UNIVERSITI TUNKU ABDUL RAHMAN
Faculty of Information and Communication Technology

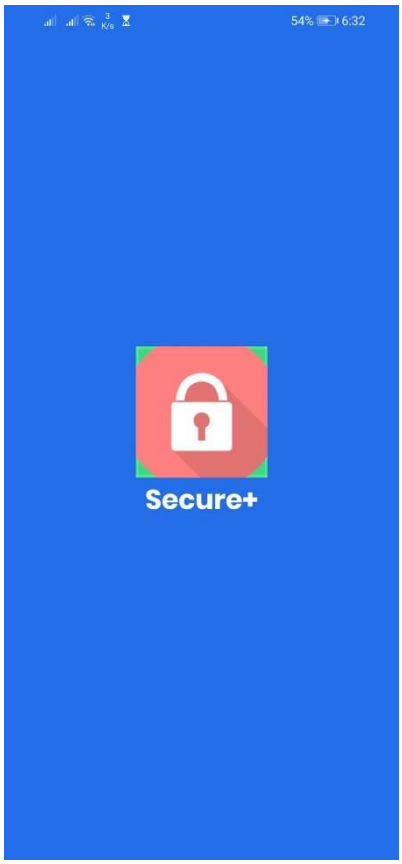
UCCD3223 Mobile Applications Development
(January 2022 Trimester)

Individual Practical Assignment

Name	**Ng Warren Cin Yen**
Student ID	**19ACB03437**
Course	**CN**
Practical Group	**P1**
Lecturer	**Mr.Tan Chiang Kang @ Thang Chiang Kang**

Marking scheme	Marks	Remarks
Correctness of output	× 3	
Design of UI	× 2.5	
User Friendliness	× 2.5	
Neat Program Documentation		
Report Format		
TOTAL		

SplashArt



This is the first UI that will presented to users. In this activity, the content view is set to activity_splash.xml and it uses Linear Layout with Image View and Text View.

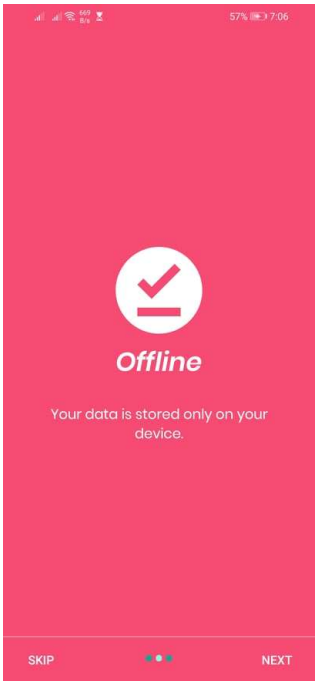
```
setContentView(R.layout.activity_splash);

//Retrieve ID
TextView password_manager = findViewById(R.id.password_manager);
password_manager.setText("Secure+");
```

The time out duration for this activity is set to 2 seconds time out duration

```
//timeout timer for splash art
private static int timeout = 2000;
```

WelcomeScreen



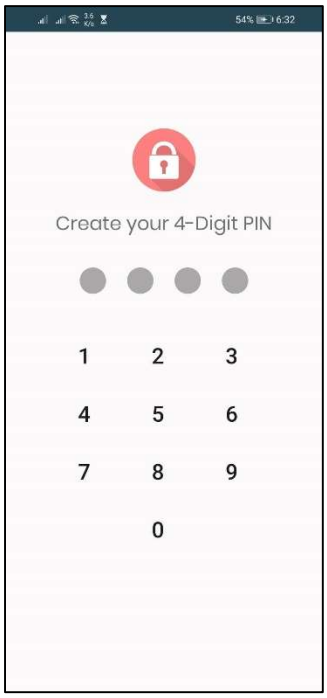
If the users are first time using the app, the app will present greeting slides to user to show the functions of this app to the user.

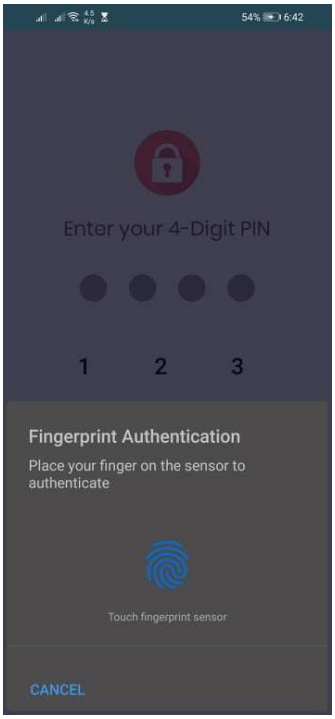
```
//Layouts of welcome slides, in order
layouts = new int[]{
        R.layout.welcome_slide1,
        R.layout.welcome_slide2,
        R.layout.welcome_slide3};
```

There are 3 welcome slides in this activity, and if the user is not the first time using the app this activity will be skipped and will directed to the PIN activity.

```
//Checking for first time launch - before calling setContentView()
prefManager = new SharedPrefManager( context: this);
if (!prefManager.isFirstTimeLaunch()) {
    launchHomeScreen();
    finish();
}
```

PIN





If user is first time user, they are required to create a 4-Digit PIN. The PIN is required every time to gain access to the app. A pin lock listener is used to get the input by user.

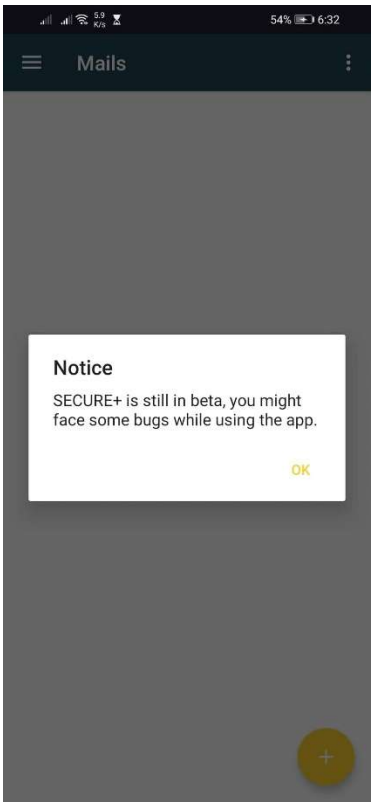
```
//Create PinLock Listener
PinLockListener mPinLockListener = new PinLockListener() {
```

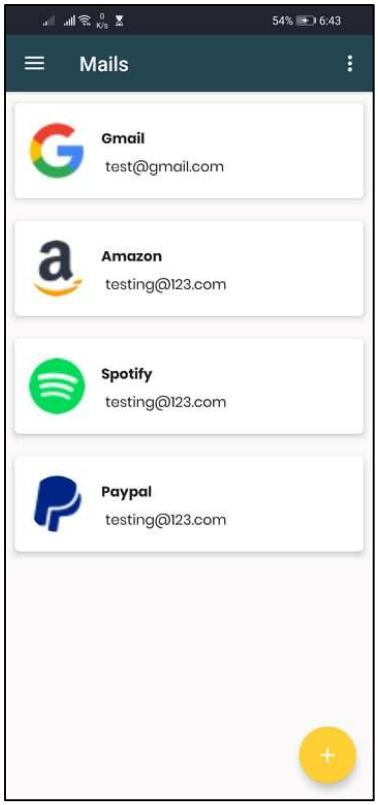
Besides that, this app also implements biometrics authentication which adds extra protection layer to the app.

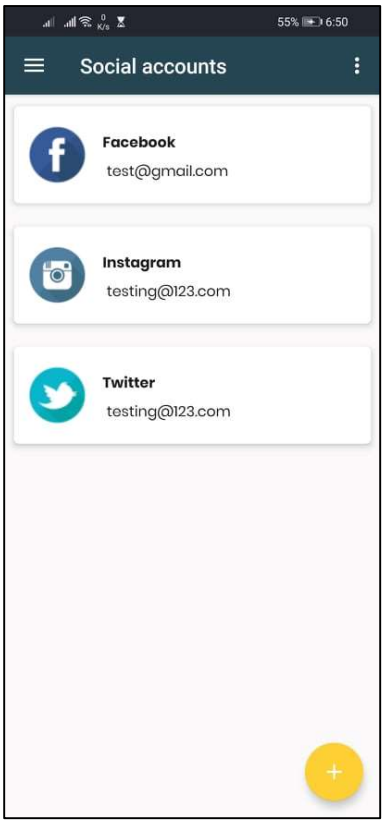
```
//onAuthenticationSucceeded is called when a fingerprint is matched successfully
@Override
public void onAuthenticationSucceeded(@NonNull BiometricPrompt.AuthenticationResult result) {
    super.onAuthenticationSucceeded(result);
    //Print a message to Logcat//
    startActivity(new Intent( packageContext: MasterLock.this, Home.class));
    finish();
      Log.d(TAG, "Fingerprint recognised successfully");
}

//onAuthenticationFailed is called when the fingerprint do not match
@Override
public void onAuthenticationFailed() {
    super.onAuthenticationFailed();
    //Print a message to Logcat//
      Log.d(TAG, "Fingerprint not recognised");
}
```

Main Page





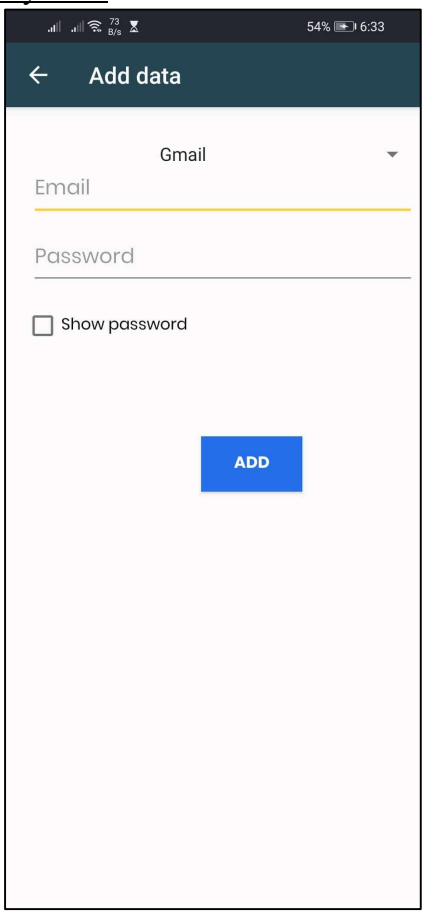


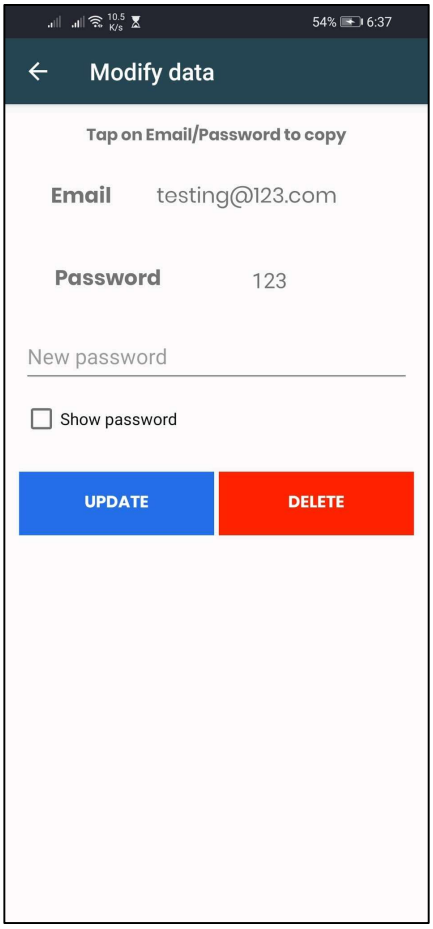
In the main page, user is allowed to add new entry by using the right bottom add button to add Mails or Social accounts.

An alert box will pop up for the first-time user uses the app.

```
//Build ALert Dialog
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder( context: this);
alertDialogBuilder.setTitle("Notice");
alertDialogBuilder.setMessage("SECURE+ is still in beta, you might face some bugs while using the app.");
```

Add/Modify Data



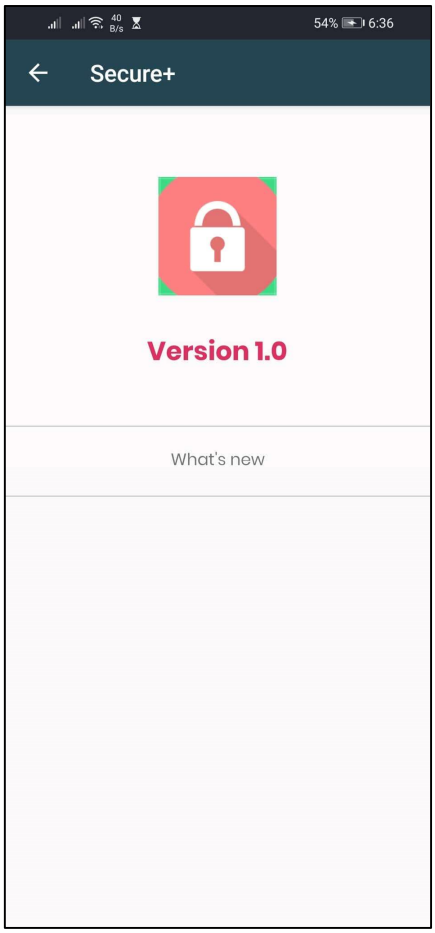


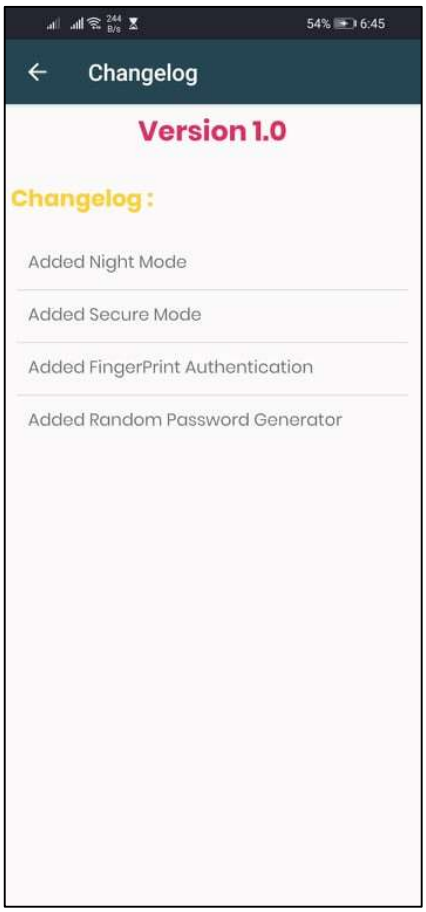
User can add or modify email-password freely just by typing in the credentials. To add, just simply type in the required data and press add button. To modify data, user can choose whether to update or delete the data. To update the password, user must type in new password and press update. To delete the credential, just simply press delete button.

There is a checkbox which will shows the user their password if the check box is selected.

```
//IF Checkbox CHECKED, reveal password; IF Checkbox UNCHECKED, hidden password
cb.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            password.setInputType(InputType.TYPE_NUMBER_VARIATION_PASSWORD);
        } else {
            password.setInputType(129);
        }
    }
});
```

About

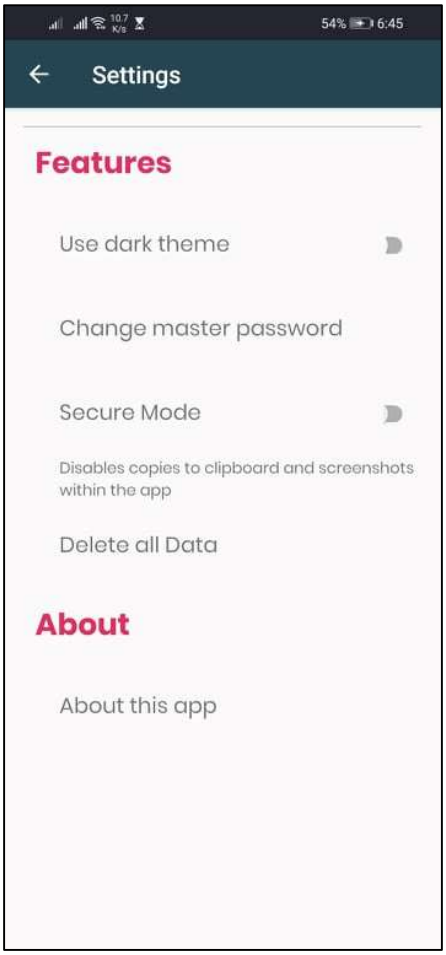




This Activity shows the features in the app. Users can get to know what the latest feature is being implemented in the app.

```
//What's new feature String
String[] whatsnew = {
        "Added Night Mode",
        "Added Secure Mode",
        "Added FingerPrint Authentication",
        "Added Random Password Generator",};
```

Settings



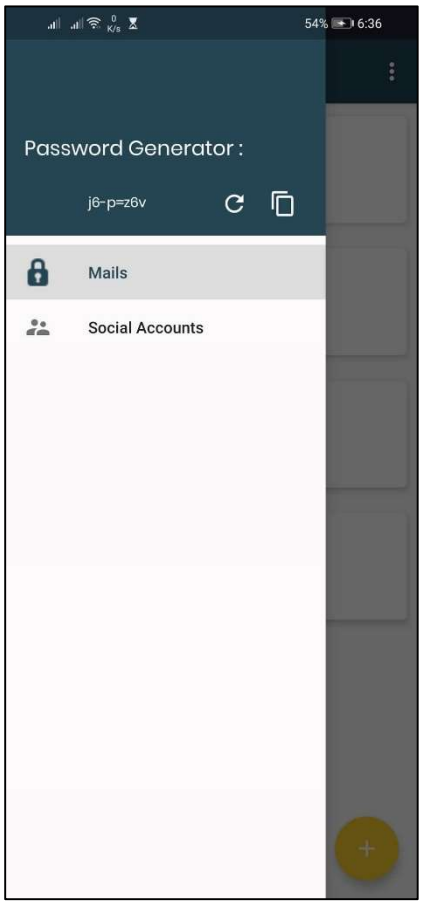
In Settings: there consists of a few features:

- Dark Theme
- Change master password
- Secure Mode (Disables copies to clipboard and screenshots within the app)
- Delete all Data
- About this app

Users can edit the settings according to their preferences

Features

Users are able to navigate through the app by using the top left drawer or top right lists in the main page to access more feature.



Random Password Generator

- Users can use the password generated randomly by the app
- By pressing REFRESH button a new password will be generated
- By pressing the COPY button, the password will be copied to clipboard and proceeds to the Add Data Activity.

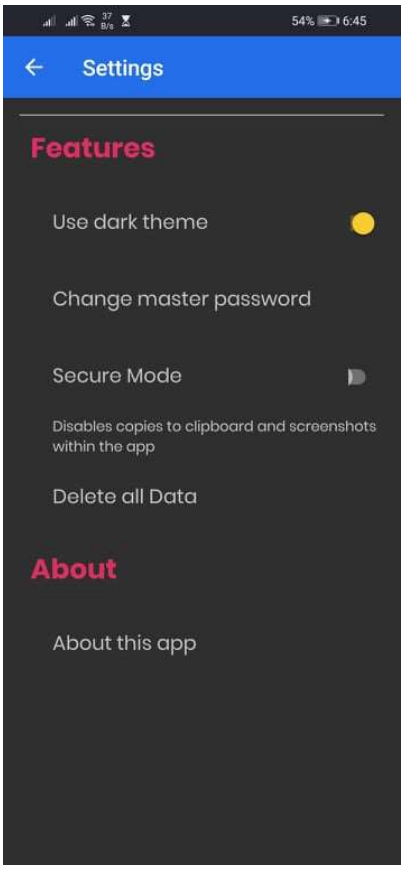
```
private String generatePassword() {

    //Creating Random object
    Random random = new Random();

    //Limits  length of the generated password
    int limit = (int) (Math.random() * 10 + 5);

    //Build String using string builder
    StringBuilder password = new StringBuilder();
    for (int itr = 0; itr < limit; itr++) {
        password.append(collection.charAt(random.nextInt(collection.length())));
    }
    return password.toString();
}
```

Source code that shows the generatePassword() functions.

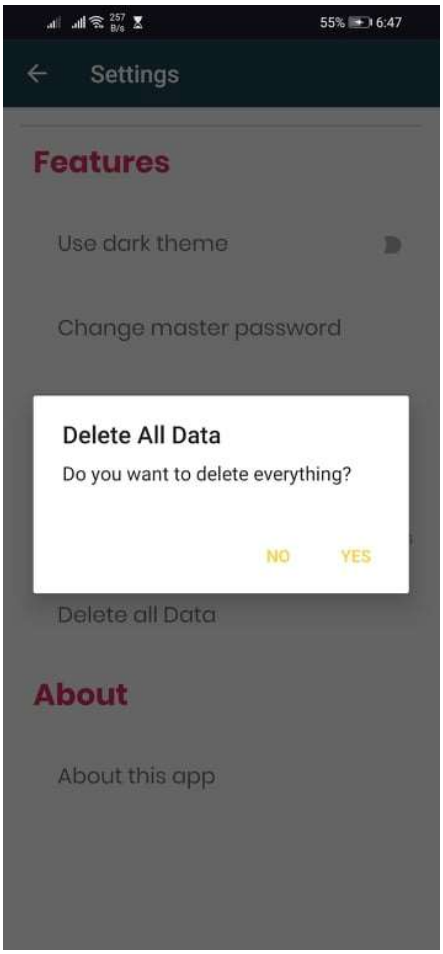


Dark Theme:

- Users can choose whether to use light/dark theme
- Adjustable through a button.

```
//Set theme mode
UIPref = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
boolean onDarkTheme = UIPref.getBoolean(PREF_DARK, b: false);
if (onDarkTheme) {
    dark_theme.setChecked(onDarkTheme);
}
```

Source code of setting the theme mode.



Delete all data

- Users can delete all of their data with just one button
- A confirmation dialog box will pop up before deleting all the data

```
@Override
public void onClick(DialogInterface arg0, int arg1) {
    MailViewModel passwordViewModel = new MailViewModel(getApplication());
    progressBar.setVisibility(View.VISIBLE);
    //deletes all data
    passwordViewModel.deleteAllNotes();
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putBoolean(NO_DATA, b: false).apply();
    progressBar.setVisibility(View.GONE);
    Toast.makeText(getApplicationContext(), text: "Deleted", Toast.LENGTH_SHORT).show();
```

Source code of Onclick of yes button, all of the data will be deleted.

Change 4-Digit PIN

- Users can change the 4-Digit PIN anytime
- Good for security
- Does not have worry about the exposure of the PIN

```
public void changePasswordToPIN(View view) {
    Intent intent = new Intent(getApplicationContext(), ChangePassword.class);
    intent.putExtra(ChangePassword.EXTRA_TYPE_PASS, TYPE_PASS_1);
    startActivity(intent);
}
```

Source Code for changing 4-Digit PIN

Secure Mode:

- Disable copies of Strings to clipboard
- Disable the ability to screenshot in the app

```
//SecureCodeMode functions
private void secureCodeMode(boolean state) {
    final SharedPreferences.Editor editor = sharedPreferences.edit();
    if (state) {

        //remove copy to clipboard and screenshot ability
        editor.putBoolean(PREF_KEY_SCM_COPY,  b: false).apply();
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE, WindowManager.LayoutParams.FLAG_SECURE);
        Toast.makeText(getApplicationContext(),  text: "Success. Restart app to apply changes", Toast.LENGTH_LONG).show();
    } else {

        //set copy to clipboard and screenshot ability
        editor.putBoolean(PREF_KEY_SCM_SCREENSHOTS,  b: true).apply();
        getWindow().clearFlags(WindowManager.LayoutParams.FLAG_SECURE);
        Toast.makeText(getApplicationContext(),  text: "Secure code mode is inactive", Toast.LENGTH_LONG).show();
    }
}
```

AES encryption/decryption:

- Encrypt the email and password data pairs

```
// AES encryption process, encrypt email/password
try {
    String encEmail = AES_Utils.encrypt(text_email, HASH);
    String encPass = AES_Utils.encrypt(text_password, HASH);
    intent.putExtra(EXTRA_EMAIL, encEmail);
    intent.putExtra(EXTRA_ENCRYPT, encPass);
} catch (Exception e) {
    e.printStackTrace();
}
setResult(RESULT_OK, intent);
finish();
```

Appendix

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="my.edu.utar.pwmanager">

    <uses-feature
        android:name="android.hardware.fingerprint"
        android:required="false" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.USE_BIOMETRIC" />

    <application
        android:allowBackup="false"
        android:icon="@mipmap/ic_launcher"
        android:installLocation="internalOnly"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name="my.edu.utar.pwmanager.SplashArt"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="my.edu.utar.pwmanager.ChangePassword" />
        <activity
            android:name="my.edu.utar.pwmanager.About"
            android:label="Changelog" />
        <activity
            android:name="my.edu.utar.pwmanager.AddEntry"
            android:label="Add data" />
        <activity
            android:name="my.edu.utar.pwmanager.ModifyEntry"
            android:label="Modify data" />
        <activity android:name="my.edu.utar.pwmanager.Help" />
        <activity
            android:name="my.edu.utar.pwmanager.Home"
            android:label="@string/title_activity_home"
            android:launchMode="singleTop"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name="my.edu.utar.pwmanager.Settings"
            android:label="@string/title_activity_settings" />
        <activity
            android:name="my.edu.utar.pwmanager.MasterLock"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
```

```
            android:name="my.edu.utar.pwmanager.WelcomeScreen"
            android:theme="@style/AppTheme.NoActionBar">

        </activity>

        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>

</manifest>
```

PwClass.java

```
package my.edu.utar.pwmanager.classFramework;

import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity(tableName = "entry_table")
public class PwClass {

    @PrimaryKey(autoGenerate = true)
    private int id;
    private String provider;
    private String providerName;
    private String email;
    private String cat;

    public PwClass(String provider, String providerName, String email, String
cat) {
        this.provider = provider;
        this.providerName = providerName;
        this.email = email;
        this.cat = cat;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getProvider() {
        return provider;
    }

    public String getProviderName() {
        return providerName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getCat() {
        return cat;
    }
}
```

PwDAO.java

```java
//Declaring abstract for functions within Database

package my.edu.utar.pwmanager.database;

import androidx.lifecycle.LiveData;
import androidx.room.Dao;
import androidx.room.Delete;
import androidx.room.Insert;
import androidx.room.Query;
import androidx.room.Update;

import my.edu.utar.pwmanager.classFramework.PwClass;

import java.util.List;

@Dao
public interface PwDAO {

    @Insert
    void insert(PwClass pwClass);

    @Update
    void update(PwClass pwClass);

    @Delete
    void delete(PwClass pwClass);

    @Query("DELETE FROM entry_table")
    void deleteAllNotes();


    @Query("SELECT * FROM entry_table")
    LiveData<List<PwClass>> getAllCreds();

    @Query("SELECT * FROM entry_table WHERE provider = 'mail'")
    LiveData<List<PwClass>> getAllMails();

    @Query("SELECT * FROM entry_table WHERE provider = 'social'")
    LiveData<List<PwClass>> getAllSocial();
}
```

PwDB.java

```
package my.edu.utar.pwmanager.database;

import android.content.Context;
import android.os.AsyncTask;
import androidx.annotation.NonNull;
import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;
import androidx.sqlite.db.SupportSQLiteDatabase;
import my.edu.utar.pwmanager.classFramework.PwClass;

@Database(entities = {PwClass.class}, version = 5)
public abstract class PwDB extends RoomDatabase {

    private static PwDB instance;

    public abstract PwDAO pwDao();

    public static synchronized PwDB getInstance(Context context) {
        if (instance == null) {
            instance = Room.databaseBuilder(context.getApplicationContext(),
                    PwDB.class, "PwDatabase")
                    .setJournalMode(JournalMode.TRUNCATE)
                    .fallbackToDestructiveMigration()
                    .addCallback(roomCallback)
                    .build();
        }
        return instance;
    }

    private static RoomDatabase.Callback roomCallback = new
RoomDatabase.Callback() {
        @Override
        public void onCreate(@NonNull SupportSQLiteDatabase db) {
            super.onCreate(db);
            new PopulateDbAsyncTask(instance).execute();
        }
    };

    private static class PopulateDbAsyncTask extends AsyncTask<Void, Void,
Void> {
        private PwDAO pwDAO;

        private PopulateDbAsyncTask(PwDB db) {
            pwDAO = db.pwDao();
        }

        @Override
        protected Void doInBackground(Void... voids) {
            return null;
        }
    }
}
```

PwRepos.java

```
package my.edu.utar.pwmanager.database;

import android.app.Application;
import android.os.AsyncTask;
import androidx.lifecycle.LiveData;
import my.edu.utar.pwmanager.classFramework.PwClass;
import java.util.List;

public class PwRepos {
    private PwDAO pwDAO;
    private LiveData<List<PwClass>> allEntries, mailEntries, socialEntries;

    //Repository class for getting the entries
    public PwRepos(Application application) {
        PwDB database = PwDB.getInstance(application);
        pwDAO = database.pwDao();
        allEntries = pwDAO.getAllCreds();
        mailEntries = pwDAO.getAllMails();
        socialEntries = pwDAO.getAllSocial();
    }

    //insert function
    public void insert(PwClass pwClass) {
        new InsertEntry(pwDAO).execute(pwClass);
    }

    //update function
    public void update(PwClass pwClass) {
        new UpdateEntry(pwDAO).execute(pwClass);
    }

    //delete function
    public void delete(PwClass pwClass) {
        new DeleteEntry(pwDAO).execute(pwClass);
    }

    //delete all function
    public void deleteAllNotes() {
        new DeleteAllEntry(pwDAO).execute();
    }

    public LiveData<List<PwClass>> getAllNotes() {
        return allEntries;
    }

    public LiveData<List<PwClass>> getAllMails() {
        return mailEntries;
    }

    public LiveData<List<PwClass>> getAllSocial() {
        return socialEntries;
    }
```

```
//Insert set of entry into database
    private static class InsertEntry extends AsyncTask<PwClass, Void, Void> {
        private PwDAO pwDAO;
        private InsertEntry(PwDAO pwDAO) {
            this.pwDAO = pwDAO;
        }

        @Override
        protected Void doInBackground(PwClass... pwClass) {
            pwDAO.insert(pwClass[0]);
            return null;
        }
    }

    //Update the data of set of entry in the database
    private static class UpdateEntry extends AsyncTask<PwClass, Void, Void> {
        private PwDAO pwDAO;

        private UpdateEntry(PwDAO pwDAO) {
            this.pwDAO = pwDAO;
        }
        @Override
        protected Void doInBackground(PwClass... pwClass) {
            pwDAO.update(pwClass[0]);
            return null;
        }
    }

    //Delete the data of selected set of entry in the database
    private static class DeleteEntry extends AsyncTask<PwClass, Void, Void> {
        private PwDAO pwDAO;

        private DeleteEntry(PwDAO pwCredDao) {
            this.pwDAO = pwCredDao;
        }
        @Override
        protected Void doInBackground(PwClass... pwClass) {
            pwDAO.delete(pwClass[0]);
            return null;
        }
    }

    //Delete all data in the database
    private static class DeleteAllEntry extends AsyncTask<Void, Void, Void> {
        private PwDAO pwDAO;

        private DeleteAllEntry(PwDAO pwDAO) {
            this.pwDAO = pwDAO;
        }

        @Override
        protected Void doInBackground(Void... voids) {
            pwDAO.deleteAllNotes();
            return null;
        }
    }
}
```

HomeFragment.java

```
package my.edu.utar.pwmanager.Fragments.homepage;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;
import my.edu.utar.pwmanager.R;
import my.edu.utar.pwmanager.Utilities.RecyclerViewAdapter;

public class HomeFragment extends Fragment {

    String PREF_NAME = "appEssentials";
    SharedPreferences sharedPreferences;
    String PREF_KEY_SECURE_CORE_MODE = "SECURE_CORE";
    private HomeViewModel homeViewModel;
    private RecyclerViewAdapter adapter;

    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        homeViewModel = ViewModelProviders.of(this).get(HomeViewModel.class);

        View root = inflater.inflate(R.layout.fragment_home, container,
false);

        final TextView textView = root.findViewById(R.id.text_home);

        homeViewModel.getText().observe(getViewLifecycleOwner(), new
Observer<String>() {
            @Override
            public void onChanged(@Nullable String s) {
                textView.setText(s);
            }
        });
        sharedPreferences =
this.getActivity().getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);

        if (sharedPreferences.getBoolean(PREF_KEY_SECURE_CORE_MODE, false)) {

getActivity().getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
WindowManager.LayoutParams.FLAG_SECURE);
        }
        return root;
    }
}
```

HomeViewModel.java

```
package my.edu.utar.pwmanager.Fragments.homepage;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

public class HomeViewModel extends ViewModel {

    private MutableLiveData<String> Text;

    public HomeViewModel() {
        Text = new MutableLiveData<>();
        Text.setValue("Home fragment");
    }

    public LiveData<String> getText() {
        return Text;
    }

}
```

MailFragment.java

```
package my.edu.utar.pwmanager.Fragments.mail;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.ImageButton;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import my.edu.utar.pwmanager.AddEntry;
import my.edu.utar.pwmanager.ModifyEntry;
import my.edu.utar.pwmanager.R;
import my.edu.utar.pwmanager.Utilities.RecyclerViewAdapter;
import my.edu.utar.pwmanager.classFramework.PwClass;
import my.edu.utar.pwmanager.database.PwRepos;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import java.util.List;
import static android.app.Activity.RESULT_OK;


// Mails
public class MailFragment extends Fragment {

    String PREF_NAME = "Settings";
    String PREF_KEY_SECURE_CORE_MODE = "SECURE_CORE";

    public static final String NO_DATA = "NO DATA";
    private static final int ADD_RECORD = 1;
    private static final int MODIFY_RECORD = 2;
    private static final int DELETE_RECORD = 3;

    private static final String PROVIDER = "mail";
    private static final String TAG = "P FRAG ";
    boolean status = false;

    private TextView empty;
    private MailViewModel passwordViewModel;
    private PwRepos repository;

    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
```

```
        View root = inflater.inflate(R.layout.fragment_password, container,
false);
        ProgressBar progressBar = root.findViewById(R.id.progress_bar);
        FloatingActionButton fab = root.findViewById(R.id.fab);

        empty = root.findViewById(R.id.empty);

        SharedPreferences sharedPreferences =
this.getActivity().getSharedPreferences(PROVIDER, Context.MODE_PRIVATE);

        SharedPreferences sp =
this.getActivity().getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
        if (sp.getBoolean(PREF_KEY_SECURE_CORE_MODE, false)) {
            try {
                ImageButton copyImage = root.findViewById(R.id.copy);
                copyImage.setEnabled(false);
            } catch (Exception e) {
                e.getStackTrace();
            }

getActivity().getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
WindowManager.LayoutParams.FLAG_SECURE);
        }

        status = sharedPreferences.getBoolean(NO_DATA, false);
        if (status) {
            empty.setVisibility(View.GONE);
        } else {
            empty.setText(NO_DATA);
        }

        RecyclerView recyclerView = root.findViewById(R.id.recycler_view);
        recyclerView.setLayoutManager(new
LinearLayoutManager(this.getContext()));
        recyclerView.setHasFixedSize(true);

        //Declare viewAdapter
        final RecyclerViewAdapter viewAdapter = new RecyclerViewAdapter();
        recyclerView.setAdapter(viewAdapter);


        //Declare ViewModel for pw
        passwordViewModel = new
ViewModelProvider(this).get(MailViewModel.class);

        //Retrieving email entries
        passwordViewModel.getAllMails().observe(getViewLifecycleOwner(), new
Observer<List<PwClass>>() {
            @Override
            public void onChanged(List<PwClass> pwCreds) {
                viewAdapter.setCreds(pwCreds);
            }
        });


        //Defining the gestures action on screen
```

```
        ItemTouchHelper.SimpleCallback itemTouchHelperCallback = new
ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT) {
            @Override
            public boolean onMove(@NonNull RecyclerView recyclerView,
@NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder
target) {
                return false;
            }

            @Override
            public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder,
int direction) {

passwordViewModel.delete(viewAdapter.getCredAt(viewHolder.getAdapterPosition(
)));
                Toast.makeText(getContext(), "Entry deleted",
Toast.LENGTH_SHORT).show();
            }
        };
        new
ItemTouchHelper(itemTouchHelperCallback).attachToRecyclerView(recyclerView);


        //Declaring next activity using intent
        viewAdapter.setOnItemClickListener(new
RecyclerViewAdapter.onItemClickListener() {
            @Override
            public void onItemClick(PwClass pwClass) {
                Intent intent = new Intent(getActivity(), ModifyEntry.class);
                intent.putExtra(ModifyEntry.EXTRA_ID, pwClass.getId());
                intent.putExtra(ModifyEntry.EXTRA_PROVIDER_NAME,
pwClass.getProviderName());
                intent.putExtra(ModifyEntry.EXTRA_EMAIL, pwClass.getEmail());
                intent.putExtra(ModifyEntry.EXTRA_ENCRYPT, pwClass.getCat());
                startActivityForResult(intent, MODIFY_RECORD);
            }
        });


        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getContext(), AddEntry.class);
                intent.putExtra(AddEntry.EXTRA_PROVIDER, PROVIDER);
                startActivityForResult(intent, ADD_RECORD);

            }
        });
        return root;
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        if (requestCode == ADD_RECORD && resultCode == RESULT_OK) {

            String providerName =
```

```
data.getStringExtra(AddEntry.EXTRA_PROVIDER_NAME);
            String enc_passwd = data.getStringExtra(AddEntry.EXTRA_ENCRYPT);
            String enc_email = data.getStringExtra(AddEntry.EXTRA_EMAIL);

            PwClass pwClass = new PwClass(PROVIDER, providerName, enc_email,
enc_passwd);

            //Showing "No data" or not on activity if list is empty
            SharedPreferences sharedPreferences =
this.getActivity().getSharedPreferences(PROVIDER, Context.MODE_PRIVATE);
            sharedPreferences.edit().putBoolean(NO_DATA, true).apply();

            empty.setVisibility(View.GONE);
            passwordViewModel.insert(pwClass);
            //Shows status of the data after succesfully saved
            Toast.makeText(getContext(), "Saved", Toast.LENGTH_SHORT).show();

        } else if (requestCode == MODIFY_RECORD && resultCode == RESULT_OK) {
            int id = data.getIntExtra(ModifyEntry.EXTRA_ID, -1);

            //If cannot update data
            if (id == -1) {
                Toast.makeText(getContext(), "Cannot be updated!",
Toast.LENGTH_LONG).show();
                return;
            }

            String providerName =
data.getStringExtra(ModifyEntry.EXTRA_PROVIDER_NAME);
            String enc_passwd =
data.getStringExtra(ModifyEntry.EXTRA_ENCRYPT);
            String enc_email = data.getStringExtra(ModifyEntry.EXTRA_EMAIL);

            PwClass pwClass = new PwClass(PROVIDER, providerName, enc_email,
enc_passwd);

            //Shows status of the data after modify
            pwClass.setId(id);
            if (!data.getBooleanExtra(ModifyEntry.EXTRA_DELETE, false)) {
                // (TAG, "Provider: " + PROVIDER + " EMAIL: " + enc_email + "
ENC_DATA: " + enc_passwd);
                passwordViewModel.update(pwClass);
                Toast.makeText(getContext(), "Updated",
Toast.LENGTH_SHORT).show();
            } else {
                passwordViewModel.delete(pwClass);
                Toast.makeText(getContext(), "Deleted",
Toast.LENGTH_SHORT).show();
            }
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

MailViewModel.java

```
package my.edu.utar.pwmanager.Fragments.mail;

import android.app.Application;

import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;

import my.edu.utar.pwmanager.classFramework.PwClass;
import my.edu.utar.pwmanager.database.PwRepos;

import java.util.List;

//Mail Fragments
public class MailViewModel extends AndroidViewModel {
    private PwRepos repository;
    private LiveData<List<PwClass>> allCreds, mailCreds;

    public MailViewModel(@NonNull Application application) {
        super(application);
        repository = new PwRepos(application);
        allCreds = repository.getAllNotes();
        mailCreds = repository.getAllMails();

    }

    public void insert(PwClass pwClass) {
        repository.insert(pwClass);
    }

    public void update(PwClass pwClass) {
        repository.update(pwClass);
    }

    public void delete(PwClass pwClass) {
        repository.delete(pwClass);
    }

    public void deleteAllNotes() {
        repository.deleteAllNotes();
    }

    public LiveData<List<PwClass>> getAllCreds() {
        return allCreds;
    }

    public LiveData<List<PwClass>> getAllMails() {
        return mailCreds;
    }
}
```

SocialFragment.java

```
package my.edu.utar.pwmanager.Fragments.social;

import android.app.Application;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.ImageButton;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import my.edu.utar.pwmanager.AddEntry;
import my.edu.utar.pwmanager.ModifyEntry;
import my.edu.utar.pwmanager.R;
import my.edu.utar.pwmanager.Utilities.RecyclerViewAdapter;
import my.edu.utar.pwmanager.classFramework.PwClass;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.List;

import static android.app.Activity.RESULT_OK;

// Social
public class SocialFragment extends Fragment {

    String PREF_NAME = "Settings";
    String PREF_KEY_SECURE_CORE_MODE = "SECURE_CORE";

    private static final String TAG = "S_FRAG";
    private static final String NO_DATA = "NO DATA";
    private static final int ADD_RECORD = 1;
    private static final int MODIFY_RECORD = 2;
    private static final int DELETE_RECORD = 3;
    public static final String PROVIDER = "social";

    private static Application application;
    boolean status = false;

    TextView empty;
    SocialViewModel socialViewModel;
```

```
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {

        View root = inflater.inflate(R.layout.social_fragment, container,
false);

        ProgressBar progressBar = root.findViewById(R.id.progress_bar);
        FloatingActionButton fab = root.findViewById(R.id.fab);

        empty = root.findViewById(R.id.empty);

        SharedPreferences sharedPreferences =
this.getActivity().getSharedPreferences(PROVIDER, Context.MODE_PRIVATE);
        SharedPreferences sp =
this.getActivity().getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);

        if (sp.getBoolean(PREF_KEY_SECURE_CORE_MODE, false)) {
            try {
                ImageButton copyImage = root.findViewById(R.id.copy);
                copyImage.setEnabled(false);
            } catch (Exception e) {
                e.getStackTrace();
            }

getActivity().getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
WindowManager.LayoutParams.FLAG_SECURE);
        }

        status = sharedPreferences.getBoolean(NO_DATA, false);
        if (status) {
            empty.setVisibility(View.GONE);
        } else {
            empty.setText(NO_DATA);
        }

        RecyclerView recyclerView = root.findViewById(R.id.recycler_view);
        recyclerView.setLayoutManager(new
LinearLayoutManager(this.getContext()));
        recyclerView.setHasFixedSize(true);

        //Declare viewAdapter
        final RecyclerViewAdapter viewAdapter = new RecyclerViewAdapter();
        recyclerView.setAdapter(viewAdapter);

        //Declare ViewModel for pw
        socialViewModel = new
ViewModelProvider(this).get(SocialViewModel.class);

        //Retrieving social entries
        socialViewModel.getAllSocial().observe(getViewLifecycleOwner(), new
Observer<List<PwClass>>() {
            @Override
            public void onChanged(List<PwClass> pwClass) {
                viewAdapter.setCreds(pwClass);
            }
        });
```

```
        //Defining the gestures action on screen
        ItemTouchHelper.SimpleCallback itemTouchHelperCallback = new
ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT) {
            @Override
            public boolean onMove(@NonNull RecyclerView recyclerView,
@NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder
target) {
                return false;
            }

            @Override
            public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder,
int direction) {

socialViewModel.delete(viewAdapter.getCredAt(viewHolder.getAdapterPosition())
);
                Toast.makeText(getContext(), "Entry deleted",
Toast.LENGTH_SHORT).show();
            }
        };

        new
ItemTouchHelper(itemTouchHelperCallback).attachToRecyclerView(recyclerView);

        //Declaring next activity using intent
        viewAdapter.setOnItemClickListener(new
RecyclerViewAdapter.onItemClickListener() {
            @Override
            public void onItemClick(PwClass pwCred) {
                Log.d(TAG, "Onclick");
                Intent intent = new Intent(getActivity(), ModifyEntry.class);
                intent.putExtra(ModifyEntry.EXTRA_ID, pwCred.getId());
                intent.putExtra(ModifyEntry.EXTRA_PROVIDER_NAME,
pwCred.getProviderName());
                intent.putExtra(ModifyEntry.EXTRA_EMAIL, pwCred.getEmail());
                intent.putExtra(ModifyEntry.EXTRA_ENCRYPT, pwCred.getCat());
                startActivityForResult(intent, MODIFY_RECORD);
            }
        });
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getContext(), AddEntry.class);
                intent.putExtra(AddEntry.EXTRA_PROVIDER, PROVIDER);
                startActivityForResult(intent, ADD_RECORD);
            }
        });
        return root;
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        if (requestCode == ADD_RECORD && resultCode == RESULT_OK) {

            String providerName =
data.getStringExtra(ModifyEntry.EXTRA_PROVIDER_NAME);
```

```
            String enc_passwd = data.getStringExtra(AddEntry.EXTRA_ENCRYPT);
            String enc_email = data.getStringExtra(AddEntry.EXTRA_EMAIL);

            PwClass pwCred = new PwClass(PROVIDER, providerName, enc_email,
enc_passwd);
            Log.d(TAG, "Provider: " + PROVIDER + " EMAIL: " + enc_email + "
ENC_DATA: " + enc_passwd);

            //Showing "No data" or not on activity if list is empty
            SharedPreferences sharedPreferences =
this.getActivity().getSharedPreferences(PROVIDER, Context.MODE_PRIVATE);
            sharedPreferences.edit().putBoolean(NO_DATA, true).apply();

            empty.setVisibility(View.GONE);
            socialViewModel.insert(pwCred);

            //Shows status of the data after succesfully saved
            Toast.makeText(getContext(), "Saved", Toast.LENGTH_SHORT).show();

        } else if (requestCode == MODIFY_RECORD && resultCode == RESULT_OK) {
            int id = data.getIntExtra(ModifyEntry.EXTRA_ID, -1);

            //If cannot update data
            if (id == -1) {
                Toast.makeText(getContext(), "Cannot be updated!",
Toast.LENGTH_LONG).show();
                return;
            }

            String providerName =
data.getStringExtra(ModifyEntry.EXTRA_PROVIDER_NAME);
            String enc_passwd =
data.getStringExtra(ModifyEntry.EXTRA_ENCRYPT);
            String enc_email = data.getStringExtra(ModifyEntry.EXTRA_EMAIL);

            PwClass pwClass = new PwClass(PROVIDER, providerName, enc_email,
enc_passwd);

            //Shows status of the data after modify
            pwClass.setId(id);
            if (!data.getBooleanExtra(ModifyEntry.EXTRA_DELETE, false)) {
                Log.d(TAG, "Provider: " + PROVIDER + " EMAIL: " + enc_email +
" ENC_DATA: " + enc_passwd);
                socialViewModel.update(pwClass);
                Toast.makeText(getContext(), "Updated",
Toast.LENGTH_SHORT).show();

            } else {
                socialViewModel.delete(pwClass);
                Toast.makeText(getContext(), "Deleted",
Toast.LENGTH_SHORT).show();
            }

        }
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

SocialViewModel.java

```java
package my.edu.utar.pwmanager.Fragments.social;

import android.app.Application;
import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;

import my.edu.utar.pwmanager.classFramework.PwClass;
import my.edu.utar.pwmanager.database.PwRepos;

import java.util.List;

//Social Fragments
public class SocialViewModel extends AndroidViewModel {
    private PwRepos repository;
    private LiveData<List<PwClass>> allCreds, mailCreds;

    public SocialViewModel(@NonNull Application application) {
        super(application);
        repository = new PwRepos(application);
        mailCreds = repository.getAllSocial();

    }

    public void insert(PwClass pwClass) {
        repository.insert(pwClass);
    }

    public void update(PwClass pwClass) {
        repository.update(pwClass);
    }

    public void delete(PwClass pwClass) {
        repository.delete(pwClass);
    }

    public void deleteAllNotes() {
        repository.deleteAllNotes();
    }

    public LiveData<List<PwClass>> getAllCreds() {
        return allCreds;
    }

    public LiveData<List<PwClass>> getAllSocial() {
        return mailCreds;
    }
}
```

AES_Utils.java

```
//AES Utilities for String encryption/decryption

package my.edu.utar.pwmanager.Utilities;

import java.nio.charset.StandardCharsets;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

public class AES_Utils {

    public static String encrypt(String cleartext, String keyValue)
            throws Exception {
        byte[] rawKey = getRawKey(keyValue);
        byte[] result = encrypt(rawKey, cleartext.getBytes());
        return toHex(result);
    }

    public static String decrypt(String encrypted, String keyValue)
            throws Exception {
        byte[] enc = toByte(encrypted);
        byte[] result = decrypt(enc, keyValue);
        return new String(result);
    }

    private static byte[] getRawKey(String keyValueString) throws Exception {
        byte[] keyValue = keyValueString.getBytes(StandardCharsets.UTF_8);
        SecretKey key = new SecretKeySpec(keyValue, "AES");
        byte[] raw = key.getEncoded();
        return raw;
    }

    private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception
{
        SecretKey skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(clear);
        return encrypted;
    }

    private static byte[] decrypt(byte[] encrypted, String keyValueString)
            throws Exception {
        byte[] keyValue = keyValueString.getBytes(StandardCharsets.UTF_8);
        SecretKey skeySpec = new SecretKeySpec(keyValue, "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
        byte[] decrypted = cipher.doFinal(encrypted);
        return decrypted;
    }

    public static byte[] toByte(String hexString) {
        int len = hexString.length() / 2;
```

```
        byte[] result = new byte[len];
        for (int i = 0; i < len; i++)
            result[i] = Integer.valueOf(hexString.substring(2 * i, 2 * i +
2),
                    16).byteValue();
        return result;
    }

    public static String toHex(byte[] buf) {
        if (buf == null)
            return "";
        StringBuffer result = new StringBuffer(2 * buf.length);
        for (int i = 0; i < buf.length; i++) {
            appendHex(result, buf[i]);
        }
        return result.toString();
    }

    private final static String HEX = "0123456789ABCDEF";

    private static void appendHex(StringBuffer sb, byte b) {
        sb.append(HEX.charAt((b >> 4) & 0x0f)).append(HEX.charAt(b & 0x0f));
    }
}
```

RecyclerViewAdapter.java

```
package my.edu.utar.pwmanager.Utilities;

import android.content.Context;
import android.content.SharedPreferences;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;
import androidx.security.crypto.EncryptedSharedPreferences;
import androidx.security.crypto.MasterKey;

import com.himanshurawat.hasher.HashType;
import com.himanshurawat.hasher.Hasher;

import org.jetbrains.annotations.NotNull;

import java.util.ArrayList;
import java.util.List;

import my.edu.utar.pwmanager.R;
import my.edu.utar.pwmanager.classFramework.PwClass;

public class RecyclerViewAdapter extends
RecyclerView.Adapter<RecyclerViewAdapter.viewHolder> {
    private static final String PREFS_NAME = "appEssentials";
    private List<PwClass> credsList = new ArrayList<>();
    private onItemClickListener listener;
    MasterKey masterKey = null;
    SharedPreferences sharedPreferences = null;
    String sha;

    @NotNull
    @Override
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.list_items, parent, false);
        Context context = parent.getContext();
        try {
            //x.security
            masterKey = new MasterKey.Builder(context,
MasterKey.DEFAULT_MASTER_KEY_ALIAS)
                    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
                    .build();
            //init sharedPef
            sharedPreferences = EncryptedSharedPreferences.create(
                    context,
                    PREFS_NAME,
                    masterKey,

EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
```

```
EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
            );
        } catch (Exception e) {
            e.printStackTrace();
        }
        sha = sharedPreferences.getString("HASH", "0");
        return new viewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, int position) {
        PwClass creds = credsList.get(position);
        holder.provider.setText(creds.getProviderName());
        switch (creds.getProviderName()) {
            case "Amazon":
                holder.providerImage.setImageResource(R.drawable.amazon);
                break;
            case "Apple":
                holder.providerImage.setImageResource(R.drawable.apple);
                break;
            case "Facebook":
                holder.providerImage.setImageResource(R.drawable.facebook);
                break;
            case "Flickr":
                holder.providerImage.setImageResource(R.drawable.flickr);
                break;
            case "Foursquare":
                holder.providerImage.setImageResource(R.drawable.foursquare);
                break;
            case "Github":
                holder.providerImage.setImageResource(R.drawable.github);
                break;
            case "Gmail":
                holder.providerImage.setImageResource(R.drawable.google);
                break;
            case "Instagram":
                holder.providerImage.setImageResource(R.drawable.instagram);
                break;
            case "Linkedin":
                holder.providerImage.setImageResource(R.drawable.linkedin);
                break;
            case "Medium":
                holder.providerImage.setImageResource(R.drawable.medium);
                break;
            case "Paypal":
                holder.providerImage.setImageResource(R.drawable.paypal);
                break;
            case "Pinterest":
                holder.providerImage.setImageResource(R.drawable.pinterest);
                break;
            case "Reddit":
                holder.providerImage.setImageResource(R.drawable.reddit);
                break;
            case "Skype":
                holder.providerImage.setImageResource(R.drawable.skype);
                break;
```

```
            case "Slack":
                holder.providerImage.setImageResource(R.drawable.slack);
                break;
            case "Snapchat":
                holder.providerImage.setImageResource(R.drawable.snapchat);
                break;
            case "Spotify":
                holder.providerImage.setImageResource(R.drawable.spotify);
                break;
            case "Stackoverflow":

holder.providerImage.setImageResource(R.drawable.stackoverflow);
                break;
            case "Tinder":
                holder.providerImage.setImageResource(R.drawable.tinder);
                break;
            case "Trello":
                holder.providerImage.setImageResource(R.drawable.trello);
                break;
            case "Tumblr":
                holder.providerImage.setImageResource(R.drawable.tumblr);
                break;
            case "Twitter":
                holder.providerImage.setImageResource(R.drawable.twitter);
                break;
            case "Wordpress":
                holder.providerImage.setImageResource(R.drawable.wordpress);
                break;
            case "Yahoo":
                holder.providerImage.setImageResource(R.drawable.yahoo);
                break;
            default:
                holder.providerImage.setImageResource(R.drawable.google);
                break;
        }
        try {
            String keyValue = Hasher.Companion.hash(sha, HashType.MD5);
            String dec = creds.getEmail();
            String decEmail = AES_Utils.decrypt(dec, keyValue);
            holder.cat1.setText(decEmail);
        } catch (Exception e) {
            e.printStackTrace();
        }
        //holder.cat1.setText(creds.getEmail());
        //holder.cat2.setText(creds.getCat2());
    }

    @Override
    public int getItemCount() {
        return credsList.size();
    }

    public void setCreds(List<PwClass> credsList) {
        this.credsList = credsList;
        notifyDataSetChanged();
    }
```

```
    public PwClass getCredAt(int position) {
        return credsList.get(position);
    }

    class viewHolder extends RecyclerView.ViewHolder {
        private TextView provider, cat1, cat2;
        private ImageView providerImage;

        public viewHolder(View view) {
            super(view);
            providerImage = view.findViewById(R.id.image);
            provider = view.findViewById(R.id.provider);
            //Email field
            cat1 = view.findViewById(R.id.imp_cat);
            //cat2 = view.findViewById(R.id.imp_cat2);
            view.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    int pos = getAdapterPosition();
                    if (listener != null && pos != RecyclerView.NO_POSITION)
{
                        listener.onItemClick(credsList.get(pos));
                    }
                }
            });
            view.setOnLongClickListener(new View.OnLongClickListener() {
                @Override
                public boolean onLongClick(View v) {
                    Log.d("OnLongClick", "Long Click");
                    return false;
                }
            });
        }
    }

    public interface onItemClickListener {
        void onItemClick(PwClass viyCred);
    }
    public void setOnItemClickListener(onItemClickListener listener) {
        this.listener = listener;
    }
}
```

SharedPrefManager.java

```
package my.edu.utar.pwmanager.Utilities;

import android.content.Context;
import android.content.SharedPreferences;

public class SharedPrefManager {

    SharedPreferences pref;
    SharedPreferences.Editor editor;
    Context _context;

    // Shared preferences file name
    private static final String SharedPref_filename = "appEssentials";

    private static final String firstTime = "IsFirstTimeLaunch";

    public SharedPrefManager(Context context) {
        this._context = context;
        pref = _context.getSharedPreferences(SharedPref_filename,
Context.MODE_PRIVATE);
        editor = pref.edit();
    }

    public void setFirstTimeLaunch(boolean isFirstTime) {
        editor.putBoolean(firstTime, isFirstTime);
        editor.apply();
    }

    public boolean isFirstTimeLaunch() {
        return pref.getBoolean(firstTime, true);
    }

}
```

About.java

```
package my.edu.utar.pwmanager;

import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.view.MenuItem;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

public class About extends AppCompatActivity {

    TextView tv;

    String version;
    //What's new feature String
    String[] whatsnew = {
            "Added Night Mode",
            "Added Secure Mode",
            "Added FingerPrint Authentication",
            "Added Random Password Generator",};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_about);
        //Set version text
        version = "Version 1.0" ;
        tv = findViewById(R.id.version);
        tv.setText(version);
        ActionBar ab = getSupportActionBar();
        if (ab != null) {
            ab.setDisplayHomeAsUpEnabled(true);
        }
        //List whatsnew array in the features listview
        ArrayAdapter adapter = new ArrayAdapter(this,
R.layout.list_whats_new, whatsnew);
        ListView listView = findViewById(R.id.listView);
        listView.setAdapter(adapter);
    }
    //IF app icon in action bar clicked, go to parent activity.
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                this.finish();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

AddEntry.java

```
package my.edu.utar.pwmanager;

import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Build;
import android.os.Bundle;
import android.text.InputType;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.security.crypto.EncryptedSharedPreferences;
import androidx.security.crypto.MasterKey;
import my.edu.utar.pwmanager.Utilities.AES_Utils;
import com.himanshurawat.hasher.HashType;
import com.himanshurawat.hasher.Hasher;
import java.io.IOException;
import java.security.GeneralSecurityException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class AddEntry extends AppCompatActivity implements
View.OnClickListener {

    final String PREFS_NAME = "appEssentials";
    SharedPreferences sharedPreferences = null;
    String PREF_KEY_SECURE_CORE_MODE = "SECURE_CORE";
    MasterKey masterKey = null;

    String[] providersEmail = {
            "Gmail", "Outlook", "Amazon", "Protonmail", "Yahoo",
            "Apple", "Paypal", "Github", "Spotify", "Stackoverflow",
            "Trello", "Wordpress", "Other"
    };
    String[] providersSocial = {
            "Facebook", "Instagram", "Twitter", "Medium", "Flickr",
            "Foursquare", "Reddit", "Slack", "Snapchat", "Tinder",
            "Linkedin", "Pinterest", "Tumblr", "Other"
    };

    String providerNameString, passwordFromCOPY;
```

```
    Button addBtn;
    Spinner providerName;
    TextView tv;
    String provider;

    private EditText email, password;

    public static final String EXTRA_PROVIDER_NAME =
"my.edu.utar.pwmanager.EXTRA_PROVIDER_NAME";
    public static final String EXTRA_PROVIDER =
"my.edu.utar.pwmanager.EXTRA_PROVIDER";
    public static final String EXTRA_ENCRYPT =
"my.edu.utar.pwmanager.EXTRA_ENCRYPT";
    public static final String EXTRA_EMAIL =
"my.edu.utar.pwmanager.EXTRA_EMAIL";
    public static final String EXTRA_IV = "my.edu.utar.pwmanager.EXTRA_IV";
    public static final String EXTRA_SALT =
"my.edu.utar.pwmanager.EXTRA_SALT";
    public static final String PASSWORD = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add);

        ActionBar ab = getSupportActionBar();
        if (ab != null) {
            ab.setDisplayHomeAsUpEnabled(true);
        }

        //Retrieve id
        providerName = findViewById(R.id.provider_name);
        email = findViewById(R.id.add_email);
        password = findViewById(R.id.add_password);

        CheckBox cb = findViewById(R.id.add_show_password);

        addBtn = findViewById(R.id.add_record);
        tv = findViewById(R.id.prov_tv);


        // Encrypted SharedPrefs
        try {
            //MK Security
            masterKey = new MasterKey.Builder(getApplicationContext(),
MasterKey.DEFAULT_MASTER_KEY_ALIAS)
                    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
                    .build();
            //initialize sharedPef
            sharedPreferences = EncryptedSharedPreferences.create(
                    getApplicationContext(),
                    PREFS_NAME,
                    masterKey,

EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,

EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
```

```
            );
        } catch (GeneralSecurityException | IOException e) {
            e.printStackTrace();
        }

        if (sharedPreferences.getBoolean(PREF_KEY_SECURE_CORE_MODE, false)) {

            getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
WindowManager.LayoutParams.FLAG_SECURE);
        }


        //IF Checkbox CHECKED, reveal password; IF Checkbox UNCHECKED, hidden
password
        cb.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked) {

password.setInputType(InputType.TYPE_NUMBER_VARIATION_PASSWORD);
                } else {
                    password.setInputType(129);
                }
            }
        });
        addBtn.setOnClickListener(this);
    }

    //Checks for Onstart, Social or Mail page.
    @Override
    protected void onStart() {
        super.onStart();
        provider = getIntent().getStringExtra(EXTRA_PROVIDER);

        if (provider == null) provider = "mail";
        passwordFromCOPY = getIntent().getStringExtra(PASSWORD);
        assert provider != null;

        switch (provider) {
            case "social":
                email.setHint("Username/Email");

                ArrayAdapter arrayAdapterSocial = new ArrayAdapter(this,
android.R.layout.simple_spinner_item, providersSocial);

arrayAdapterSocial.setDropDownViewResource(android.R.layout.simple_spinner_dr
opdown_item);

                //Setting the ArrayAdapter data on the Spinner
                providerName.setAdapter(arrayAdapterSocial);
                providerName.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
                    @Override
                    public void onItemSelected(AdapterView<?> parent, View
view, int position, long id) {
```

```
                        providerNameString =
parent.getItemAtPosition(position).toString();
                    }

                    @Override
                    public void onNothingSelected(AdapterView<?> parent) { }
                });
                break;

            default:
                email.setHint("Email");
                password.setText(passwordFromCOPY);

                ArrayAdapter arrayAdapterEmail = new ArrayAdapter(this,
android.R.layout.simple_spinner_item, providersEmail);

arrayAdapterEmail.setDropDownViewResource(android.R.layout.simple_spinner_dro
pdown_item);

                //Setting the ArrayAdapter data on the Spinner
                providerName.setAdapter(arrayAdapterEmail);
                providerName.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
                    @Override
                    public void onItemSelected(AdapterView<?> parent, View
view, int position, long id) {
                        providerNameString =
parent.getItemAtPosition(position).toString();
                    }

                    @Override
                    public void onNothingSelected(AdapterView<?> parent) { }
                });
                break;
        }

    }

    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.add_record) {
            save_data();
        }
    }

    //IF app icon in action bar clicked, go to parent activity.
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                this.finish();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
    //Retrieve input and save data
```

```
    private void save_data() {
        String text_email, text_password;
        text_email = email.getText().toString();
        text_password = password.getText().toString();
        String sha = sharedPreferences.getString("HASH", "0");
        String HASH = Hasher.Companion.hash(sha, HashType.MD5);

        //If email input is blank, Focuses and set Error("Required") on Input
Box
        if (provider.equals("mail")) {
            if (text_email.trim().isEmpty()) {
                email.setError("Required");
                email.requestFocus();
                return;
            }
            //Uses Pattern to check email formats using regex
            String emailRegex = "(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-
9!#$%&'*+/=?^_`{|}~-]+)*|\"(?:[\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\x21\\x23-
\\x5b\\x5d-\\x7f]|\\\\[\\x01-\\x09\\x0b\\x0c\\x0e-\\x7f])*\")@(?:(?:[a-z0-
9](?:[a-z0-9-]*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\\[(?:(?:25[0-
5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?|[a-z0-9-]*[a-z0-9]:(?:[\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\x21-\\x5a\\x53-
\\x7f]|\\\\[\\x01-\\x09\\x0b\\x0c\\x0e-\\x7f])+)\\])";
            Pattern pattern = Pattern.compile(emailRegex);
            Matcher matcher = pattern.matcher(text_email);

            //If email input is blank, Focuses and set Error on Input Box
            if (!matcher.matches()) {
                email.setError("Enter valid email");
                email.requestFocus();
                return;
            }
        }
        //If password input is blank, Focuses and set Error("Required") on
Input Box
        if (text_password.trim().isEmpty()) {
            password.setError("Required");
            password.requestFocus();
            return;
        }
        Intent intent = new Intent();
        intent.putExtra(EXTRA_PROVIDER_NAME, providerNameString);

        // AES encryption process, encrypt email/password
        try {
            String encEmail = AES_Utils.encrypt(text_email, HASH);
            String encPass = AES_Utils.encrypt(text_password, HASH);
            intent.putExtra(EXTRA_EMAIL, encEmail);
            intent.putExtra(EXTRA_ENCRYPT, encPass);
        } catch (Exception e) {
            e.printStackTrace();
        }
        setResult(RESULT_OK, intent);
        finish();
    }
}
```

ChangePassword.java

```
package my.edu.utar.pwmanager;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.InputFilter;
import android.text.InputType;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Toast;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.security.crypto.EncryptedSharedPreferences;
import androidx.security.crypto.MasterKey;

import java.io.IOException;
import java.security.GeneralSecurityException;

public class ChangePassword extends AppCompatActivity {

    final String PREFS_NAME = "appEssentials";
    SharedPreferences sharedPreferences = null;
    String PREF_KEY_SECURE_CORE_MODE = "SECURE_CORE";
    MasterKey masterKey;

    String TYPE_PASS_1 = "PIN";
    String TYPE_PASS_2 = "PASSWORD";
    String PREF = "HASH";
    String PREF_NAME = "Settings";

    EditText old_password_et, new_password_1_et, new_password_2_et;
    Button submit;
    String old_password, new_password_1, new_password_2;
    String TYPE_PASSWORD;

    public static final String EXTRA_TYPE_PASS =
"my.edu.utar.pwmanager.EXTRA_TYPE_PASS";


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_change_pass);

        ActionBar ab = getSupportActionBar();
        if (ab != null) {
            ab.setDisplayHomeAsUpEnabled(true);
        }
```

```
        old_password_et = findViewById(R.id.old_password);
        new_password_1_et = findViewById(R.id.change_password_1);
        new_password_2_et = findViewById(R.id.change_password_2);
        submit = findViewById(R.id.submit);

        // Encrypted SharedPrefs
        try {
            //MK.security
            masterKey = new MasterKey.Builder(getApplicationContext(),
MasterKey.DEFAULT_MASTER_KEY_ALIAS)
                    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
                    .build();

            //initialize SharedPref
            sharedPreferences = EncryptedSharedPreferences.create(
                    getApplicationContext(),
                    PREFS_NAME,
                    masterKey,

EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,

EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
            );
        } catch (GeneralSecurityException | IOException e) {
            e.printStackTrace();
        }

        if (sharedPreferences.getBoolean(PREF_KEY_SECURE_CORE_MODE, false)) {
            try {
                ImageButton copyImage = findViewById(R.id.copy);
                copyImage.setEnabled(false);
            } catch (Exception e) {
                e.getStackTrace();
            }
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
WindowManager.LayoutParams.FLAG_SECURE);
        }

        TYPE_PASSWORD = getIntent().getStringExtra(EXTRA_TYPE_PASS);

        preBuiltFormalities(TYPE_PASSWORD);

        //Onclick Listener for Button:Changing password
        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (validate()) {
                    savePassword(new_password_1);
                    Toast.makeText(getApplicationContext(), "Saved!",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }


    private boolean validate() {
```

```
        //Taking input to string
        old_password = old_password_et.getText().toString();
        new_password_1 = new_password_1_et.getText().toString();
        new_password_2 = new_password_2_et.getText().toString();

        //Fetching hash from sharedPref
        String PREF_VAL = sharedPreferences.getString(PREF, "0");
        Log.d(PREF, PREF_VAL);
        if (!PREF_VAL.equals(old_password)) {
            old_password_et.requestFocus();
            old_password_et.setError("Wrong Password");
            Log.d(PREF, "Previous: " + PREF_VAL + "Previous password: " +
old_password);
            return false;
        }
        if (!(new_password_1.equals(new_password_2))) {
            new_password_2_et.requestFocus();
            new_password_2_et.setError("Password mismatch");
            return false;
        }
        return true;
    }

    private void savePassword(String password) {
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString(PREF, password).apply();
    }

    private void preBuiltFormalities(String TYPE) {
        if (TYPE.equals(TYPE_PASS_1)) {
            new_password_1_et.setInputType(InputType.TYPE_CLASS_NUMBER |
InputType.TYPE_NUMBER_VARIATION_PASSWORD);
            new_password_2_et.setInputType(InputType.TYPE_CLASS_NUMBER |
InputType.TYPE_NUMBER_VARIATION_PASSWORD);

            new_password_1_et.setFilters(new InputFilter[]{new
InputFilter.LengthFilter(4)});
            new_password_2_et.setFilters(new InputFilter[]{new
InputFilter.LengthFilter(4)});

        } else if (TYPE.equals(TYPE_PASS_2)) {
            new_password_1_et.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_NUMBER_VARIATION_PASSWORD);
            new_password_2_et.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_NUMBER_VARIATION_PASSWORD);

            new_password_1_et.setFilters(new InputFilter[]{new
InputFilter.LengthFilter(16)});
            new_password_2_et.setFilters(new InputFilter[]{new
InputFilter.LengthFilter(16)});
        }
    }

    //IF app icon in action bar clicked, go to parent activity.
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
```

```
        switch (item.getItemId()) {
            case android.R.id.home:
                this.finish();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

Help.java

```
package my.edu.utar.pwmanager;

import android.content.Intent;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

public class Help extends AppCompatActivity {
    String version;
    TextView tv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_help);

        version = "Version 1.0";

        tv = findViewById(R.id.version);
        tv.setText(version);

        ActionBar ab = getSupportActionBar();
        if (ab != null) {
            ab.setDisplayHomeAsUpEnabled(true);
        }
    }

    //Create a link to next activity
    public void whats_new(View view) {
        startActivity(new Intent(this, About.class));
    }


    //IF app icon in action bar clicked, go to parent activity.
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                this.finish();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

Home.java

```
package my.edu.utar.pwmanager;

import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.lifecycle.ViewModelProvider;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;
import androidx.security.crypto.EncryptedSharedPreferences;
import androidx.security.crypto.MasterKey;

import my.edu.utar.pwmanager.classFramework.PwClass;
import my.edu.utar.pwmanager.Fragments.mail.MailViewModel;
import com.github.javiersantos.appupdater.AppUpdater;
import com.github.javiersantos.appupdater.enums.Display;
import com.google.android.material.navigation.NavigationView;

import java.io.IOException;
import java.security.GeneralSecurityException;
import java.util.Random;

public class Home extends AppCompatActivity {


    final String PREFS_NAME = "appEssentials";
    SharedPreferences sharedPreferences;
    String PREF_KEY_SECURE_CORE_MODE = "SECURE_CORE";
    MasterKey masterKey = null;

    public static final String NO_DATA = "NO DATA";
    private static final int ADD_RECORD = 1;
    private static final String TAG = "HOME";
    private static final String PROVIDER = "mail";
```

```
    String PASSWORD = "";

    private static final String collection =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*_=+-";

    AlertDialog.Builder builder;
    boolean secureCodeModeState;
    private AppBarConfiguration mAppBarConfiguration;
    AppUpdater appUpdater;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        getIntent().setAction("1");
        setContentView(R.layout.activity_home);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        //Retrieve ID
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        View view = navigationView.getHeaderView(0);
        ImageButton imageButton = view.findViewById(R.id.refresh);
        final TextView textView1 = view.findViewById(R.id.generate_password);
        builder = new AlertDialog.Builder(this);

        // Encrypted SharedPrefs
        try {
            //MK.security
            masterKey = new MasterKey.Builder(getApplicationContext(),
MasterKey.DEFAULT_MASTER_KEY_ALIAS)
                    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
                    .build();

            //initialize sharedPref
            sharedPreferences = EncryptedSharedPreferences.create(
                    getApplicationContext(),
                    PREFS_NAME,
                    masterKey,

EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,

EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
            );
        } catch (GeneralSecurityException | IOException e) {
            e.printStackTrace();
        }

        //Checks for SECURE CORE mode
        secureCodeModeState =
sharedPreferences.getBoolean(PREF_KEY_SECURE_CORE_MODE, false);

        if (secureCodeModeState) {
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
```

```
WindowManager.LayoutParams.FLAG_SECURE);
        }

        //Passing mail/social ID
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_password,
                R.id.nav_social)
                .setDrawerLayout(drawer)
                .build();

        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);

        //Initial random password generator
        String Password = generatePassword();
        textView1.setText(Password);

        //Listen Activity for password refresh button
        imageButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                nav_refresh();
            }
        });

        //Display notification
        appUpdater = new AppUpdater(this)
                .showEvery(3)
                .setDisplay(Display.NOTIFICATION)
                .setDisplay(Display.DIALOG);
    }


    @Override
    protected void onStart() {
        super.onStart();
        if (sharedPreferences.getBoolean("FIRSTNOTICE", true)) {

            //Build ALert Dialog
            AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(this);
            alertDialogBuilder.setTitle("Notice");
            alertDialogBuilder.setMessage("SECURE+ is still in beta, you
might face some bugs while using the app.");

            //OK Button
            alertDialogBuilder.setPositiveButton("OK", new
DialogInterface.OnClickListener() {

                @Override
                public void onClick(DialogInterface arg0, int arg1) {
                    SharedPreferences.Editor editor =
sharedPreferences.edit();
                    editor.putBoolean("FIRSTNOTICE", false).apply();
```

```
                }
            });
            //Negative button
            AlertDialog alertDialog = alertDialogBuilder.create();
            alertDialog.show();

            if (!secureCodeModeState)
                Log.d("Update", String.valueOf(secureCodeModeState));
            appUpdater.start();
        }
    }


    // Inflates the menu; this adds items to the action bar if it is present.
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.home, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }


    //Intent linking to other activity
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_settings:
                startActivity(new Intent(getApplicationContext(),
Settings.class));
                return true;
            case R.id.action_help:
                startActivity(new Intent(getApplicationContext(),
Help.class));
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    //Functions: copy random password
    public void copy(View view) {

        if (sharedPreferences.getBoolean(PREF_KEY_SECURE_CORE_MODE, false)) {

            ImageButton copyImage = findViewById(R.id.copy);

            //Checks for SECURE CORE MODE, if enabled, not allowed to copy
            copyImage.setEnabled(false);
            Toast.makeText(this, "Secure code mode is Enabled. Copying is not
```

```
allowed  ", Toast.LENGTH_SHORT).show();

        } else {
            TextView textView = findViewById(R.id.generate_password);
            final String gn_password = textView.getText().toString().trim();

            //Clipboard manager to copy strings to clipboard
            ClipboardManager clipboard = (ClipboardManager)
                    getSystemService(Context.CLIPBOARD_SERVICE);
            ClipData clip = ClipData.newPlainText("Password", gn_password);

            if (clipboard != null) {
                //Copies strings to clipboard
                clipboard.setPrimaryClip(clip);
                Toast.makeText(getApplicationContext(), "Copied!",
Toast.LENGTH_SHORT).show();
            }

            builder.setMessage("Do you want to add this password to the
database?")
                    .setTitle("Alert")
                    .setCancelable(true)
                    //Set listener, if yes, copy to clipboard
                    .setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int
which) {
                            //If yes, Calls to Add Activity
                            Intent intent = new
Intent(getApplicationContext(), AddEntry.class);

                            //cast the copied strings into PASSWORD field
                            intent.putExtra(PASSWORD, gn_password);
                            startActivityForResult(intent, ADD_RECORD);
                        }
                    })
                    .setNegativeButton("No", new
DialogInterface.OnClickListener() {
                        //"No" button to decline
                        @Override
                        public void onClick(DialogInterface dialog, int
which) {
                            dialog.cancel();
                        }
                    });
        }
        AlertDialog alert = builder.create();
        alert.show();
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        if (requestCode == ADD_RECORD && resultCode == RESULT_OK) {

            //Attributes
```

```
            String providerName =
data.getStringExtra(AddEntry.EXTRA_PROVIDER_NAME);
            String enc_passwd = data.getStringExtra(AddEntry.EXTRA_ENCRYPT);
            String enc_email = data.getStringExtra(AddEntry.EXTRA_EMAIL);

            //Create new PwClass that store credential
            PwClass pwClass = new PwClass(PROVIDER, providerName, enc_email,
enc_passwd);
            Log.d(TAG, "Provider: " + PROVIDER + " EMAIL: " + enc_email + "
ENC_DATA: " + enc_passwd);

            SharedPreferences sharedPreferences =
this.getApplicationContext().getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);

            MailViewModel passwordViewModel = new
ViewModelProvider(this).get(MailViewModel.class);

            //insert data into viewmodel
            passwordViewModel.insert(pwClass);
            Toast.makeText(getApplicationContext(), "Saved",
Toast.LENGTH_SHORT).show();
        }
        super.onActivityResult(requestCode, resultCode, data);
    }

    private String generatePassword() {

        //Creating Random object
        Random random = new Random();

        //Limits  length of the generated password
        int limit = (int) (Math.random() * 10 + 5);

        //Build String using string builder
        StringBuilder password = new StringBuilder();
        for (int itr = 0; itr < limit; itr++) {

password.append(collection.charAt(random.nextInt(collection.length())));
        }
        return password.toString();
    }

    public void nav_refresh() {
        // refresh/generate new password on textview
        TextView textView = findViewById(R.id.generate_password);
        String Password = generatePassword();
        textView.setText(Password);
    }


}
```

MasterLock.java

```
package my.edu.utar.pwmanager;

import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.biometric.BiometricPrompt;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.FragmentActivity;
import androidx.security.crypto.EncryptedSharedPreferences;
import androidx.security.crypto.MasterKey;

import com.andrognito.pinlockview.IndicatorDots;
import com.andrognito.pinlockview.PinLockListener;
import com.andrognito.pinlockview.PinLockView;

import java.io.IOException;
import java.security.GeneralSecurityException;
import java.util.concurrent.Executor;
import java.util.concurrent.Executors;
//MLock Master Lock

public class MasterLock extends AppCompatActivity {

    final String PREFS_NAME = "appEssentials";
    SharedPreferences sharedPreferences = null;
    MasterKey masterKey = null;

    final String PREF_KEY = "firstRun";
    final String HASH = "HASH";
    final int DOESNT_EXIST = -1;

    TextView mlock_tv_greet, mlock_tv_pp;
    PinLockView mPinLockView;
    IndicatorDots mIndicatorDots;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setStatusBarGradiant(this);
        setContentView(R.layout.activity_mlock);

        //Calls function for biometrics authentication
```

```
        biometricsAuth();

        //retrieve ID
        mlock_tv_greet = findViewById(R.id.mlock_l_tv_greet);
        mIndicatorDots = findViewById(R.id.indicator_dots);
        mPinLockView = findViewById(R.id.pin_lock_view);

        // Encrypted SharedPrefs
        try {
            //MK.security
            masterKey = new MasterKey.Builder(getApplicationContext(),
MasterKey.DEFAULT_MASTER_KEY_ALIAS)
                    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
                    .build();

            //initialize sharedpPef
            sharedPreferences = EncryptedSharedPreferences.create(
                    getApplicationContext(),
                    PREFS_NAME,
                    masterKey,

EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,

EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
            );
        } catch (GeneralSecurityException | IOException e) {
            e.printStackTrace();
        }

        //Checks whether is firstTime, if yes display "Create master
password"
        if (sharedPreferences.getBoolean(PREF_KEY, true)) {

            mlock_tv_greet.setText(R.string.mlock_st_create_password);

            //Setting biometric auth button to invis, forcing user to set up
master password
            findViewById(R.id.launchAuthentication).setVisibility(View.GONE);
        }

        //Create PinLock Listener
        PinLockListener mPinLockListener = new PinLockListener() {
            @Override
            public void onComplete(String pin) {

                //First time logging in, create new HASH, and stores pin
                if (sharedPreferences.getBoolean(PREF_KEY, true)) {

                    sharedPreferences.edit().putString(HASH, pin).apply();
//                    String HASH = new
String(Hex.encodeHex(DigestUtils.sha(pin)));
                    sharedPreferences.edit().putBoolean(PREF_KEY,
false).apply();

                    Toast.makeText(getApplicationContext(), "Welcome",
Toast.LENGTH_SHORT).show();
                    startActivity(new Intent(getApplicationContext(),
```

```
Home.class));
                    finish();
                } else {//compares with HASH, if pin matches, proceeds to
Home
                    String sp = sharedPreferences.getString(HASH, "0");
                    if (sp.equals(pin)) {
//                        Toast.makeText(getApplicationContext(), "Successful
login", Toast.LENGTH_SHORT).show();
                        startActivity(new Intent(getApplicationContext(),
Home.class));
                        finish();
                    } else {//if pin does not matches, display "Wrong
Password"
                        Toast.makeText(getApplicationContext(), "Wrong
password", Toast.LENGTH_SHORT).show();
                    }
                }

            }

            //Checks PIN condition
            @Override
            public void onEmpty() {
//                Log.d(TAG, "Pin empty");
            }

            @Override
            public void onPinChange(int pinLength, String intermediatePin) {
                //Log.d(TAG, "Pin changed, new length " + pinLength + " with
intermediate pin " + intermediatePin);
            }
        };

        //Create listener for pinlock using indicator dots
        mPinLockView.setPinLockListener(mPinLockListener);
        mPinLockView.attachIndicatorDots(mIndicatorDots);
    }

    //Functions: biometric authentication
    public void biometricsAuth() {

        //Create single thread
        Executor newExecutor = Executors.newSingleThreadExecutor();
        FragmentActivity activity = this;

        //Start listening for authentication events
        final BiometricPrompt myBiometricPrompt = new
BiometricPrompt(activity, newExecutor, new
BiometricPrompt.AuthenticationCallback() {
            @Override
            //onAuthenticationError is called when a fatal error occur
            public void onAuthenticationError(int errorCode, @NonNull
CharSequence errString) {
                super.onAuthenticationError(errorCode, errString);
                if (errorCode == BiometricPrompt.ERROR_NEGATIVE_BUTTON) {
                } else {
                    //Print a message to Logcat//
```

```
//                    Log.d(TAG, "An unrecoverable error occurred");
                }
            }

            //onAuthenticationSucceeded is called when a fingerprint is
matched successfully
            @Override
            public void onAuthenticationSucceeded(@NonNull
BiometricPrompt.AuthenticationResult result) {
                super.onAuthenticationSucceeded(result);
                //Print a message to Logcat//
                startActivity(new Intent(MasterLock.this, Home.class));
                finish();
//                Log.d(TAG, "Fingerprint recognised successfully");
            }

            //onAuthenticationFailed is called when the fingerprint do not
match
            @Override
            public void onAuthenticationFailed() {
                super.onAuthenticationFailed();
                //Print a message to Logcat//
//                Log.d(TAG, "Fingerprint not recognised");
            }
        });

        //Create the BiometricPrompt instance//
        final BiometricPrompt.PromptInfo promptInfo = new
BiometricPrompt.PromptInfo.Builder()
                //Add some text to the dialog//
                .setTitle("Fingerprint Authentication")
                .setDescription("Place your finger on the sensor to
authenticate")
                .setNegativeButtonText("Cancel")
                //Build the dialog//
                .build();
        findViewById(R.id.launchAuthentication).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                myBiometricPrompt.authenticate(promptInfo);
            }
        });
    }

    // Setting Gradient on statusbar
    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public static void setStatusBarGradiant(Activity activity) {
        Window window = activity.getWindow();

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS)
;
        window.setStatusBarColor(ContextCompat.getColor(activity,
R.color.bg_color_primary));
    }
}
```

ModifyEntry.java

```
package my.edu.utar.pwmanager;

import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.InputType;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.security.crypto.EncryptedSharedPreferences;
import androidx.security.crypto.MasterKey;

import my.edu.utar.pwmanager.Utilities.AES_Utils;
import com.himanshurawat.hasher.HashType;
import com.himanshurawat.hasher.Hasher;

import java.io.IOException;
import java.security.GeneralSecurityException;

public class ModifyEntry extends AppCompatActivity implements
View.OnClickListener {

    String PREF_NAME = "appEssentials";
    private static final String PREFS_NAME = "appEssentials";
    SharedPreferences sharedPreferences = null;
    String PREF_KEY_SECURE_CORE_MODE = "SECURE_CORE";
    MasterKey masterKey = null;

    EditText newPassword;
    TextView emailText, oldPassword;
    String provName, email, passwd, decPass;
    CheckBox show_change_password, show_password;
    Button changePasswordButton, updateBtn, deleteBtn;
    LinearLayout newPasswordLayout;

    public static final String TAG = "MODIFY";
    public static final String EXTRA_DELETE = "DELETE";
    public static final String EXTRA_PROVIDER_NAME =
"my.edu.utar.pwmanager.EXTRA_PROVIDER_NAME";
    public static final String EXTRA_ID = "my.edu.utar.pwmanager.EXTRA_ID";
    public static final String EXTRA_ENCRYPT =
```

```
"my.edu.utar.pwmanager.EXTRA_ENCRYPT";
    public static final String EXTRA_EMAIL =
"my.edu.utar.pwmanager.EXTRA_EMAIL";
    public static final String EXTRA_IV = "my.edu.utar.pwmanager.EXTRA_IV";
    public static final String EXTRA_SALT =
"my.edu.utar.pwmanager.EXTRA_SALT";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_modify);

        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            actionBar.setDisplayHomeAsUpEnabled(true);
        }

        //Retrieve ID
        emailText = findViewById(R.id.modify_email);
        oldPassword = findViewById(R.id.modify_old_password);
        show_password = findViewById(R.id.show_password);
        changePasswordButton = findViewById(R.id.change_password_button);
        newPassword = findViewById(R.id.modify_new_password);
        show_change_password = findViewById(R.id.modify_show_password);
        updateBtn = findViewById(R.id.modify_update);
        deleteBtn = findViewById(R.id.modify_delete);

        updateBtn.setEnabled(false);

        // Encrypted SharedPrefs
        try {
            //MK.security
            masterKey = new MasterKey.Builder(getApplicationContext(),
MasterKey.DEFAULT_MASTER_KEY_ALIAS)
                    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
                    .build();

            //initialize sharedPef
            sharedPreferences = EncryptedSharedPreferences.create(
                    getApplicationContext(),
                    PREFS_NAME,
                    masterKey,

EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,

EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
            );
        } catch (GeneralSecurityException | IOException e) {
            e.printStackTrace();
        }

        if (sharedPreferences.getBoolean(PREF_KEY_SECURE_CORE_MODE, false)) {
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
WindowManager.LayoutParams.FLAG_SECURE);
        }

        //Retrieve HASH
```

```
        String sha = sharedPreferences.getString("HASH", "0");
        String HASH = Hasher.Companion.hash(sha, HashType.MD5);

        //Decryption Process
        Intent intent = getIntent();
        provName = intent.getStringExtra(EXTRA_PROVIDER_NAME);
        email = intent.getStringExtra(EXTRA_EMAIL);
        passwd = intent.getStringExtra(EXTRA_ENCRYPT);
        try {
            String decEmail = AES_Utils.decrypt(email, HASH);
            decPass = AES_Utils.decrypt(passwd, HASH);

            emailText.setText(decEmail);
            oldPassword.setText(decPass);
        } catch (Exception e) {
            e.printStackTrace();
        }
        //Check Box to reveal or hide new password
        show_change_password.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked) {

newPassword.setInputType(InputType.TYPE_NUMBER_VARIATION_PASSWORD);
                } else {
                    newPassword.setInputType(129);
                }
            }
        });

        //Check Box to reveal or hide old password
        show_password.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked) {

oldPassword.setInputType(InputType.TYPE_NUMBER_VARIATION_PASSWORD);
                } else {
                    oldPassword.setInputType(129);
                }
            }
        });
        //Apply listener to buttons
        updateBtn.setOnClickListener(this);
        deleteBtn.setOnClickListener(this);
        changePasswordButton.setOnClickListener(this);
    }

    private void changePassword() {
        updateBtn.setEnabled(true);
        findViewById(R.id.show_password).setVisibility(View.GONE);
        changePasswordButton.setVisibility(View.GONE);
        newPasswordLayout = findViewById(R.id.change_password);
```

```
        newPasswordLayout.setVisibility(View.VISIBLE);
    }


    private void delete_data() {

        Intent intent = new Intent();
        intent.putExtra(EXTRA_DELETE, true);
        intent.putExtra(EXTRA_EMAIL, email);
        intent.putExtra(EXTRA_ENCRYPT, passwd);
        int id = getIntent().getIntExtra(EXTRA_ID, -1);

        //Checks for condition, removes data using intent
        if (id != -1) {
            intent.putExtra(EXTRA_ID, id);
        }
        setResult(RESULT_OK, intent);
        finish();
    }

    private void modify_data() {
        String text_old_password, text_new_password;
        text_old_password = oldPassword.getText().toString();
        text_new_password = newPassword.getText().toString();

        //Retrieve HASH
        String sha = sharedPreferences.getString("HASH", "0");
        String HASH = Hasher.Companion.hash(sha, HashType.MD5);

        //Prompt user to enter valid password
        if (text_old_password.trim().isEmpty()) {
            oldPassword.setError("Required");
            oldPassword.requestFocus();
            return;
        }
        if (text_new_password.trim().isEmpty()) {
            newPassword.setError("Required");
            newPassword.requestFocus();
            return;
        }

        //Encrypt new password
        Intent intent = new Intent();
        intent.putExtra(EXTRA_PROVIDER_NAME, provName);
        intent.putExtra(EXTRA_EMAIL, email);
        try {
            String encPass = AES_Utils.encrypt(text_new_password, HASH);
            intent.putExtra(EXTRA_ENCRYPT, encPass);
        } catch (Exception e) {
            e.printStackTrace();
        }
        int id = getIntent().getIntExtra(EXTRA_ID, -1);
        if (id != -1) {
            intent.putExtra(EXTRA_ID, id);
        }
        setResult(RESULT_OK, intent);
        finish();
```

```
    }

    //Set Listener for buttons
    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.modify_update && updateBtn.isEnabled()) {
            modify_data();
        } else if (v.getId() == R.id.modify_delete) {
            delete_data();
        } else if (v.getId() == R.id.change_password_button) {
            changePassword();
        }
    }

    //IF app icon in action bar clicked, go to parent activity.
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                this.finish();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    public void copy_email(View view) {
        if (sharedPreferences.getBoolean(PREF_KEY_SECURE_CORE_MODE, false)) {
            //checks for SECURE MODE condition, disable copying
            Toast.makeText(this, "Secure code mode is Enabled. Copying is not
allowed  ", Toast.LENGTH_SHORT).show();
        } else {
            //Parse string into gn_email and copy to clipboard
            TextView textView = findViewById(R.id.modify_email);
            final String gn_email = textView.getText().toString().trim();

            ClipboardManager clipboard = (ClipboardManager)
                    getSystemService(Context.CLIPBOARD_SERVICE);
            ClipData clip = ClipData.newPlainText("Email", gn_email);

            if (clipboard != null) {
                clipboard.setPrimaryClip(clip);
                Toast.makeText(getApplicationContext(), "Email Copied!",
Toast.LENGTH_SHORT).show();
            }
        }
    }

    public void copy_password(View view) {
        if (sharedPreferences.getBoolean(PREF_KEY_SECURE_CORE_MODE, false)) {
            //checks for SECURE MODE condition, disable copying
            Toast.makeText(this, "Secure code mode is Enabled. Copying is not
allowed  ", Toast.LENGTH_SHORT).show();
        } else {
            //Parse string into gn_email and copy to clipboard
            TextView textView = findViewById(R.id.modify_old_password);
            final String gn_password = textView.getText().toString().trim();
```

```
            ClipboardManager clipboard = (ClipboardManager)
                    getSystemService(Context.CLIPBOARD_SERVICE);
            ClipData clip = ClipData.newPlainText("Password", gn_password);

            if (clipboard != null) {
                clipboard.setPrimaryClip(clip);
                Toast.makeText(getApplicationContext(), "Password Copied!",
Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

Settings.java

```
package my.edu.utar.pwmanager;

import android.Manifest;
import android.app.Activity;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.widget.CompoundButton;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatDelegate;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.security.crypto.EncryptedSharedPreferences;
import androidx.security.crypto.MasterKey;
import my.edu.utar.pwmanager.database.PwDB;
import my.edu.utar.pwmanager.Fragments.mail.MailViewModel;
import com.google.android.material.switchmaterial.SwitchMaterial;
import java.io.IOException;
import java.security.GeneralSecurityException;
import ir.androidexception.roomdatabasebackupandrestore.Backup;
import ir.androidexception.roomdatabasebackupandrestore.OnWorkFinishListener;
import ir.androidexception.roomdatabasebackupandrestore.Restore;

public class Settings extends AppCompatActivity {

    final String PREFS_NAME = "appEssentials";
    SharedPreferences sharedPreferences = null;
    MasterKey masterKey = null;

    SharedPreferences UIPref;
    String PREF_DARK = "DARK_THEME";

    String PREF_KEY_SECURE_CORE_MODE = "SECURE_CORE";
    boolean secureCodeModeState;
    String PREF_KEY_SCM_COPY = "SCM_COPY";
    String PREF_KEY_SCM_SCREENSHOTS = "SCM_SCREENSHOTS";
    String NO_DATA = "NO DATA";
    String TYPE_PASS_1 = "PIN";
    String TYPE_PASS_2 = "PASSWORD";

    String PREF_KEY = "MASTER_PASSWORD";
    String PACKAGE_NAME;
```

```
    TextView change_password, delete_data, about_app;
    ProgressBar progressBar;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.settings_activity);

        //retrieve package name
        PACKAGE_NAME = getApplicationContext().getPackageName();

        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            actionBar.setDisplayHomeAsUpEnabled(true);
        }

        // Encrypted SharedPrefs
        try {
            //MK.security
            masterKey = new MasterKey.Builder(getApplicationContext(),
MasterKey.DEFAULT_MASTER_KEY_ALIAS)
                    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
                    .build();

            //initialize sharedpPef
            sharedPreferences = EncryptedSharedPreferences.create(
                    getApplicationContext(),
                    PREFS_NAME,
                    masterKey,

EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,

EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
            );
        } catch (GeneralSecurityException | IOException e) {
            e.printStackTrace();
        }

        //Retrieve ID
        change_password = findViewById(R.id.change_master_password);
        delete_data = findViewById(R.id.delete_all_data);
        about_app = findViewById(R.id.about_app);
        progressBar = findViewById(R.id.progress_bar);
        final SwitchMaterial secureCoreModeSwitch =
findViewById(R.id.secure_core_mode);
        final SwitchMaterial dark_theme = findViewById(R.id.ask_dark_theme);

        //Retrieve conditions
        secureCodeModeState =
sharedPreferences.getBoolean(PREF_KEY_SECURE_CORE_MODE, false);
        final boolean askPasswordLaunchState =
sharedPreferences.getBoolean(PREF_KEY, true);
        secureCoreModeSwitch.setChecked(secureCodeModeState);

        //Set theme mode
        UIPref = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
```

```
        boolean onDarkTheme = UIPref.getBoolean(PREF_DARK, false);
        if (onDarkTheme) {
            dark_theme.setChecked(onDarkTheme);
        }

        final SharedPreferences.Editor editor = sharedPreferences.edit();


        final SharedPreferences.Editor UIEditor = UIPref.edit();

        //Button to set dark/light theme
        dark_theme.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked) {
                    // Enable Dark theme
                    AppCompatDelegate.setDefaultNightMode(
                            AppCompatDelegate.MODE_NIGHT_YES
                    );
                    UIEditor.putBoolean(PREF_DARK, true).apply();
                } else {
                    // Disable Dark theme
                    AppCompatDelegate.setDefaultNightMode(
                            AppCompatDelegate.MODE_NIGHT_NO
                    );
                    UIEditor.putBoolean(PREF_DARK, false).apply();
                }
            }
        });

        //Creates listener for Secure Mode button
        secureCoreModeSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked) {
                    // Removing
                    secureCodeMode(true);
                    editor.putBoolean(PREF_KEY_SECURE_CORE_MODE,
true).apply();
                } else {
                    secureCodeMode(false);
                    editor.putBoolean(PREF_KEY_SECURE_CORE_MODE,
false).apply();
                }
            }
        });
    }

    //SecureCodeMode functions
    private void secureCodeMode(boolean state) {
        final SharedPreferences.Editor editor = sharedPreferences.edit();
        if (state) {
```

```
            //remove copy to clipboard and screenshot ability
            editor.putBoolean(PREF_KEY_SCM_COPY, false).apply();
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
WindowManager.LayoutParams.FLAG_SECURE);
            Toast.makeText(getApplicationContext(), "Success. Restart app to
apply changes", Toast.LENGTH_LONG).show();
        } else {

            //set copy to clipboard and screenshot ability
            editor.putBoolean(PREF_KEY_SCM_SCREENSHOTS, true).apply();
            getWindow().clearFlags(WindowManager.LayoutParams.FLAG_SECURE);
            Toast.makeText(getApplicationContext(), "Secure code mode is
inactive", Toast.LENGTH_LONG).show();
        }
    }


    //Change password/PIN
    public void changePassword(View view) {
        TextView PIN = findViewById(R.id.change_master_password_option_1);
        PIN.setVisibility(View.VISIBLE);

    }

    public void changePasswordToPIN(View view) {
        Intent intent = new Intent(getApplicationContext(),
ChangePassword.class);
        intent.putExtra(ChangePassword.EXTRA_TYPE_PASS, TYPE_PASS_1);
        startActivity(intent);
    }


    public void deleteAllData(View view) {

        //Build Alert Dialog
        AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(this);
        alertDialogBuilder.setTitle("Delete All Data");
        alertDialogBuilder.setMessage("Do you want to delete everything?");
        alertDialogBuilder.setCancelable(false);

        //YES buton
        alertDialogBuilder.setPositiveButton("yes", new
DialogInterface.OnClickListener() {

            @Override
            public void onClick(DialogInterface arg0, int arg1) {
                MailViewModel passwordViewModel = new
MailViewModel(getApplication());
                progressBar.setVisibility(View.VISIBLE);
                //deletes all data
                passwordViewModel.deleteAllNotes();
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putBoolean(NO_DATA, false).apply();
                progressBar.setVisibility(View.GONE);
                Toast.makeText(getApplicationContext(), "Deleted",
Toast.LENGTH_SHORT).show();
```

```
            }
        });
        //NO button
        alertDialogBuilder.setNegativeButton("No", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                finish();
            }
        });
        AlertDialog alertDialog = alertDialogBuilder.create();
        alertDialog.show();

    }

    //Intent linking current activity to next activity
    public void aboutApp(View view) {
        startActivity(new Intent(this, About.class));
    }

    //IF app icon in action bar clicked, go to parent activity.
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                this.finish();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

SplashArt.java

```
package my.edu.utar.pwmanager;

import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.view.Window;
import android.view.WindowManager;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatDelegate;
import androidx.core.content.ContextCompat;
import androidx.security.crypto.EncryptedSharedPreferences;
import androidx.security.crypto.MasterKey;

import java.io.IOException;
import java.security.GeneralSecurityException;

public class SplashArt extends AppCompatActivity {

    final String PREFS_NAME = "appEssentials";
    SharedPreferences sharedPreferences = null;
    MasterKey masterKey = null;

    String PREF_KEY = "MASTER_PASSWORD";
    String PREF_DARK = "DARK_THEME";
    String PREF_KEY_FRUN = "FIRST RUN";

    //timeout timer for splash art
    private static int timeout = 2000;


    // Sets gradient on statusbar
    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public static void setStatusBarGradiant(Activity activity) {
        Window window = activity.getWindow();

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS)
;
        window.setStatusBarColor(ContextCompat.getColor(activity,
R.color.bg_color_splash));
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //Checks whether Light/Dark MODE
        SharedPreferences UIPref = getSharedPreferences(PREFS_NAME,
```

```
MODE_PRIVATE);
        boolean UI = UIPref.getBoolean(PREF_DARK, false);

        if (UIPref.getBoolean(PREF_DARK, false)) {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
        } else {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
        }
        setStatusBarGradiant(this);

        setContentView(R.layout.activity_splash);

        //Retrieve ID
        TextView password_manager = findViewById(R.id.password_manager);
        password_manager.setText("Secure+");

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                // Encrypted SharedPrefs
                try {
                    //MK.security
                    masterKey = new
MasterKey.Builder(getApplicationContext(),
MasterKey.DEFAULT_MASTER_KEY_ALIAS)
                            .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
                            .build();

                    //initialize sharedPef
                    sharedPreferences = EncryptedSharedPreferences.create(
                            getApplicationContext(),
                            PREFS_NAME,
                            masterKey,

EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,

EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
                    );
                } catch (GeneralSecurityException | IOException e) {
                    e.printStackTrace();
                }
                //Checks for firstTime and PasswordOnLaunch conditions
                final boolean askPasswordLaunchState =
sharedPreferences.getBoolean(PREF_KEY, true);
                final boolean firstRun =
sharedPreferences.getBoolean(PREF_KEY_FRUN, true);
                if (firstRun) {
                    //if first time using the app
                    //play welcome slides
                    startActivity(new Intent(SplashArt.this,
WelcomeScreen.class));
                } else {
                    //!FirstTime
                    if (askPasswordLaunchState) {
                        //If Password required, proceeds to MLock
```

```
                        startActivity(new Intent(SplashArt.this,
MasterLock.class));
                    } else {
                        //If Password not required, proceeds to home
                        startActivity(new Intent(SplashArt.this,
Home.class));
                        Toast.makeText(getApplicationContext(), "Consider
using password", Toast.LENGTH_SHORT).show();
                    }
                }
                finish();
            }
            //Timeout timer set for 2seconds
        }, timeout);
    }
}
```

WelcomeScreen.java

```
package my.edu.utar.pwmanager;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.text.Html;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.viewpager.widget.PagerAdapter;
import androidx.viewpager.widget.ViewPager;

import my.edu.utar.pwmanager.Utilities.SharedPrefManager;

public class WelcomeScreen extends AppCompatActivity {

    private SharedPrefManager prefManager;
    private int[] layouts;
    private TextView[] dots;
    private ViewPager viewPager;
    private myPagerAdapter myViewPagerAdapter;
    private LinearLayout dotsLayout;
    private Button btnSkip, btnNext;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //Checking for first time launch - before calling setContentView()
        prefManager = new SharedPrefManager(this);
        if (!prefManager.isFirstTimeLaunch()) {
            launchHomeScreen();
            finish();
        }

        //Making notification bar transparent
        if (Build.VERSION.SDK_INT >= 21) {

getWindow().getDecorView().setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_S
TABLE | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
        }

        setContentView(R.layout.activity_welcome);
```

```
        //Retrieve ID
        viewPager = findViewById(R.id.view_pager);
        dotsLayout = findViewById(R.id.layoutDots);
        btnSkip = findViewById(R.id.btn_skip);
        btnNext = findViewById(R.id.btn_next);


        //Layouts of welcome slides, in order
        layouts = new int[]{
                R.layout.welcome_slide1,
                R.layout.welcome_slide2,
                R.layout.welcome_slide3};

        //Adding bottom dots
        addBottomDots(0);

        //Making notification bar transparent
        changeStatusBarColor();

        //Creates myViewPagerAdapter
        myViewPagerAdapter = new myPagerAdapter();
        viewPager.setAdapter(myViewPagerAdapter);
        viewPager.addOnPageChangeListener(viewPagerPageChangeListener);
        btnSkip.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                launchHomeScreen();
            }
        });

        //Create Listener
        btnNext.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //checks for last page, if last page home screen will be
launched
                int current = getItem(+1);
                if (current < layouts.length) {
                    //move to next screen
                    viewPager.setCurrentItem(current);
                } else {
                    launchHomeScreen();
                }
            }
        });
    }

    //Functions for create bottom dots page indicator
    private void addBottomDots(int currentPage) {
        dots = new TextView[layouts.length];

        int[] colorsActive =
getResources().getIntArray(R.array.array_dot_active);
        int[] colorsInactive =
getResources().getIntArray(R.array.array_dot_inactive);

        dotsLayout.removeAllViews();
```

```
        for (int i = 0; i < dots.length; i++) {
            dots[i] = new TextView(this);
            dots[i].setText(Html.fromHtml("&#8226;"));
            dots[i].setTextSize(35);
            dots[i].setTextColor(colorsInactive[currentPage]);
            dotsLayout.addView(dots[i]);
        }

        if (dots.length > 0)
            dots[currentPage].setTextColor(colorsActive[currentPage]);
    }

    private int getItem(int i) {
        return viewPager.getCurrentItem() + i;
    }

    //Sets FirstTime=False, proceeds to MasterLock
    private void launchHomeScreen() {
        prefManager.setFirstTimeLaunch(false);
        startActivity(new Intent(WelcomeScreen.this, MasterLock.class));
        finish();
    }

    //Viewpager change listener
    ViewPager.OnPageChangeListener viewPagerPageChangeListener = new
ViewPager.OnPageChangeListener() {

        @Override
        public void onPageSelected(int position) {
            addBottomDots(position);

            // changing the next button text 'NEXT' / 'GOT IT'
            if (position == layouts.length - 1) {
                // last page. make button text to GOT IT
                btnNext.setText(getString(R.string.start));
                btnSkip.setVisibility(View.GONE);
            } else {
                // still pages are left
                btnNext.setText(getString(R.string.next));
                btnSkip.setVisibility(View.VISIBLE);
            }
        }

        @Override
        public void onPageScrolled(int arg0, float arg1, int arg2) { }

        @Override
        public void onPageScrollStateChanged(int arg0) { }
    };

    //Makes notification bar transparent
    private void changeStatusBarColor() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            Window window = getWindow();

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS)
;
```

```
            window.setStatusBarColor(Color.TRANSPARENT);
        }
    }

    //Create myPagerAdapter and Override functions
    public class myPagerAdapter extends PagerAdapter {
        private LayoutInflater layoutInflater;

        public myPagerAdapter() {
        }

        @Override
        public Object instantiateItem(ViewGroup container, int position) {
            layoutInflater = (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);

            View view = layoutInflater.inflate(layouts[position], container,
false);
            container.addView(view);

            return view;
        }

        @Override
        public int getCount() {
            return layouts.length;
        }

        @Override
        public boolean isViewFromObject(View view, Object obj) {
            return view == obj;
        }


        @Override
        public void destroyItem(ViewGroup container, int position, Object
object) {
            View view = (View) object;
            container.removeView(view);
        }
    }
}
```

Build.gradle (Project:PWmanager)

```
// Top-level build file where you can add configuration options common to all
sub-projects/modules.

buildscript {
    ext.kotlin_version = '1.5.21'

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:7.1.1'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"


        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
        maven{
            url "https://maven.google.com"
        }
        maven { url 'https://jitpack.io' }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Build.gradle (Module:PWmanager.app)

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'

android {
    compileSdkVersion 30
    compileOptions{
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = "1.8"
    }

    defaultConfig {
        applicationId "my.edu.utar.pwmanager"
        minSdkVersion 23
        targetSdkVersion 30
        versionCode 1
        versionName "0.1.4"
        resConfigs 'en'
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled true
            shrinkResources true
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }

}

dependencies {
    implementation 'androidx.security:security-crypto:1.1.0-alpha03'

    //Room
    implementation "androidx.room:room-runtime:2.3.0"
    annotationProcessor "androidx.room:room-compiler:2.3.0"

    implementation fileTree(dir: 'libs', include: ['*.jar'])

    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'

    implementation 'commons-codec:commons-codec:20041127.091804'
    implementation 'androidx.preference:preference-ktx:1.1.1'

    implementation 'com.github.ihimanshurawat:Hasher:1.2'

    //Updater
    implementation 'com.github.javiersantos:AppUpdater:2.7'
```

```
    //ROOM Backup
    implementation
'com.github.salehyarahmadi:RoomDatabaseBackupAndRestore:v1.0.1'

    //Pin Lock
    implementation 'com.andrognito.pinlockview:pinlockview:2.1.0'

    //circle imageview
    implementation 'de.hdodenhof:circleimageview:3.1.0'


    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.navigation:navigation-fragment-ktx:2.3.5'
    implementation 'androidx.navigation:navigation-ui-ktx:2.3.5'
    implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'
    implementation 'androidx.recyclerview:recyclerview:1.2.1'

    testImplementation 'junit:junit:4.13'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    // ViewModel
    implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1"
    // LiveData
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.3.1"
    // Saved state module for ViewModel
    implementation "androidx.lifecycle:lifecycle-viewmodel-savedstate:2.3.1"

    // Annotation processor
    annotationProcessor "androidx.lifecycle:lifecycle-common-java8:2.3.1"

    implementation 'androidx.cardview:cardview:1.0.0'

    implementation 'androidx.biometric:biometric:1.1.0'

    implementation 'androidx.core:core-ktx:1.7.0-alpha01'
    implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1"
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"

}
repositories {
    mavenCentral()
}
```