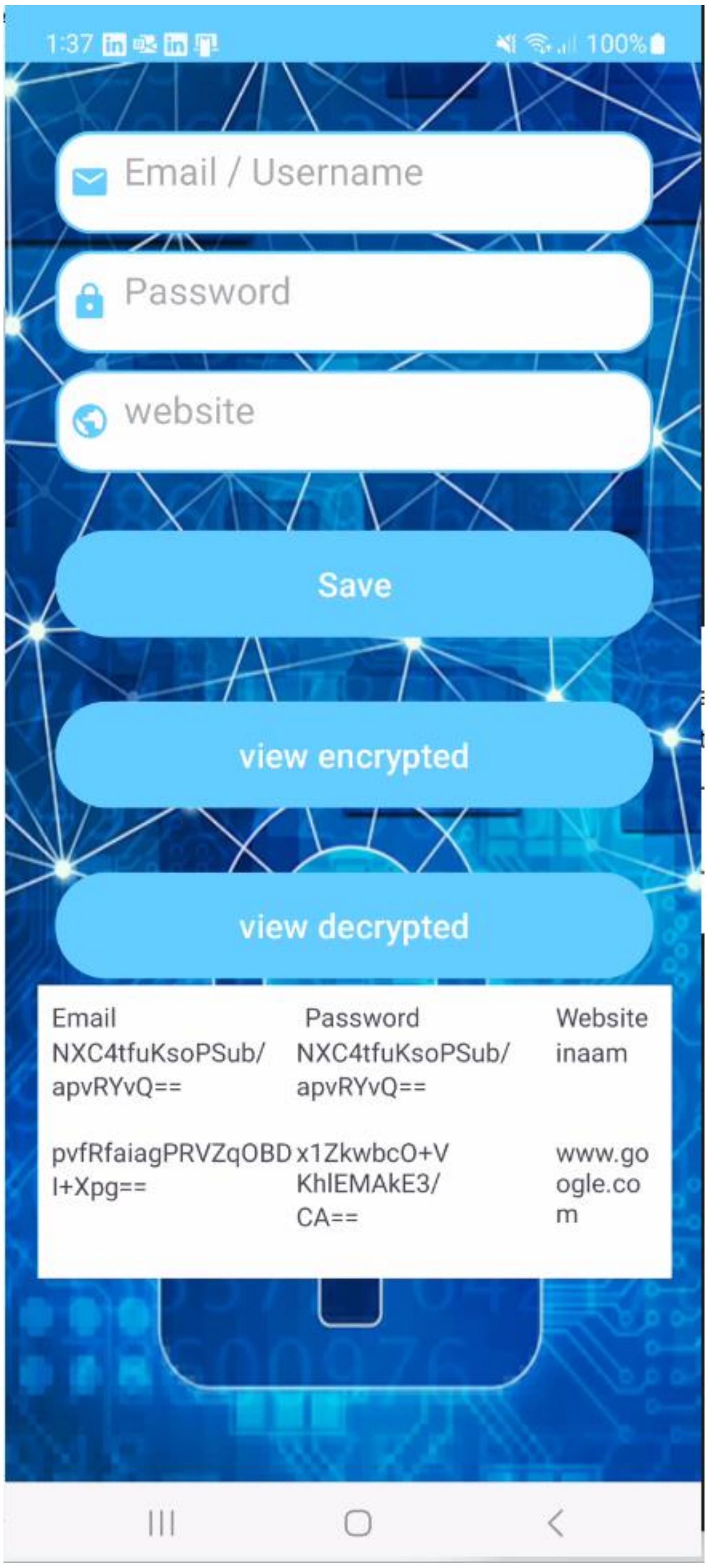
Results


APP log form
YES
NO
CREDS
YES
Autheticated
SIGNUP
Cipher AES Algorithm
DATABASE
CREDS STORE

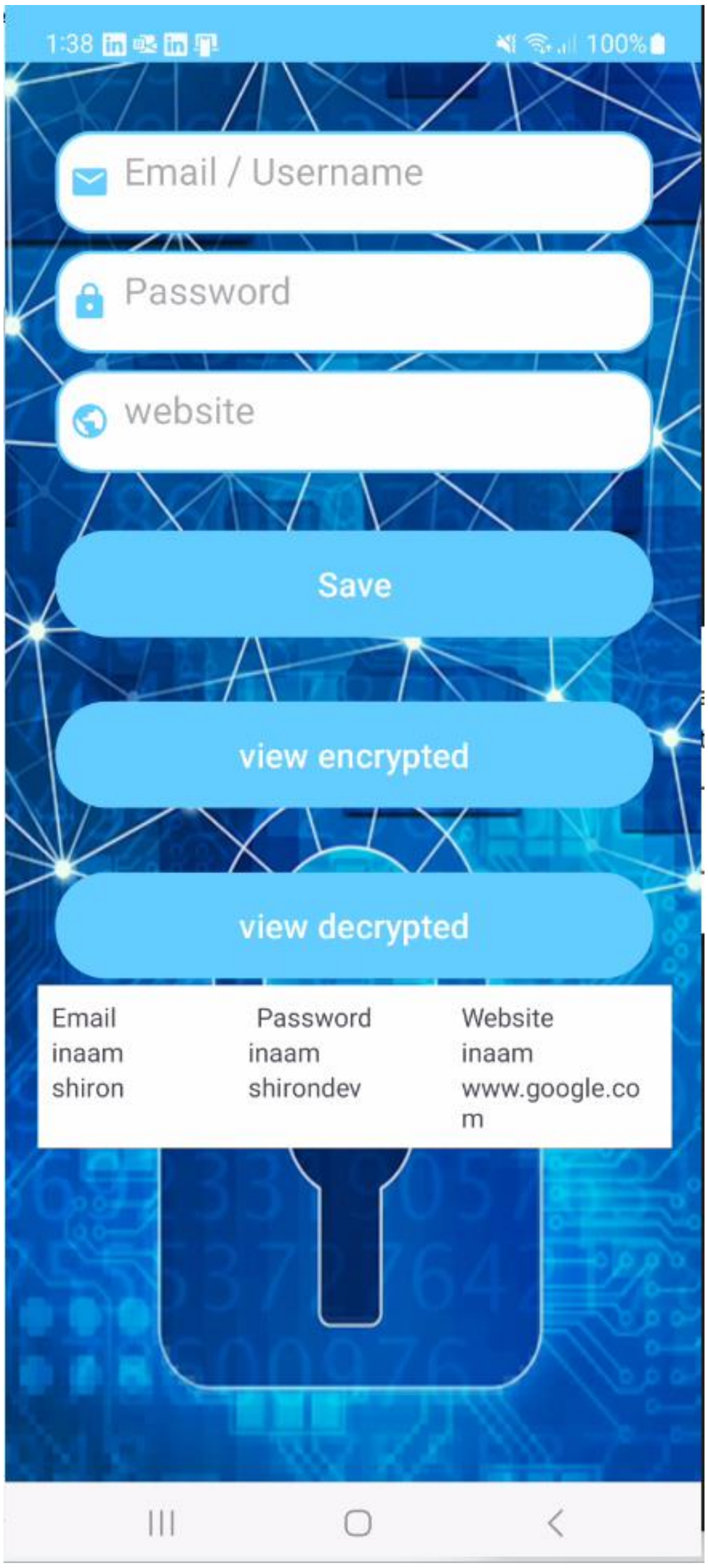

1:29
100%
Email / Username
Password
Confirm Password
Sign Up
Already an user? Login


1:33
100%
Email / Username
Password
Login
Not a user yet? Sign up

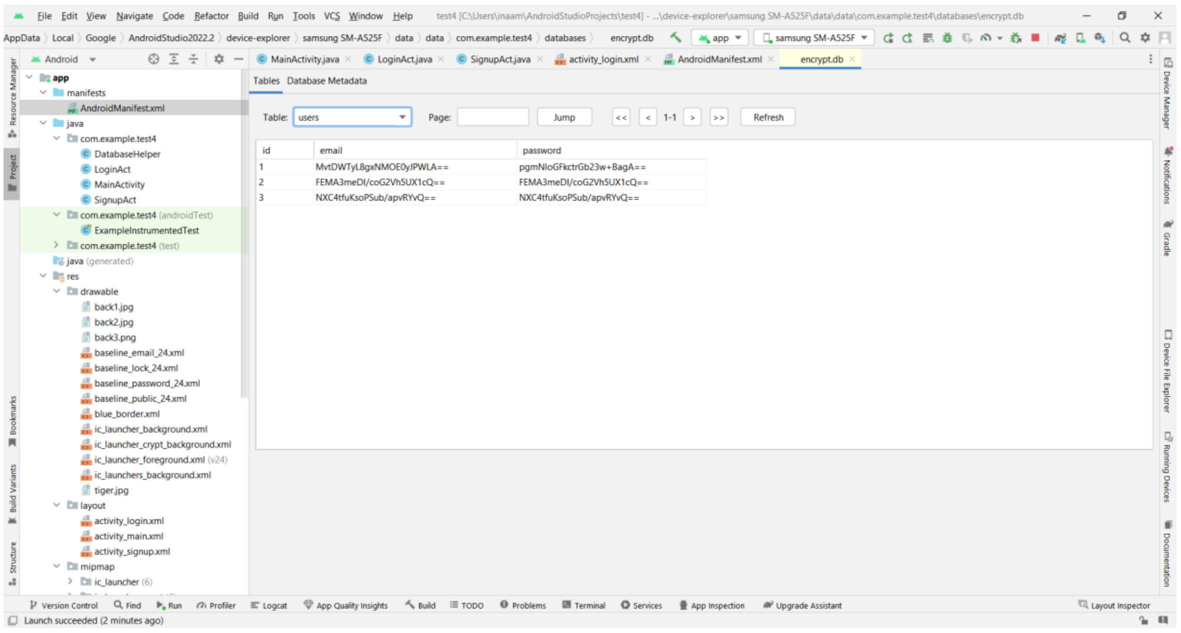

1:56
100%
Email / Username
Password
website
Save
view encrypted
view decrypted
Email
Password
Website



Email	Password	Website
NXC4tfuKsoPSub/apvRYvQ==	NXC4tfuKsoPSub/apvRYvQ==	inaam
pvfRfaiagPRVZqOBDI+Xpg==	x1ZkwbcO+VKhIEMAkE3/CA==	www.google.com



Email	Password	Website
inaam	inaam	inaam
shiron	shirondev	www.google.co m

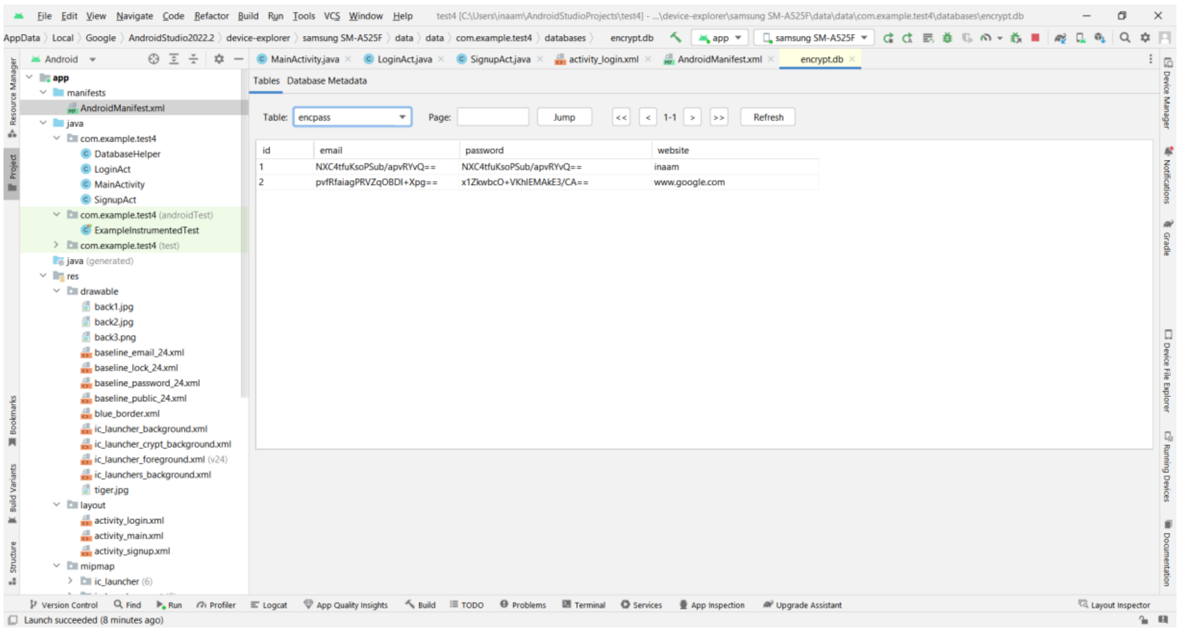

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
AppData 〉 Local 〉 Google 〉 AndroidStudio2022.2 〉 device-explorer 〉 samsung SM-A525F 〉 data 〉 data 〉 com.example.test4 〉 databases 〉 encrypt.db
MainActivity.java
LoginAct.java
SignupAct.java
activity_login.xml
AndroidManifest.xml
encrypt.db
Tables
Database Metadata
Page:
Jump
1-1
Refresh

test4 [C:\Users\inaam\AndroidStudioProjects\test4] - ...\device-explorer\samsung SM-A525F\data\data\com.example.test4\databases\encrypt.db

Table: users

id	email	password
1	MvtDWTyL8gxNMOE0yJPWLA==	pgmNloGFkctrGb23w+BagA==
2	FEMA3meDI/coG2Vh5UX1cQ==	FEMA3meDI/coG2Vh5UX1cQ==
3	NXC4tfuKsoPSub/apvRYvQ==	NXC4tfuKsoPSub/apvRYvQ==

Launch succeeded (2 minutes ago)


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
AppData 〉 Local 〉 Google 〉 AndroidStudio2022.2 〉 device-explorer 〉 samsung SM-A525F 〉 data 〉 data 〉 com.example.test4 〉 databases 〉 encrypt.db
MainActivity.java
LoginAct.java
SignupAct.java
activity_login.xml
AndroidManifest.xml
encrypt.db
Tables
Database Metadata
Page:
Jump
1-1
Refresh

test4 [C:\Users\inaam\AndroidStudioProjects\test4] - ...\device-explorer\samsung SM-A525F\data\data\com.example.test4\databases\encrypt.db

Table: encpass

id	email	password	website
1	NXC4tfuKsoPSub/apvRYvQ==	NXC4tfuKsoPSub/apvRYvQ==	inaam
2	pvfRfaiagPRVZqOBDI+Xpg==	x1ZkwbcO+VKhIEMAkE3/CA==	www.google.com

Launch succeeded (8 minutes ago)

For the report to be feasible and clear I made the diagram of representation the development of the app but for now within the ability to store data yet testing purpose still the credentials are not network connected as a centralized server

Conclusion

A decentralized password management tool called “TigerCrypt” places a strong emphasis on security, privacy, and usability. The software uses AES encryption to make sure that passwords are secure and secured. “TigerCrypt” also offers a user-friendly interface that enables users to manage their passwords easily and effectively. “TigerCrypt”'s decentralized design assures that customers' data is secure and lessens the likelihood of theft and hacking when compared to alternative password managers.

The literature study shows that while key management and synchronization can be difficult, decentralized password management has advantages over centralized systems in terms of security and privacy. User-centered design is also crucial for creating effective password managers that strike a balance between usability, security, and trust.

With an emphasis on current trends in password strength and length, the project's methodology makes use of Java-based libraries and Android Studio for platform design and development. Additionally, the project follows industry standards and makes sure that moral factors are considered.

In conclusion, “TigerCrypt” has the potential to be a helpful tool for Android users looking for a secure and straightforward way to manage their passwords. Once it is made available on the Android store, “TigerCrypt” is ready to provide customers a useful service thanks to its focus on security, decentralization, and user-friendliness.

Future Work

Implementing a distributed database system could entail investigating tools like IPFS make user data storage genuinely decentralized.

Implementing this function will create peer to peer data storage within each separated hash functions for all the smaller divided chunks of data within the tool.

Enhancing security features: Since multi-factor authentication, encryption, and integration with reputable identity suppliers are all essential components of any product that handles sensitive data, you may want to integrate additional security measures.

Enhancing user experience: You may concentrate on enhancing the tool's usability and user experience, for example, by introducing features that make it simpler to use or by carrying out user testing to pinpoint the tool's shortcomings.

References

1. Security Boulevard. (2022, September 7). JumpCloud Adds Decentralized Password Manager to Portfolio. https://securityboulevard.com/2022/09/jumpcloud-adds-decentralized-password-manager-to-portfolio/

2. Pradhan, A., & Dash, S. K. (2016). Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data. 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2040-2045. https://ieeexplore.ieee.org/abstract/document/7470800

3. Yan, J., & Blackwell, A. F. (2019). Usability, Security and Trust in Password Managers: A Quest for User-centric Properties and Features. Proceedings of the 11th Nordic Conference on Human-Computer Interaction, 1-12. https://www.researchgate.net/publication/333689920_Usability_security_and_trust_in_password_managers_A_quest_for_user-centric_properties_and_features

4. Debnath, B., & Sengupta, A. (2017). A Survey on Decentralized Password Management. 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-5. https://ieeexplore.ieee.org/abstract/document/8204069