Custom project progress report

COS30017 Software Development for Mobile Devices 2023

Trung Kien Nguyen
104053642

Table of Contents

Overview of project

My project is a cryptocurrency app using Android Development with Kotlin. Like many typical apps in the market, such as CoinGecko, Coinbase, Kraken, etc., my app has some basic functionalities as follows:

- User authentication: The app allows users to create accounts, log in with them, or reset their password in case of forgetting.
- Wallet managing: It also creates and manages cryptocurrency wallets for different cryptocurrencies, e.g. Bitcoin, Etherium, ….
- Viewing transaction history
- Buy and sell cryptocurrencies as will.

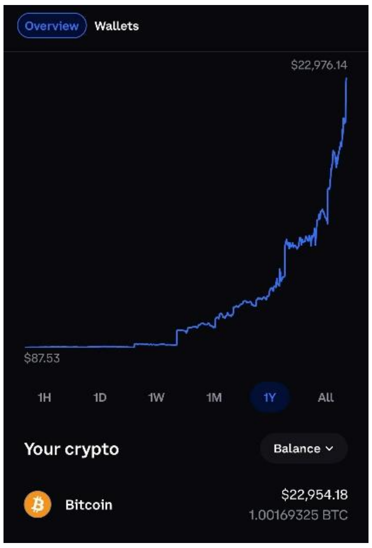
Weekly reports

Week 7

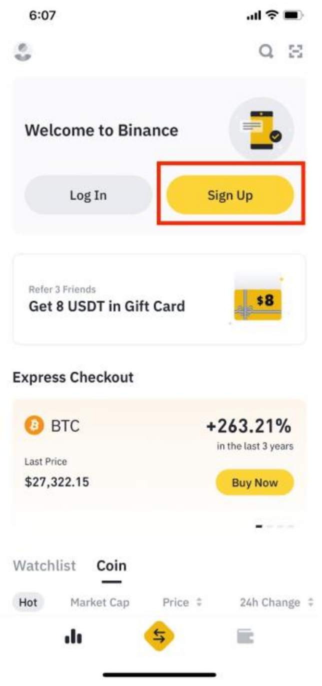
Note: I have completed Core 1, submitted Core 2, but not completed the discussion tasks. However, I decided to continue with the custom project as the resubmission for those discussions would not be available until 6 Oct 2023 (end of week 9)

In this step, I focused mainly in searching and referring to the cryptocurrency applications on the market, especially those available on the Android platform:

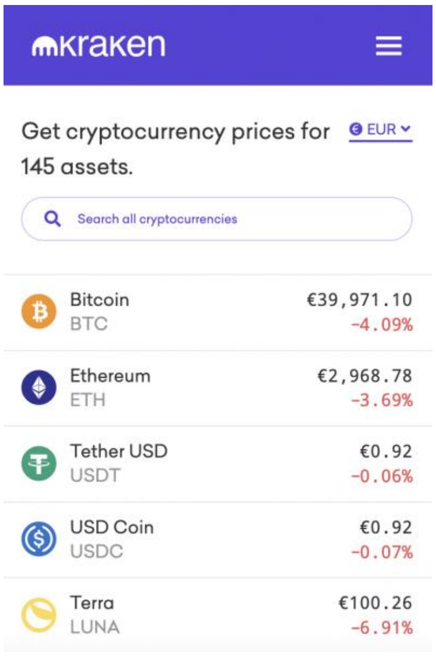
- Coinbase: In my opinion, this app has an user-friendly interface, ideal for beginners. It offers a secure and regulated platform for buying, selling, and managing cryptocurrencies.



- Binance: This offers not only a wide range of cryptocurrencies for trading, but also additional features like advanced charting tools. It's one of the most popular app among experienced traders.



- Kraken: I firmly believe that this app is well-known for its strong security features, providing access to a variety of cryptocurrencies and trading pairs.


mkraken

Get cryptocurrency prices for 145 assets. EUR

Search all cryptocurrencies

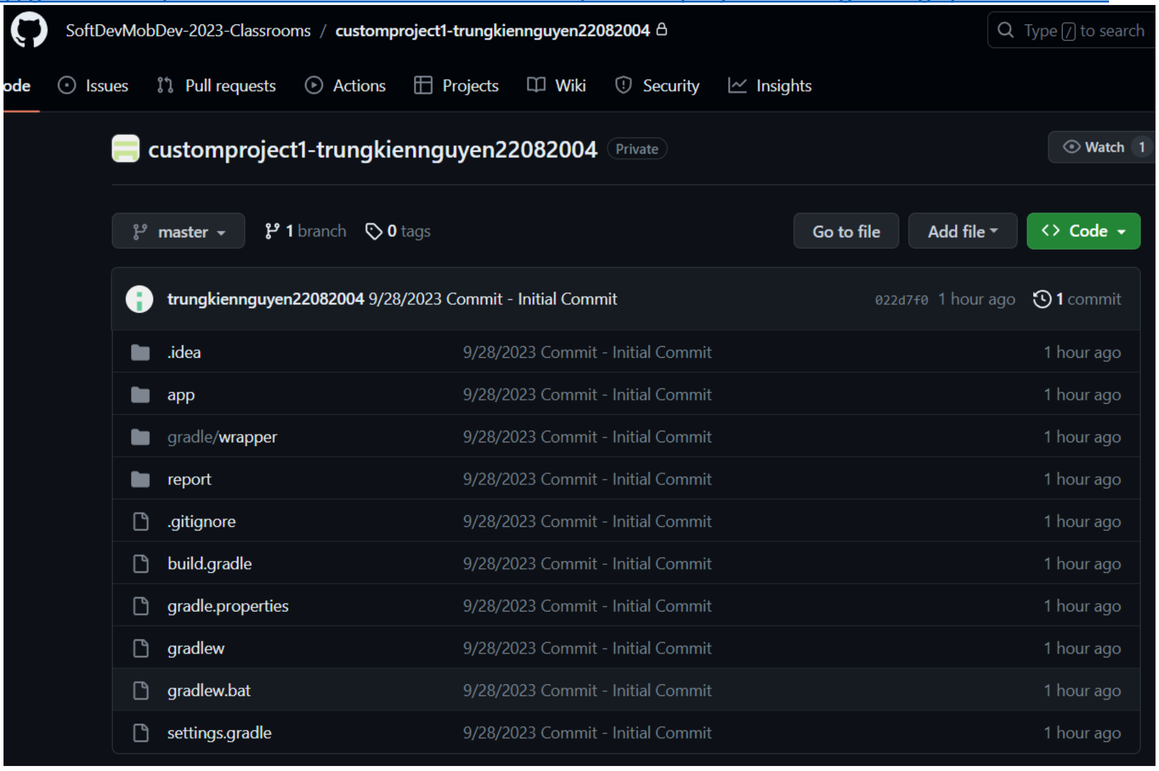
Bitcoin	BTC	€39,971.10	-4.09%
Ethereum	ETH	€2,968.78	-3.69%
Tether USD	USDT	€0.92	-0.06%
USD Coin	USDC	€0.92	-0.07%
Terra	LUNA	€100.26	-6.91%

In the following week (Week 8), I think I will try to complete the Authentication feature of the app (Login/Signup). I prefer to use the Google Firebase's authentication functionality.

Week 8

Note: I have completed Core 1, submitted Core 2, but not completed the discussion tasks. However, I decided to continue with the custom project as the resubmission for those discussions would not be available until 6 Oct 2023 (end of week 9)

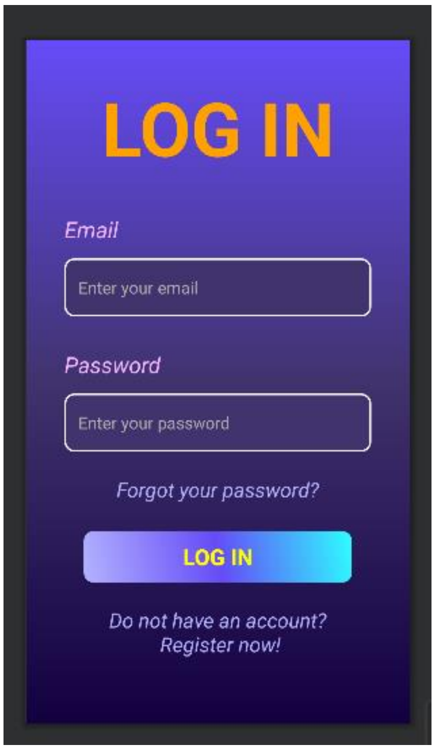
To begin with, I have set up the repository for the Custom Project via the Github Classroom link on Canvas: https://github.com/SoftDevMobDev-2023-Classrooms/customproject1-trungkiennguyen22082004



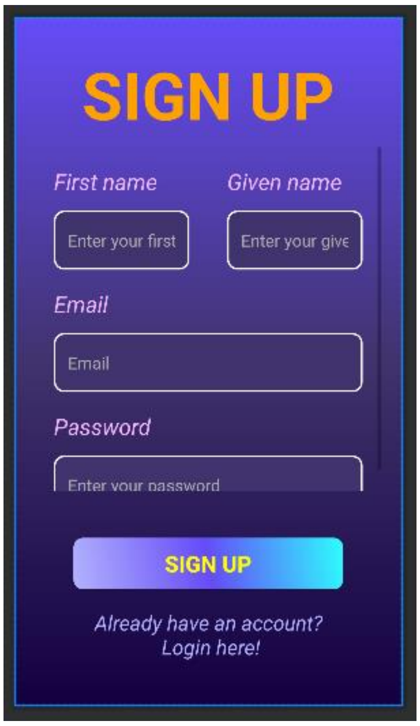
trungkiennguyen22082004 9/28/2023 Commit - Initial Commit 022d7f0 1 hour ago 1 commit

.idea	9/28/2023 Commit - Initial Commit	1 hour ago
app	9/28/2023 Commit - Initial Commit	1 hour ago
gradle/wrapper	9/28/2023 Commit - Initial Commit	1 hour ago
report	9/28/2023 Commit - Initial Commit	1 hour ago
.gitignore	9/28/2023 Commit - Initial Commit	1 hour ago
build.gradle	9/28/2023 Commit - Initial Commit	1 hour ago
gradle.properties	9/28/2023 Commit - Initial Commit	1 hour ago
gradlew	9/28/2023 Commit - Initial Commit	1 hour ago
gradlew.bat	9/28/2023 Commit - Initial Commit	1 hour ago
settings.gradle	9/28/2023 Commit - Initial Commit	1 hour ago

For the Authentication task, I have implemented two activities, including LoginActivity and SignupActivity.



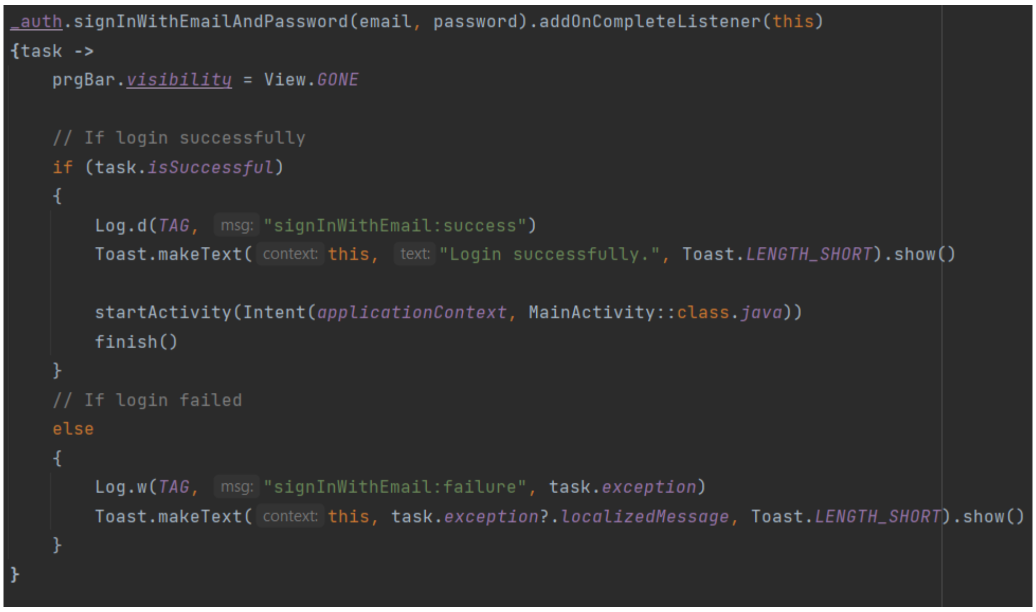
LoginActivity's layout



SignupActivity's layout

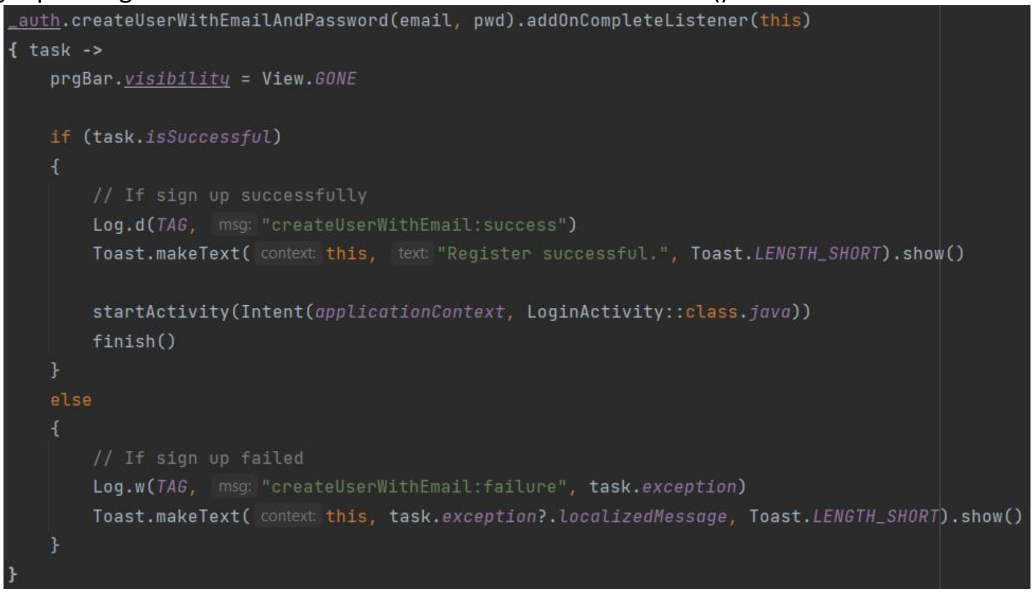
I have use the Google Firebase's functionality of Authentication using Email/Password:

- Login: Using the method "signInWithEmailAndPassword()" of FirebaseAuth



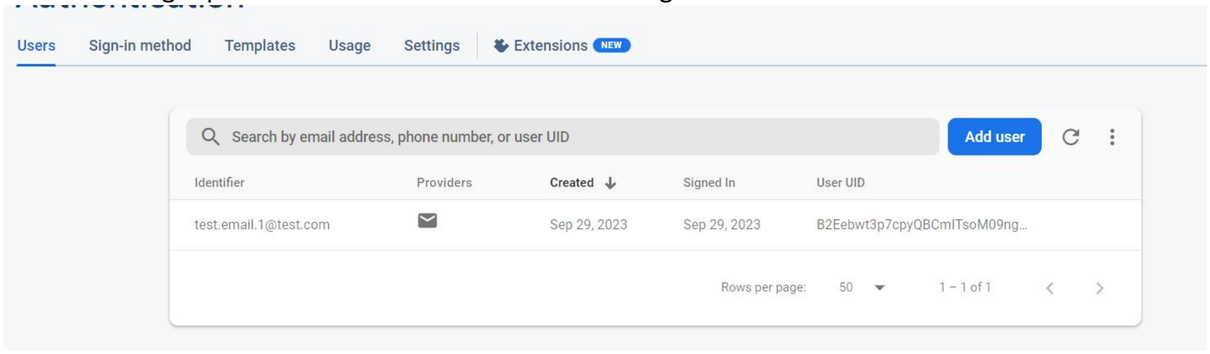
```
_auth.signInWithEmailAndPassword(email, password).addOnCompleteListener(this)
{task ->
    prgBar.visibility = View.GONE

    // If login successfully
    if (task.isSuccessful)
    {
        Log.d(TAG,  msg: "signInWithEmail:success")
        Toast.makeText( context: this,  text: "Login successfully.", Toast.LENGTH_SHORT).show()

        startActivity(Intent(applicationContext, MainActivity::class.java))
        finish()
    }
    // If login failed
    else
    {
        Log.w(TAG,  msg: "signInWithEmail:failure", task.exception)
        Toast.makeText( context: this, task.exception?.localizedMessage, Toast.LENGTH_SHORT).show()
    }
}
```

- Signup: Using the method “createUserWithEmailAndPassword()” of FirebaseAuth



```
_auth.createUserWithEmailAndPassword(email, pwd).addOnCompleteListener(this)
{ task ->
    prgBar.visibility = View.GONE

    if (task.isSuccessful)
    {
        // If sign up successfully
        Log.d(TAG,  msg: "createUserWithEmail:success")
        Toast.makeText( context: this,  text: "Register successful.", Toast.LENGTH_SHORT).show()

        startActivity(Intent(applicationContext, LoginActivity::class.java))
        finish()
    }
    else
    {
        // If sign up failed
        Log.w(TAG,  msg: "createUserWithEmail:failure", task.exception)
        Toast.makeText( context: this, task.exception?.localizedMessage, Toast.LENGTH_SHORT).show()
    }
}
```

I have use “Signup” to create some new users for testing:

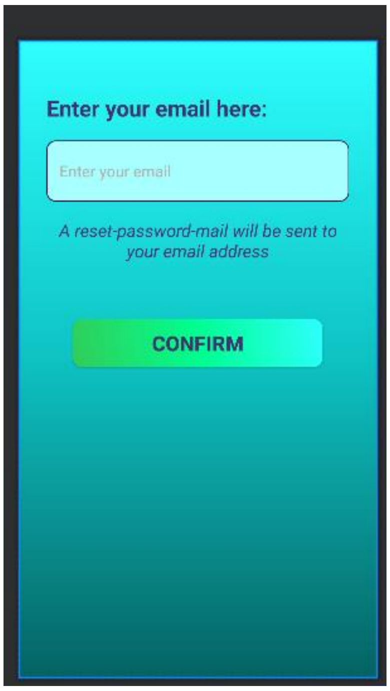


Identifier	Providers	Created	Signed In	User UID
test.email.1@test.com	✉	Sep 29, 2023	Sep 29, 2023	B2Eebwt3p7cpyQBCmITsoM09ng...

Rows per page: 50   1 – 1 of 1

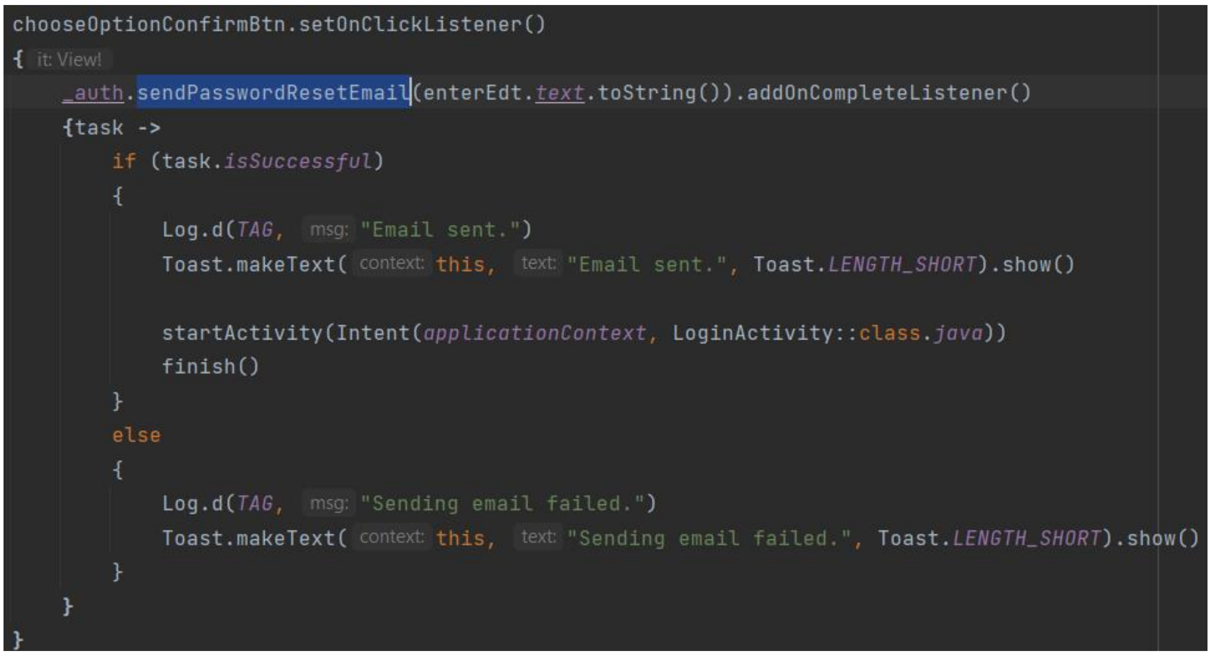
Week 9

Note: I have completed Core 1, completed Core 2, I have just submitted the Redo for discussion tasks since they had been available in the Friday morning.
Firstly, I continued my work on the Authentication functionality of my custom app, adding the forgot-password and reset-password feature:



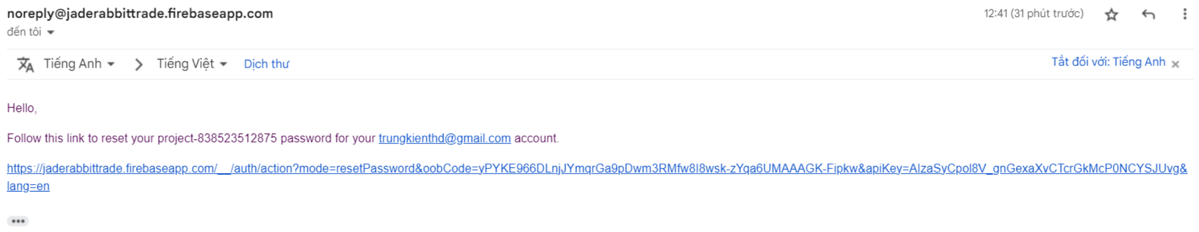
ForgotPasswordActivity's layout

I have used the "sendPasswordResetEmal()" method to sent a resetting-password email to the entered email address:



```
chooseOptionConfirmBtn.setOnClickListener()
{ it: View!
    _auth.sendPasswordResetEmail(enterEdt.text.toString()).addOnCompleteListener()
    {task ->
        if (task.isSuccessful)
        {
            Log.d(TAG, msg: "Email sent.")
            Toast.makeText( context: this,  text: "Email sent.", Toast.LENGTH_SHORT).show()

            startActivity(Intent(applicationContext, LoginActivity::class.java))
            finish()
        }
        else
        {
            Log.d(TAG, msg: "Sending email failed.")
            Toast.makeText( context: this,  text: "Sending email failed.", Toast.LENGTH_SHORT).show()
        }
    }
}
```

I have tested that functionality, the resetting-password email will look like:



noreply@jaderabbittrade.firebaseapp.com
đến tôi

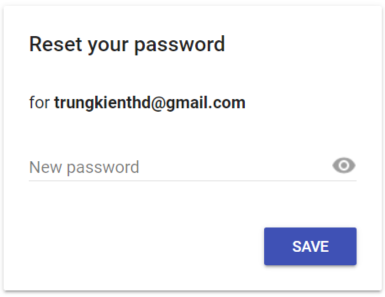
12:41 (31 phút trước)

Hello,

Follow this link to reset your project-838523512875 password for your trungkienthd@gmail.com account.

https://jaderabbittrade.firebaseapp.com/__/auth/action?mode=resetPassword&oobCode=yPYKE966DLnjJYmqrGa9pDwm3RMfw8I8wsk-zYqa6UMAAAGK-Fipkw&apiKey=AIzaSyCpol8V_gnGexaXvCTcrGkMcP0NCYSJUvg&lang=en

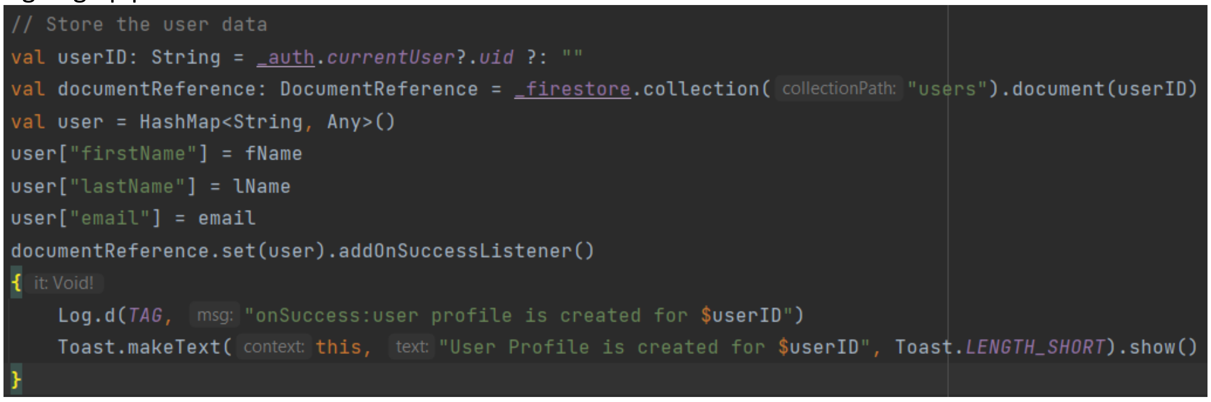
And this is the resetting-password interface:


Reset your password

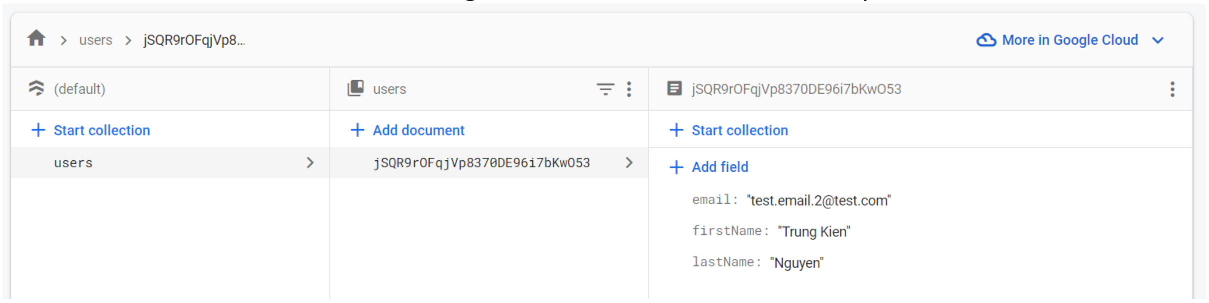
for **trungkienthd@gmail.com**

New password

Next, I stored the User data (First Name, Last Name, …) in Firebase's feature of Cloud Firestore
After create a Cloud Firestore Database, I have added the "Firestore.DocumentReference.set(user)" method to store the First name, last name and email of the user to the Firebase Cloud Firestore's database during the signing up process:


```
// Store the user data
val userID: String = _auth.currentUser?.uid ?: ""
val documentReference: DocumentReference = _firestore.collection( collectionPath: "users").document(userID)
val user = HashMap<String, Any>()
user["firstName"] = fName
user["lastName"] = lName
user["email"] = email
documentReference.set(user).addOnSuccessListener()
{ it: Void!
    Log.d(TAG,  msg: "onSuccess:user profile is created for $userID")
    Toast.makeText( context: this,  text: "User Profile is created for $userID", Toast.LENGTH_SHORT).show()
}
```

I have created a new account for testing this function, and this is the output:



email: "test.email.2@test.com"
firstName: "Trung Kien"
lastName: "Nguyen"

In the following weeks, I will continue to make the Market activity (for displaying digital assets of cryptocurrency), Transaction activity (for displaying history transaction of the user), and probably the User Activity (for viewing and modifying the user's details)

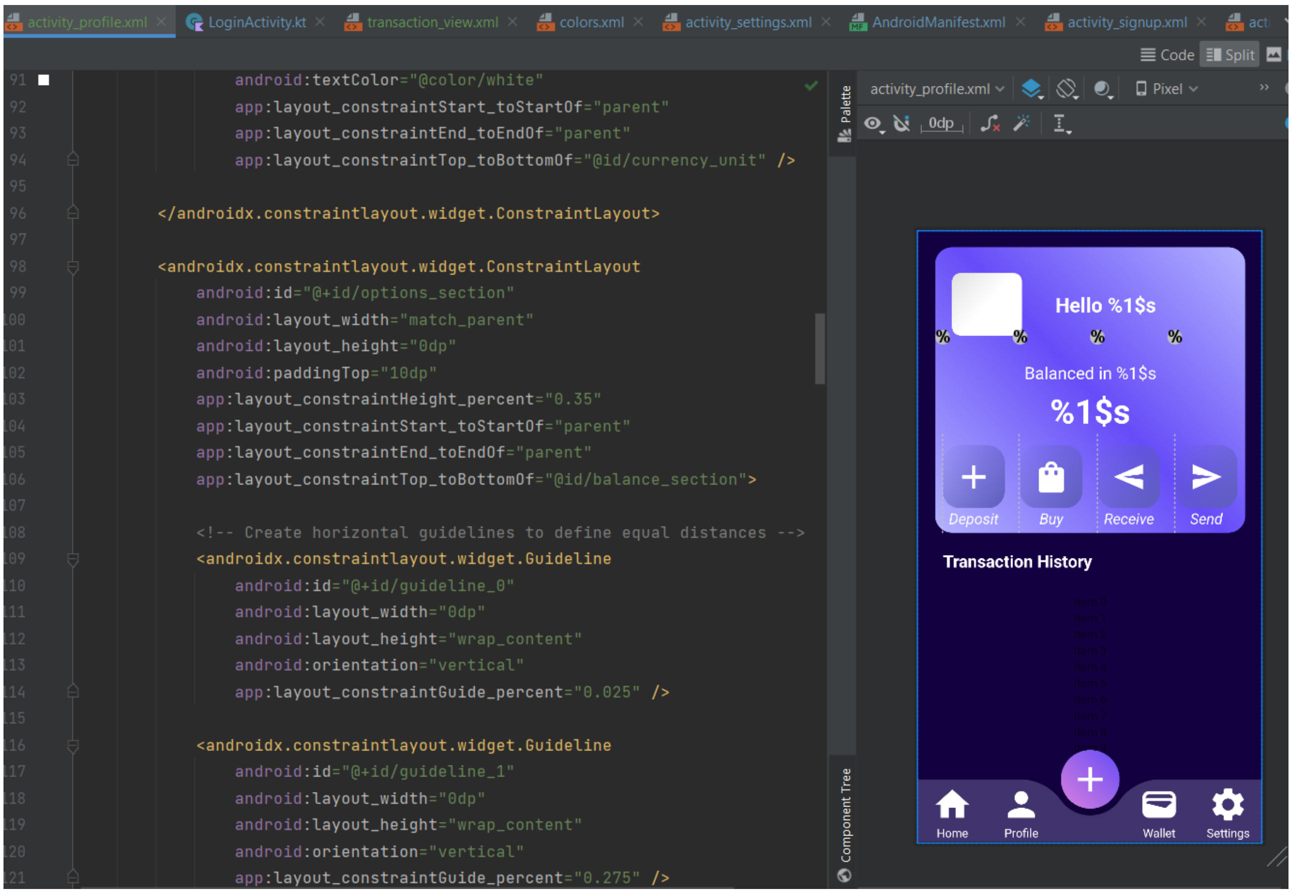
Week 10

[NOTE: as of week 10, if you have not completed Core 1, not completed Core, not submitted Core 3 nor an extension task, you will also need to justify why you should be encouraged to continue with a custom project. There is no point focusing on this task when the basics are not complete and your progress report will be marked as incomplete.]

Note: I have added the design for the main activities for my app in the "Level 1: Design evidence" part

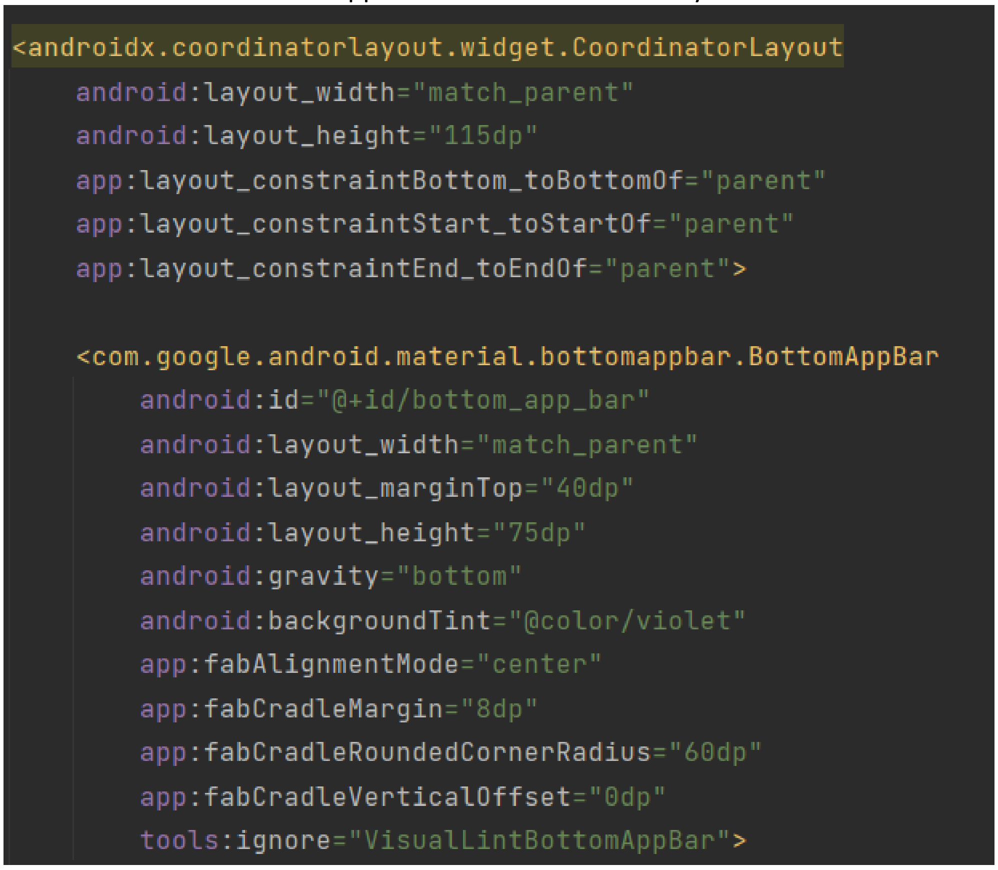
- Profile Activity:

The layout for this activity was quite easily to created for me, using a number of ConstraintLayouts and ConstraintLayout.widget.GuideLines:



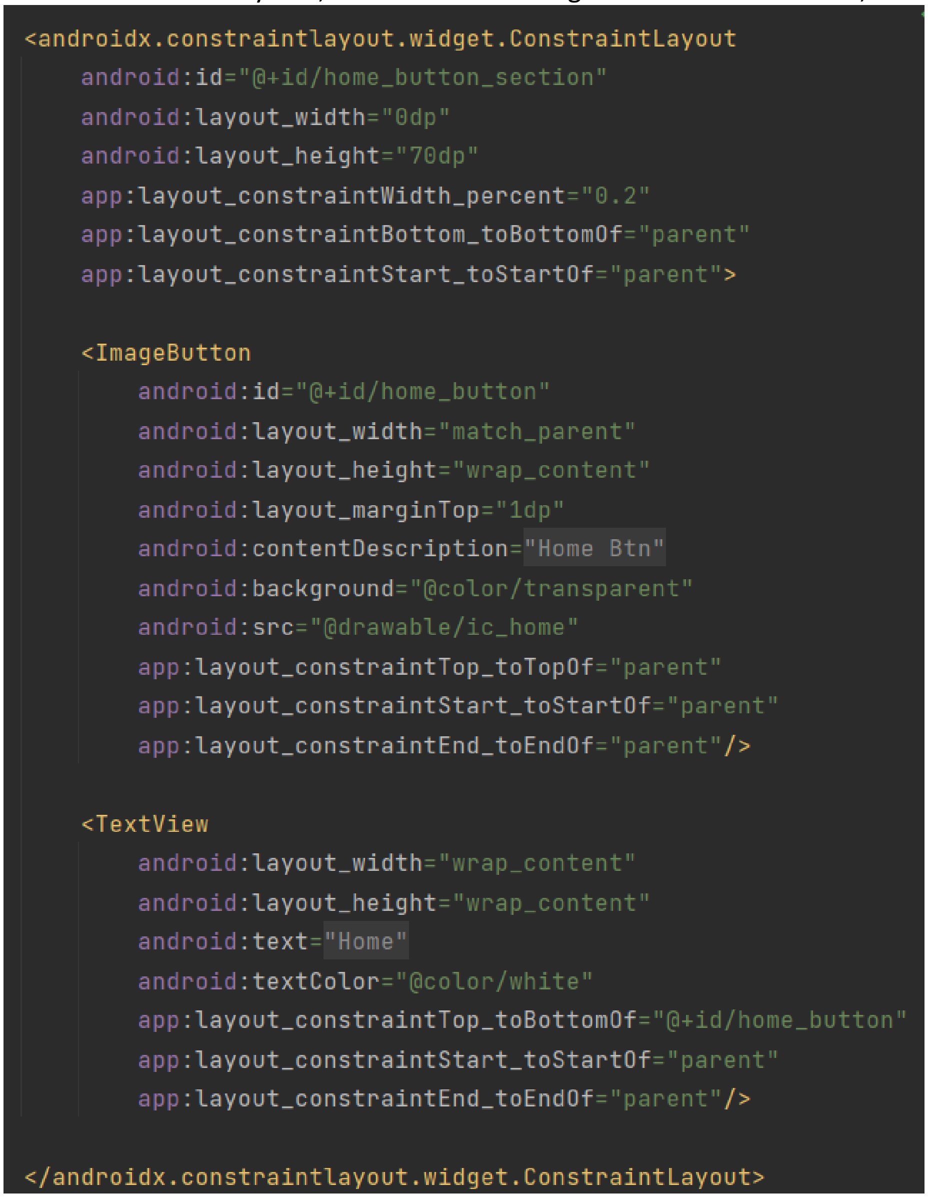
```
                android:textColor="@color/white"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintTop_toBottomOf="@id/currency_unit" />

        </androidx.constraintlayout.widget.ConstraintLayout>

        <androidx.constraintlayout.widget.ConstraintLayout
            android:id="@+id/options_section"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:paddingTop="10dp"
            app:layout_constraintHeight_percent="0.35"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toBottomOf="@id/balance_section">

            <!-- Create horizontal guidelines to define equal distances -->
            <androidx.constraintlayout.widget.Guideline
                android:id="@+id/guideline_0"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                app:layout_constraintGuide_percent="0.025" />

            <androidx.constraintlayout.widget.Guideline
                android:id="@+id/guideline_1"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                app:layout_constraintGuide_percent="0.275" />
```

For the Navigation tab, currently I left it inside the Profile Activity instead of creating it in a separate layout file. I have used the BottomAppBar inside a CoordinatorLayout for the base of the navbar:



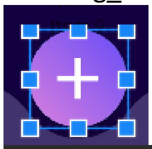
```
<androidx.coordinatorlayout.widget.CoordinatorLayout
    android:layout_width="match_parent"
    android:layout_height="115dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent">

    <com.google.android.material.bottomappbar.BottomAppBar
        android:id="@+id/bottom_app_bar"
        android:layout_width="match_parent"
        android:layout_marginTop="40dp"
        android:layout_height="75dp"
        android:gravity="bottom"
        android:backgroundTint="@color/violet"
        app:fabAlignmentMode="center"
        app:fabCradleMargin="8dp"
        app:fabCradleRoundedCornerRadius="60dp"
        app:fabCradleVerticalOffset="0dp"
        tools:ignore="VisualLintBottomAppBar">
```

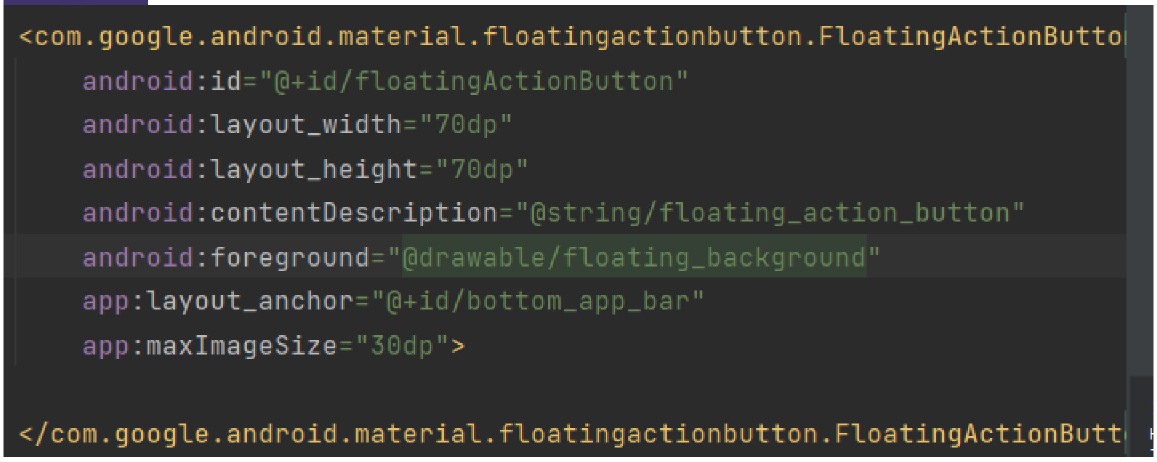
And the ConstraintLayouts, each contains a ImageButton and a TextView, for the functionality buttons:



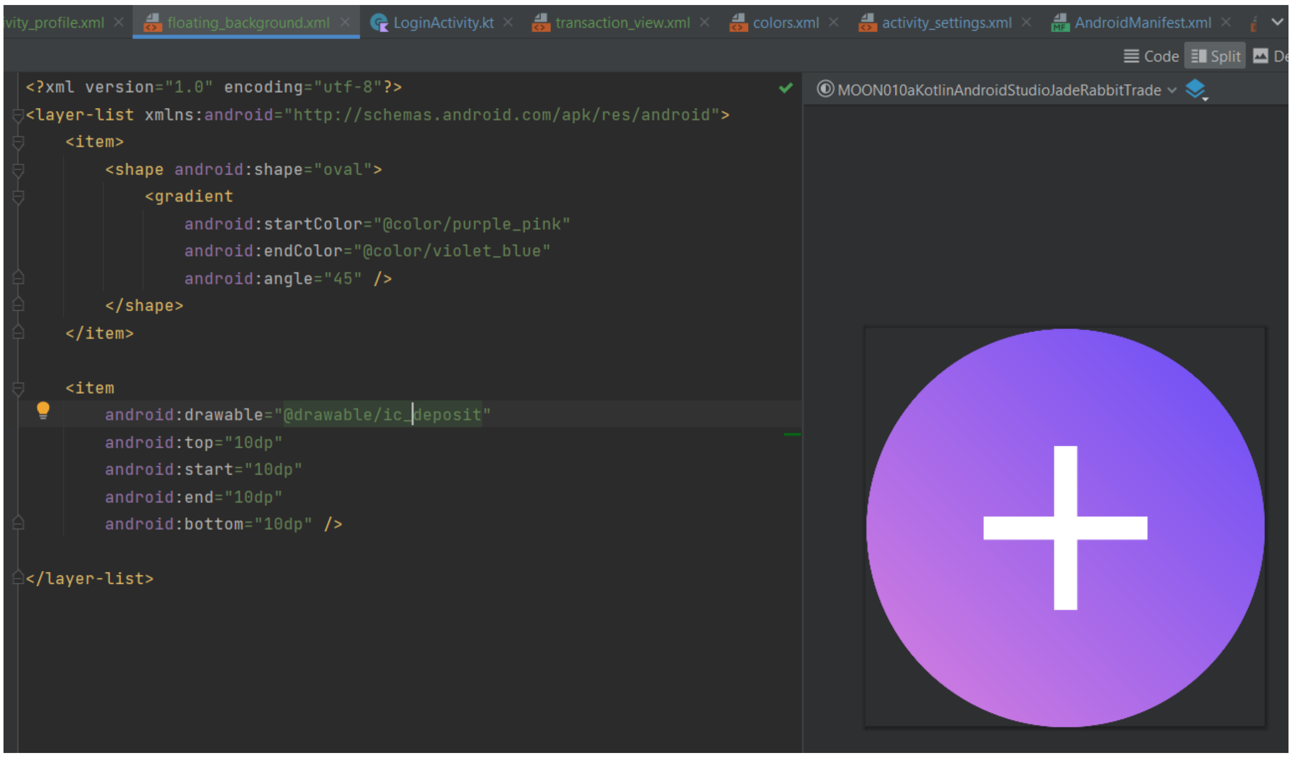
```xml
<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/home_button_section"
    android:layout_width="0dp"
    android:layout_height="70dp"
    app:layout_constraintWidth_percent="0.2"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent">

    <ImageButton
        android:id="@+id/home_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="1dp"
        android:contentDescription="Home Btn"
        android:background="@color/transparent"
        android:src="@drawable/ic_home"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Home"
        android:textColor="@color/white"
        app:layout_constraintTop_toBottomOf="@+id/home_button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Following that, a float button of the navbar was created using FloatingActionButton, with the drawable of "floating_background" as the foreground, which are shown as below:



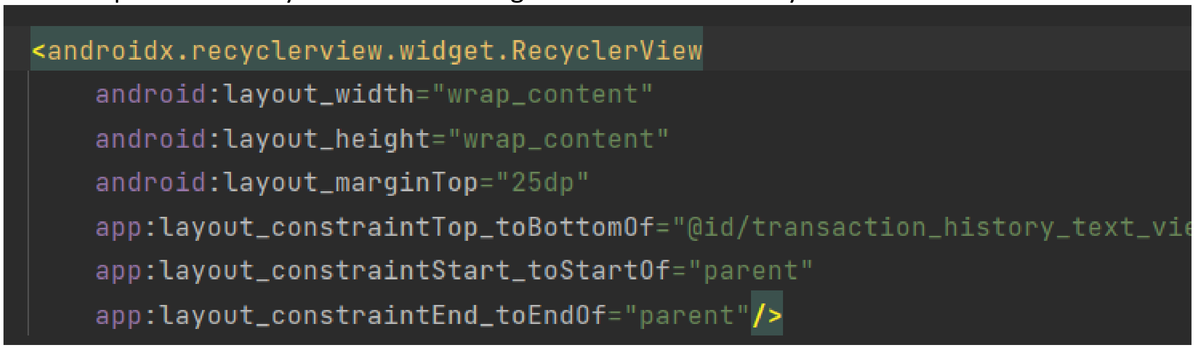


```xml
<com.google.android.material.floatingactionbutton.FloatingActionButto
    android:id="@+id/floatingActionButton"
    android:layout_width="70dp"
    android:layout_height="70dp"
    android:contentDescription="@string/floating_action_button"
    android:foreground="@drawable/floating_background"
    app:layout_anchor="@+id/bottom_app_bar"
    app:maxImageSize="30dp">

</com.google.android.material.floatingactionbutton.FloatingActionButt
```



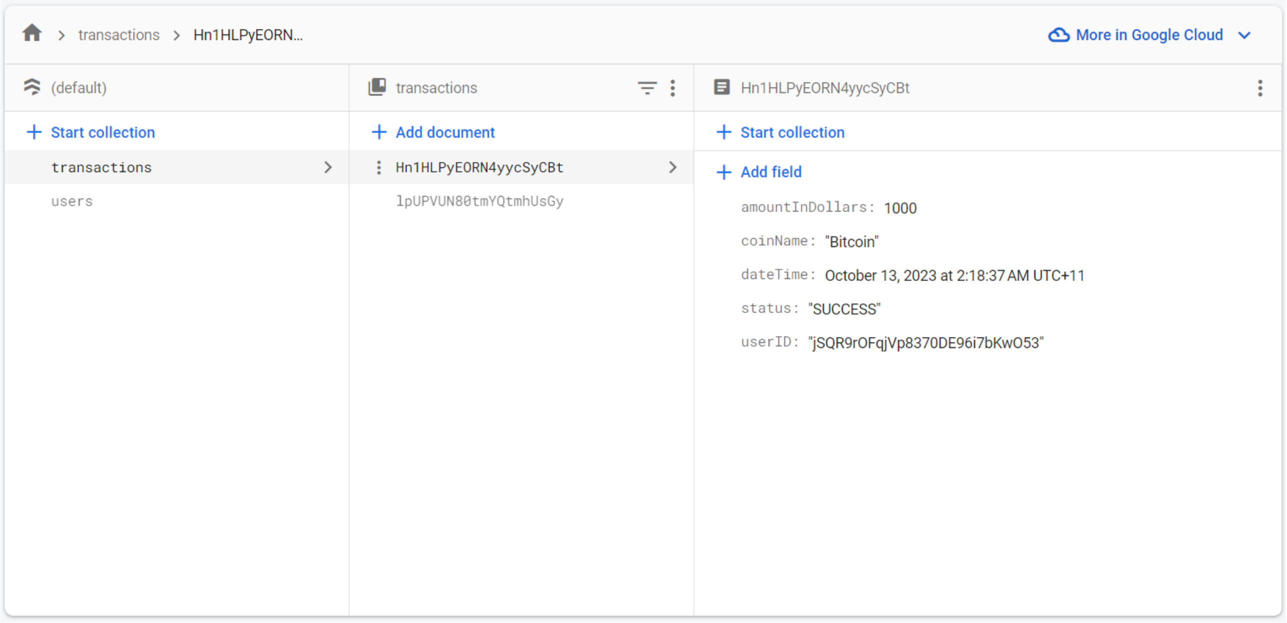
```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape android:shape="oval">
            <gradient
                android:startColor="@color/purple_pink"
                android:endColor="@color/violet_blue"
                android:angle="45" />
        </shape>
    </item>

    <item
        android:drawable="@drawable/ic_deposit"
        android:top="10dp"
        android:start="10dp"
        android:end="10dp"
        android:bottom="10dp" />

</layer-list>
```

The next part is the RecyclerView consisting the transaction history:



```
<androidx.recyclerview.widget.RecyclerView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dp"
    app:layout_constraintTop_toBottomOf="@id/transaction_history_text_vie
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"/>
```

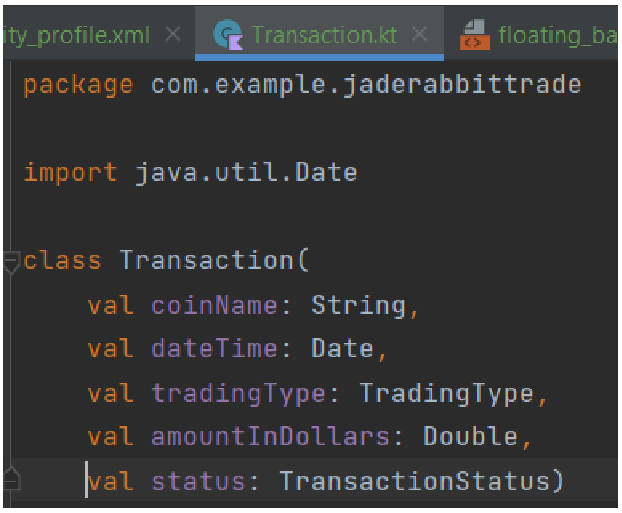
For the database, I have again used the Firestore functionality, creating a database named “transaction”:



amountInDollars: 1000
coinName: "Bitcoin"
dateTime: October 13, 2023 at 2:18:37 AM UTC+11
status: "SUCCESS"
userID: "jSQR9rOFqjVp8370DE96i7bKwO53"

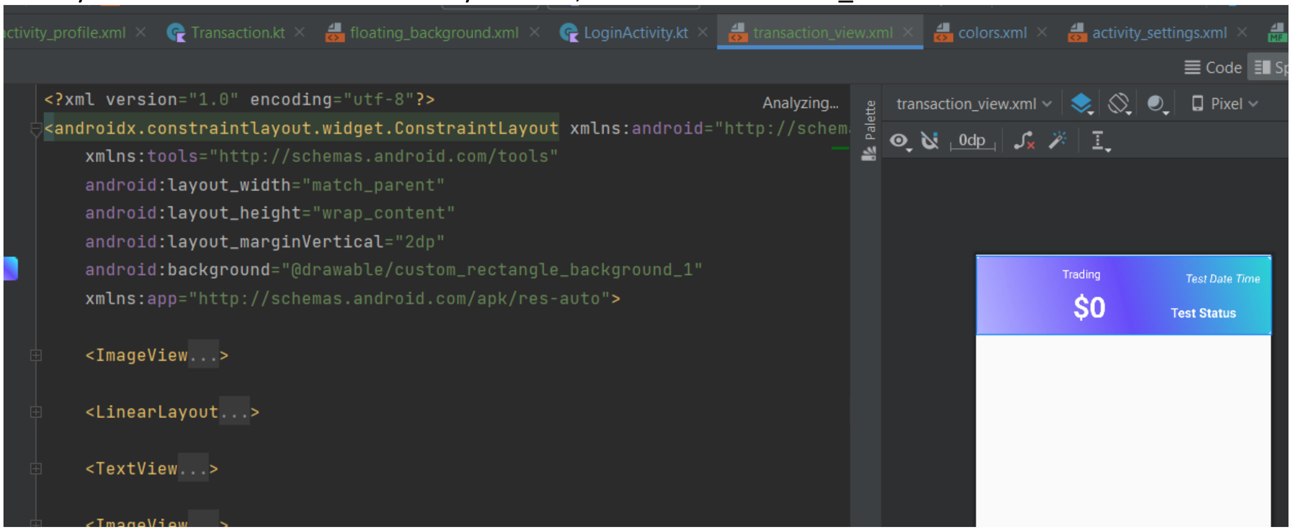
The records were entered manually by me instead of implementing in the code, as the Buy/sell functionality of the HomeActivity had not been done yet

And the corresponding Kotlin class, storing the information of a transaction, named Transaction:



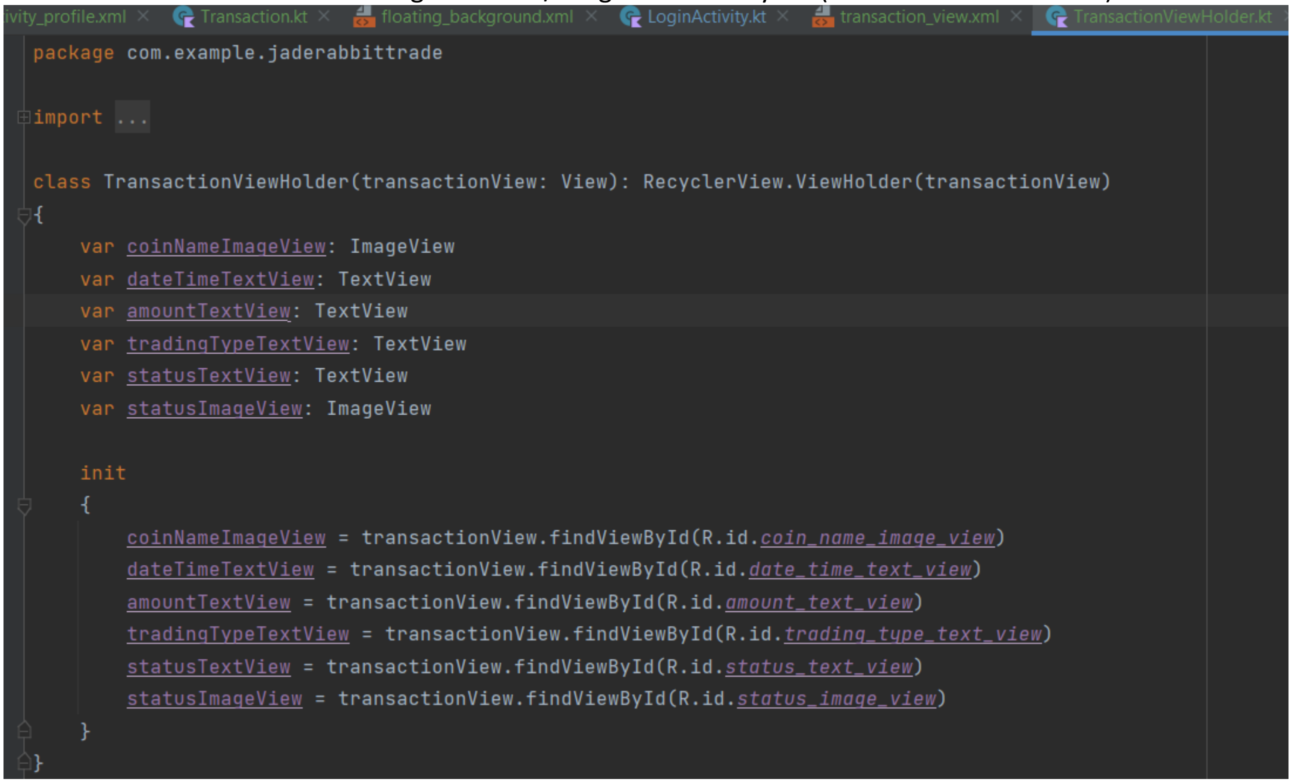
```kotlin
package com.example.jaderabbittrade

import java.util.Date

class Transaction(
    val coinName: String,
    val dateTime: Date,
    val tradingType: TradingType,
    val amountInDollars: Double,
    val status: TransactionStatus)
```

The layout for each element in the RecyclerView, which is “transaction_view.xml”:



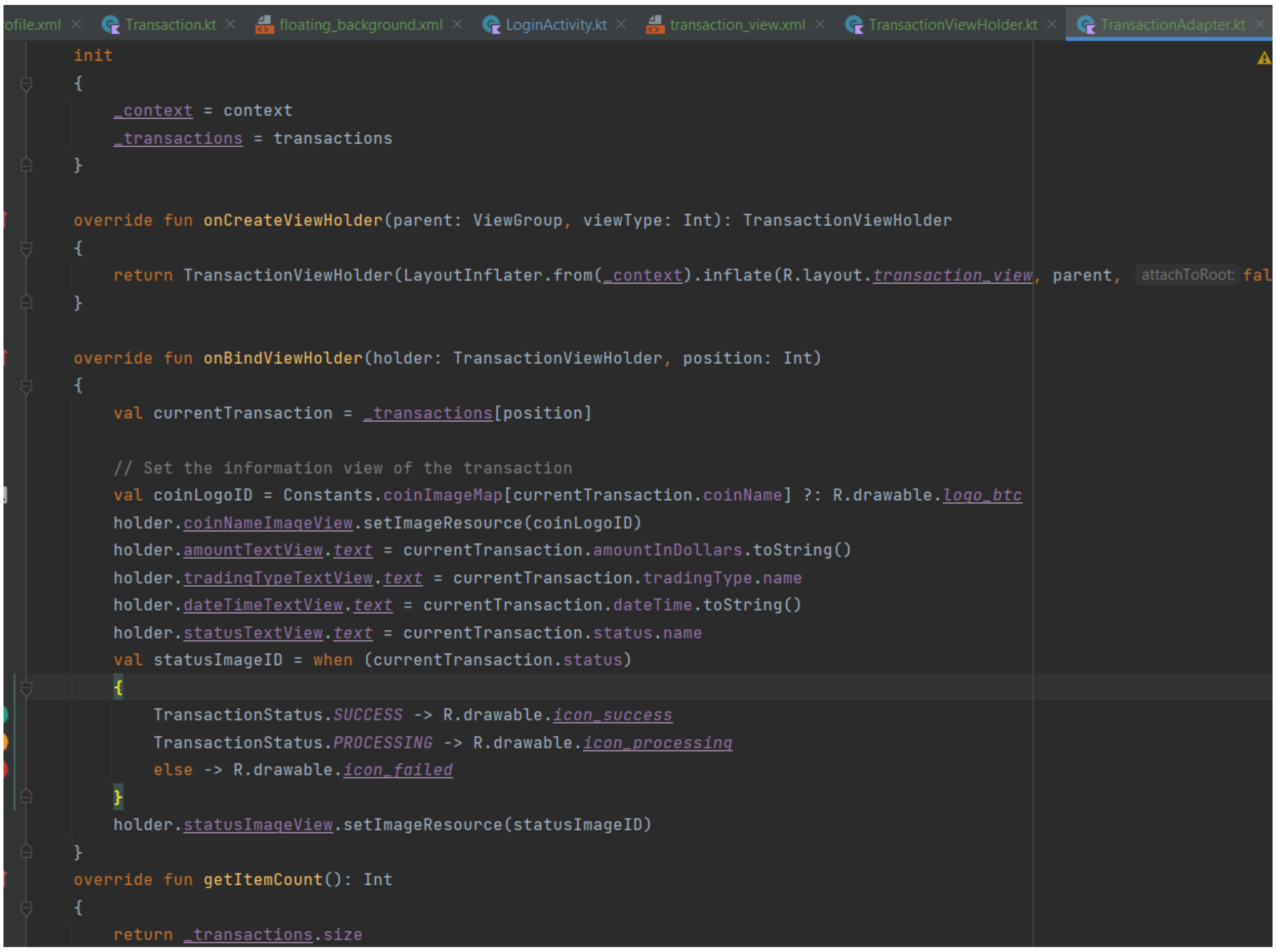
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schem
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginVertical="2dp"
    android:background="@drawable/custom_rectangle_background_1"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <ImageView...>

    <LinearLayout...>

    <TextView...>

    <ImageView...>
```

The ViewHolder class for initializing the views/widgets in that layout (TransactionViewHolder):



```kotlin
package com.example.jaderabbittrade

import ...

class TransactionViewHolder(transactionView: View): RecyclerView.ViewHolder(transactionView)
{
    var coinNameImageView: ImageView
    var dateTimeTextView: TextView
    var amountTextView: TextView
    var tradingTypeTextView: TextView
    var statusTextView: TextView
    var statusImageView: ImageView

    init
    {
        coinNameImageView = transactionView.findViewById(R.id.coin_name_image_view)
        dateTimeTextView = transactionView.findViewById(R.id.date_time_text_view)
        amountTextView = transactionView.findViewById(R.id.amount_text_view)
        tradingTypeTextView = transactionView.findViewById(R.id.trading_type_text_view)
        statusTextView = transactionView.findViewById(R.id.status_text_view)
        statusImageView = transactionView.findViewById(R.id.status_image_view)
    }
}
```

And the adapter for the RecyclerView, using those TransactionViewHolders:



ofile.xml × | Transaction.kt × | floating_background.xml × | LoginActivity.kt × | transaction_view.xml × | TransactionViewHolder.kt × | TransactionAdapter.kt ×

```
init
{
    _context = context
    _transactions = transactions
}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TransactionViewHolder
{
    return TransactionViewHolder(LayoutInflater.from(_context).inflate(R.layout.transaction_view, parent,  attachToRoot: fal
}

override fun onBindViewHolder(holder: TransactionViewHolder, position: Int)
{
    val currentTransaction = _transactions[position]

    // Set the information view of the transaction
    val coinLogoID = Constants.coinImageMap[currentTransaction.coinName] ?: R.drawable.logo_btc
    holder.coinNameImageView.setImageResource(coinLogoID)
    holder.amountTextView.text = currentTransaction.amountInDollars.toString()
    holder.tradingTypeTextView.text = currentTransaction.tradingType.name
    holder.dateTimeTextView.text = currentTransaction.dateTime.toString()
    holder.statusTextView.text = currentTransaction.status.name
    val statusImageID = when (currentTransaction.status)
    {
        TransactionStatus.SUCCESS -> R.drawable.icon_success
        TransactionStatus.PROCESSING -> R.drawable.icon_processing
        else -> R.drawable.icon_failed
    }
    holder.statusImageView.setImageResource(statusImageID)
}
override fun getItemCount(): Int
{
    return _transactions.size
```

I'm in progress of creating Home and Settings activities with the floating navbar, which is expected to be done by this Sunday (10/15/2023)

My questions for this week:

- For the task "Architectural design, with the diagrams as an appendix", do I have to include them here on the weekly reports, or just add in the final one? If they need to be added here, can I add them in the next (final) weekly report?
- Can you please give some further instructions for this requirement, or may be an example? I have read that "UML might not be suitable for your app", but is it satisfactory to use them?

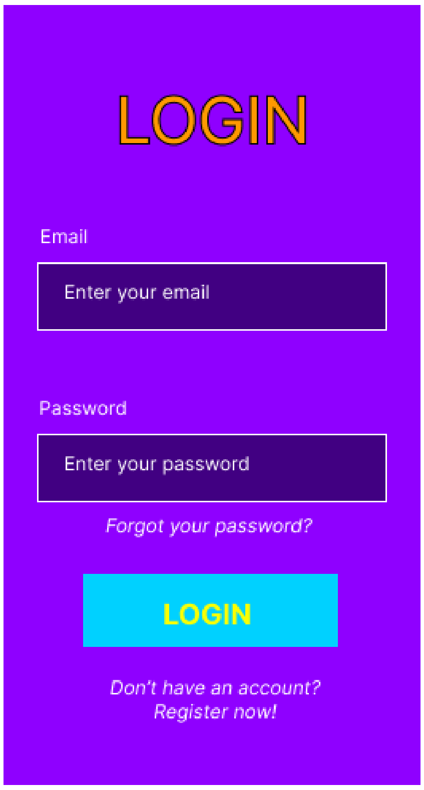
Week 11

[NOTE: as of week 11, if you have not submitted/completed all other Core/Extension tasks, you will also need to justify why you should be encouraged to continue with a custom project. There is no point focusing on this task when the basics are not complete and your progress report will be marked as incomplete.]

Level 1: Design evidence

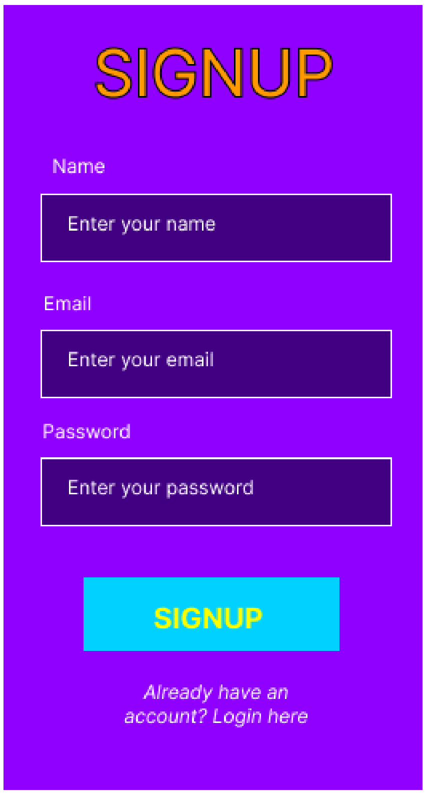
This following section includes some sketches and basic functionality of the main activities for my app:

- Login Activity:



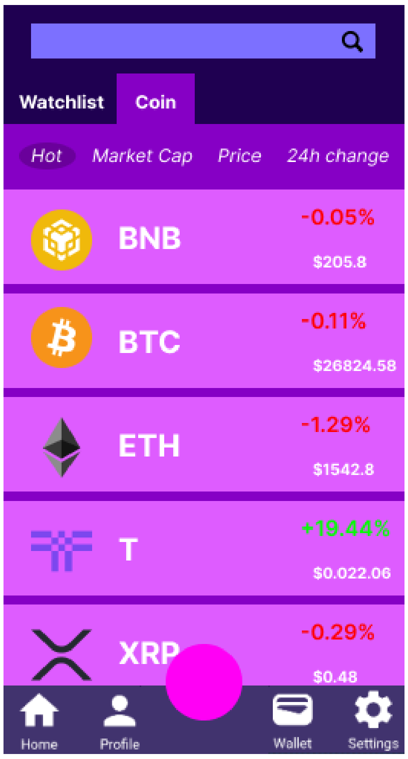
The login activity consists of two edit texts of email and password, which are required for a logging in session. The theme color will be purple, the same as the whole app.

- Signup Activity:



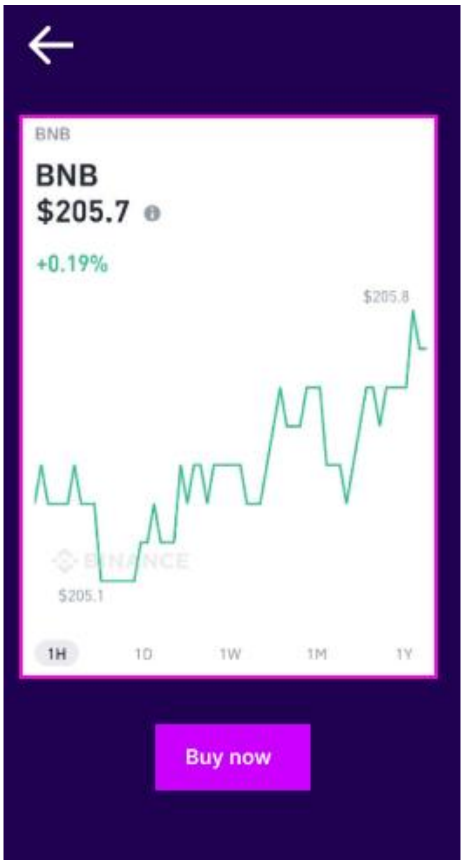
This activity is responsible for signing in purpose, using the similar layout and theme as the login activity.

- Home Activity/Fragment:



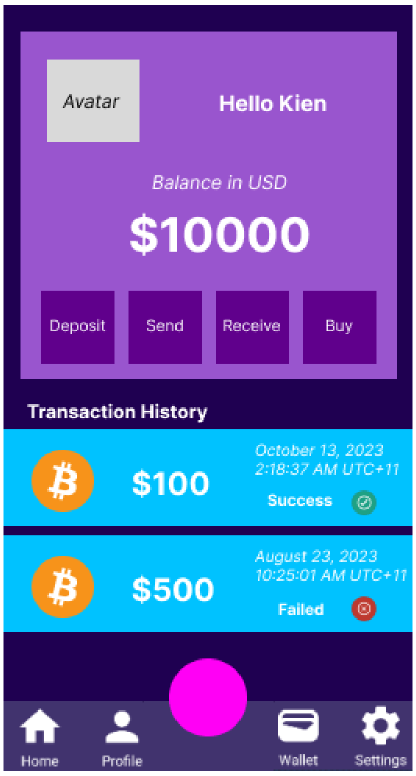
This is the base activity/fragment in the app, containing the list of purchasable cryptocurrencies in a Recycler View. This initially will have a simple search and filter function as shown in the sketch, but may be extended in a advance Searching filter using the Search Filter UI Pattern mentioned in my Assignment Extension 1. Moreover, these is a navbar created based on the Navigation Tab UI Pattern, helping the user to switch between some activities/fragments.

The layout for each's cryptocurrency's fragment, which will be displayed when clicking on each item, will be cloned from the Binance's one:



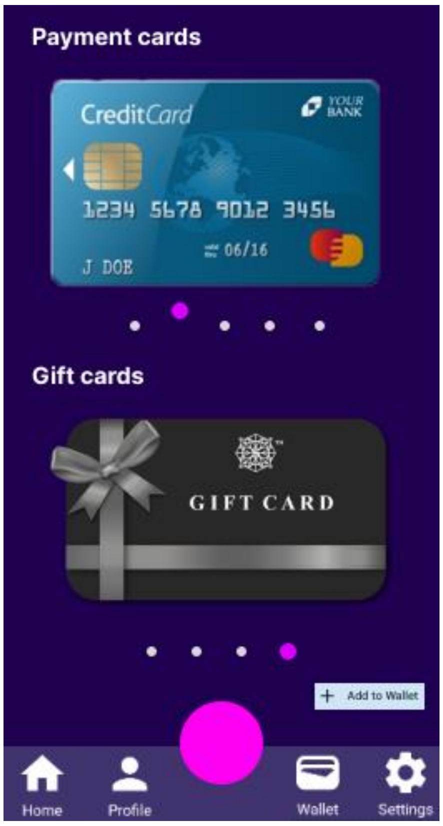
The buy/sell fragments will be designed simply, consisting of some TextViews and an EditText, so I think their sketches are unnecessary.

- Profile Activity/Fragment:



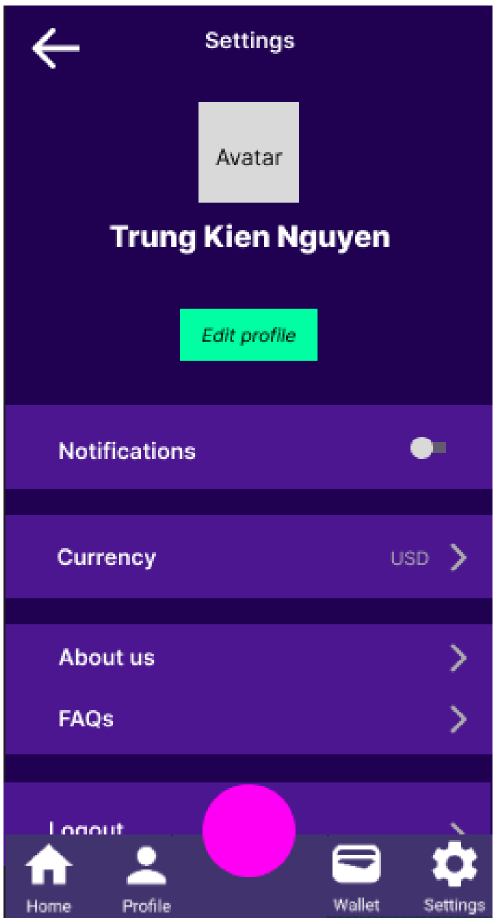
This contains some information of the User, including his/her full name, avatar, and most importantly the history of transaction as a Recycler View. This also has some functiionality button, allowing the user to deposit real-world money into, buying the cryptocurrency assets quickly (which can also be done in the Home Activity/Fragment). The Send/Receive buttons are optional, and can be replaced with Sell button.

- Wallet Activity/Fragment:



This is designed for the user to view/add their payment methods, including credit/debit cards, and gift cards, inspired by the layout of the app Google Wallet.

- Settings Activity/Fragment:



This activity/fragment is created for app settings, such as Profiles editting, Notifications allowing, Currency display, Information about the fictional organization, some FAQs, and a Logout buttons.

Level 2: App evidence

A database of “transactions” was created using Firestore functionality (provided by Google Firebase), storing the data of users’ history transaction.
The data of cryptocurrency assets (coin’s ID, current price (in dollars), recent changes in percentage, …) for the corresponding RecyclerView in the Home Activity will be set up using the same functionality.

Level 3: Extended research evidence

References