Mobile Application Development – Lab 3

Emma PERONNET – IOS3

1) Explain how you ensure the user is the right one starting the app

Thought process: The issue at stake here is to find how to restrict the access to the application. Since, it's a Bank App type, it requires maximum security. Which is why I didn't want to rely on simple authentication.

My first thought was to rely on another API and use a token. However, I didn't want to enlarge permissions and I didn't want to face security issues in the network as well. So, I decided my application would have as little interactions with the outside as possible, to shrink the surface of attack.

With this in mind, I started to enquire about biometry and came across this article[1]:

"**Why Biometric is Better Than Other Security Authentication Methods**

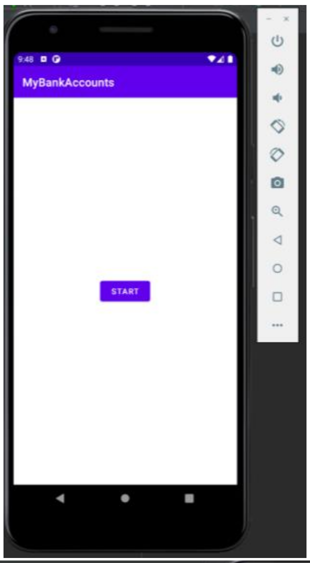
Here are the reasons why Biometrics tend to be more beneficial than other methods:

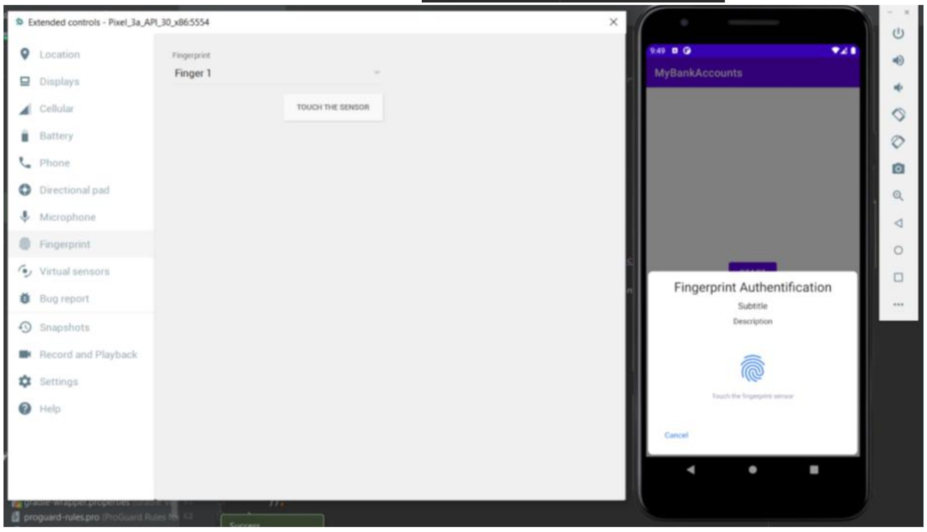
- Passwords and security patterns are easily forgotten, and if you plan on writing it down someplace, there is always a risk of someone else finding out. In biometrics, you can be sure that you won't lose or forget your fingerprint, face, or iris.
- Because everyone has their own unique biometrics, it's a much safer and hassle-free method.
- Your biometrics information remains constant over time.
- No one can steal or replicate your biometrics information.

[…]
Many banks have integrated biometric authentication features in their banking mobile applications such as 'fingerprint or touchID' to unlock a user's account and to approve safer transactions. Passwords are the root cause for more than 81 percent of data breaches, this proves that they are one of the biggest flaws in your personal as well as your company's security. Remember the time you opened your social media app to read your messages; After you've typed in your login credentials, only then did you notice that someone was looking over your shoulders. See how easy it is to hack your accounts."
So biometric doesn't require neither password and therefore nor password manager at all, and is safer and easier to implement than a session token.
However, in a real-life banking app, I would probably have used both biometric identification and a token session.

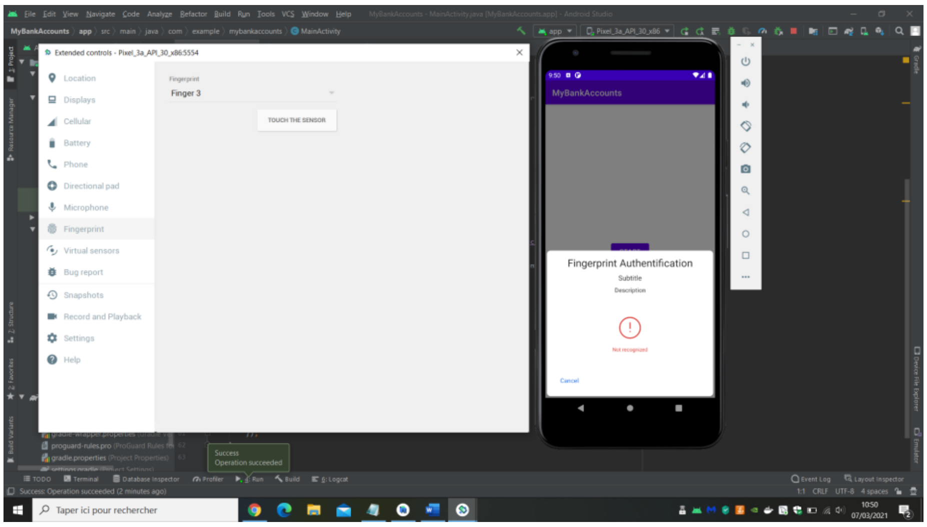
Implementation: To implement biometry in my application, I first added the permissions and made sure everything was compatible. Then I added a *biometricPrompt* object which will be activated when the user will click the *Start* button, asking for a fingerprint. If the authentication is successful, the next activity shows.

Result: On clicking the *Start* button in the home page of the application, the user needs to touch the fingerprint sensor.

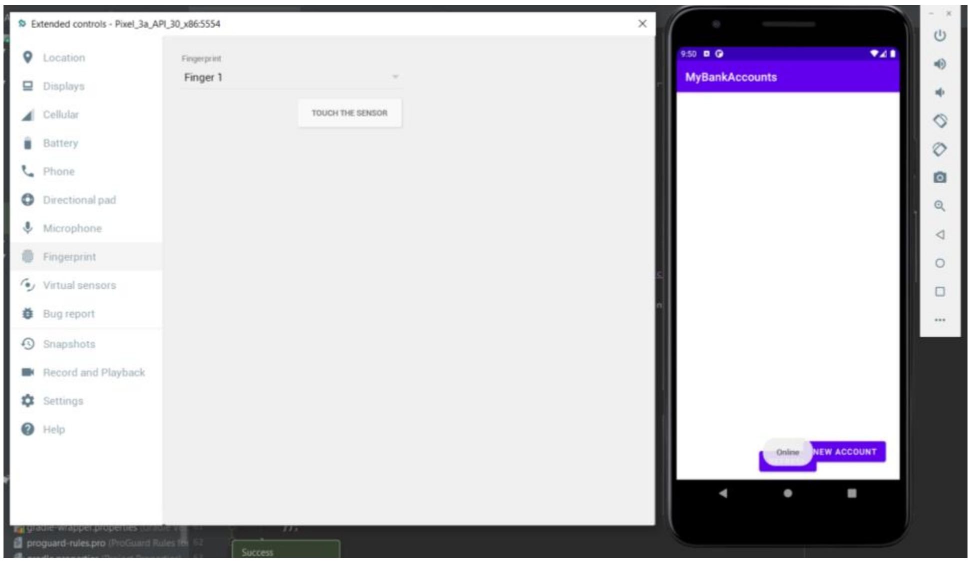




If the wrong fingerprint touches the sensor, an error message is displayed.



If the right fingerprint touches the sensor, the next activity is displayed.



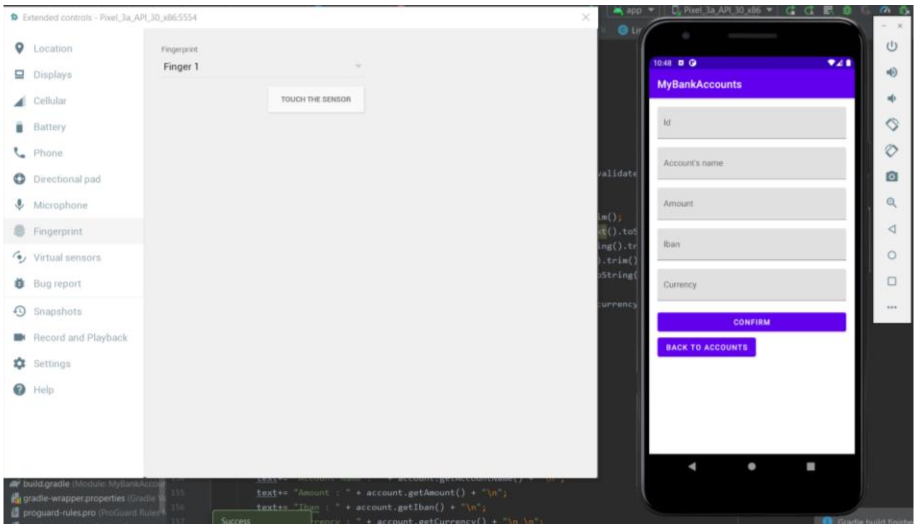
2) How do you securely save the user's data on your phone?

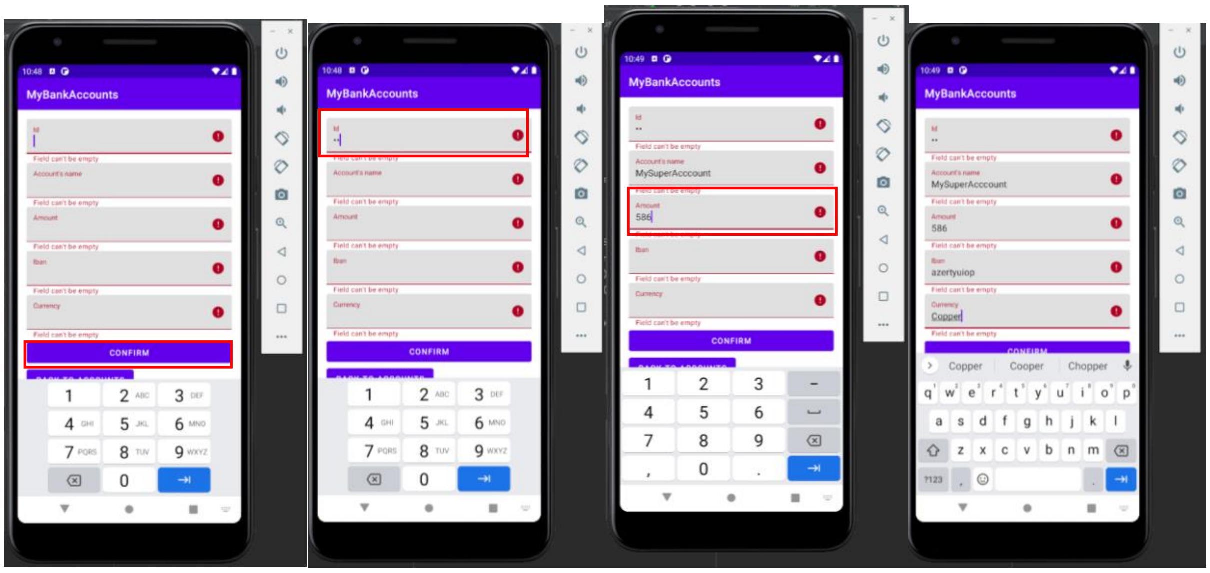
Thought process: After some research I chose to go for internal storage, which appeared to be the safest and easiest method according to the android developer's security tips[2].

Implementation: To write data on a file I used the *MODE_PRIVATE* attribute to ensure that only the user of the app was able to actually write in the file. To avoid injection as well, I have specified as much as possible the input type of the fields for creating an account (see below).

To ensure even more security, I could have used the Security Library to encrypt this local file. However, I had trouble implementing it, the *MasterKey* class was unavailable despite all my attempts to get this library. I was reassured to know[2] that Android itself already implements a first layer of protection for internal storage.

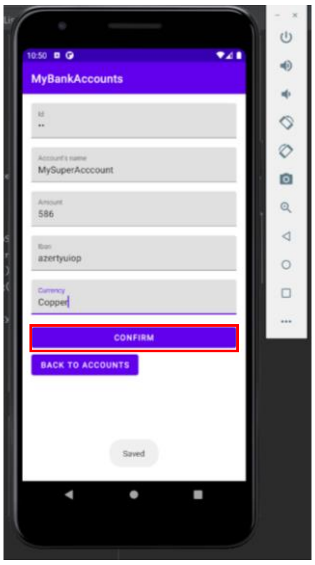
Result: So, as I said above, the input type for both the Id field and the Amount field must be numbers. Furthermore, if one of the fields, or all fields are blank, an error message is displayed.





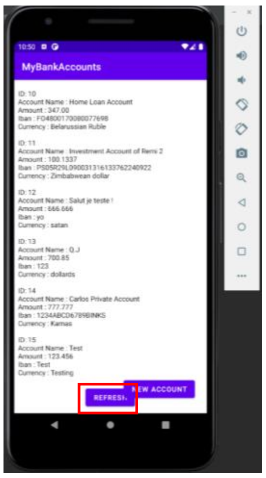
Id: --
Account's name: MySuperAccount
Amount: 586
Iban: azertyuiop
Currency: Copper

Now that all fields are filled, the user can click on the *Confirm* button to save the data into the file.



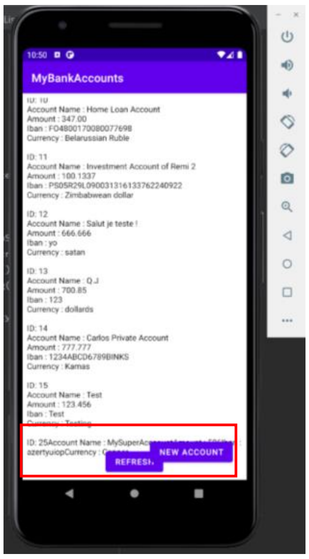
Id: --
Account's name: MySuperAccount
Amount: 586
Iban: azertyuiop
Currency: Copper

Now, the user hits the Back to Accounts button and the accounts are displayed on the screen. However, the account that had just been created is not displayed yet, one has to hit the Refresh button.



```
ID: 10
Account Name : Home Loan Account
Amount : 347.00
Iban : FO4800170080077698
Currency : Belarussian Ruble

ID: 11
Account Name : Investment Account of Remi 2
Amount : 100.1337
Iban : PS05R29L0900313161337622240922
Currency : Zimbabwean dollar

ID: 12
Account Name : Salut je teste !
Amount : 666.666
Iban : yo
Currency : satan

ID: 13
Account Name : Q.J
Amount : 700.85
Iban : 123
Currency : dollards

ID: 14
Account Name : Carlos Private Account
Amount : 777.777
Iban : 1234ABCD6789BINKS
Currency : Kamas

ID: 15
Account Name : Test
Amount : 123.456
Iban : Test
Currency : Testing
```

And now the new account is displayed on the screen.



ID: 10
Account Name : Home Loan Account
Amount : 347.00
Iban : FO4800170080077698
Currency : Belarussian Ruble

ID: 11
Account Name : Investment Account of Remi 2
Amount : 100.1337
Iban : PS05R29L090031316133762240922
Currency : Zimbabwean dollar

ID: 12
Account Name : Salut je teste !
Amount : 666.666
Iban : yo
Currency : satan

ID: 13
Account Name : Q.J
Amount : 700.85
Iban : 123
Currency : dollards

ID: 14
Account Name : Carlos Private Account
Amount : 777.777
Iban : 1234ABCD6789BINKS
Currency : Kamas

ID: 15
Account Name : Test
Amount : 123.456
Iban : Test
Currency : Testing

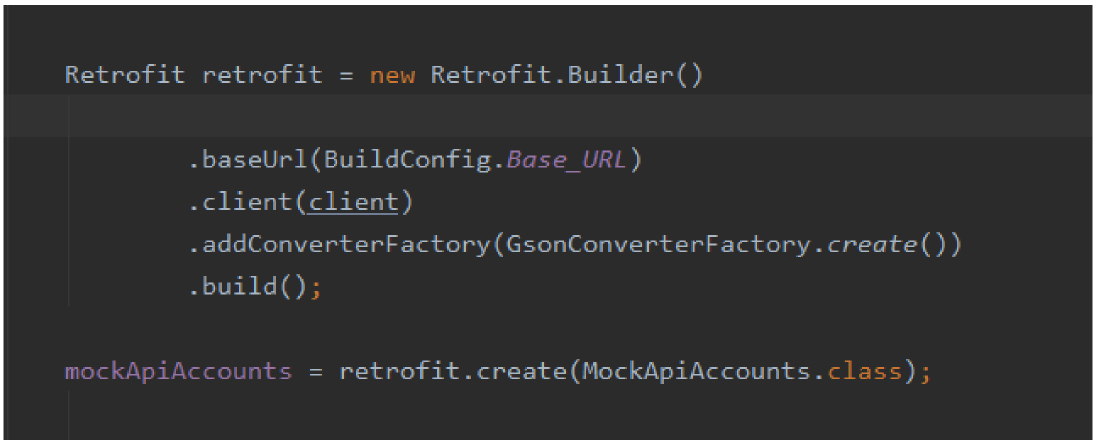
ID: 25Account Name : MySuperAc
azertyuiopCurrency : C

3) How did you hide the API URL?

<u>Thought process:</u> After some research, the best practice I found was to hide the API URL into an environmental variable in the *gradle.properties* file.

<u>Implementation:</u> So, I declared a variable call *WEBServiceBaseURL* and store the API's URL in it. Then I added a new *baseConfigType* so I could use my secret variable when building a Retrofit object later on.

<u>Result:</u> The URL is hidden in the code in the *Base_URL* variable.



```java
Retrofit retrofit = new Retrofit.Builder()

        .baseUrl(BuildConfig.Base_URL)
        .client(client)
        .addConverterFactory(GsonConverterFactory.create())
        .build();

mockApiAccounts = retrofit.create(MockApiAccounts.class);
```

Links:

1 : https://www.m2sys.com/blog/guest-blog-posts/biometrics-mobile-app-security/#:~:text=So%20whether%20you%20want%20to,more%20convenient%20and%20user%2Dfriendly.

2 : https://developer.android.com/training/articles/security-tips#StoringData