APIs and the production of Financial Technology Applications

Francisco Funes • Abbasi Bahareh • COMP499

Introduction

The mobile phone is an integral part of our everyday lives. The phone contains prized images of loved ones, easy access to our friends, and easy access to our employers and side hustles. Our financial lives as well, there are many applications to choose from to manage our finances, from first party bank applications provided by your institution of choice, or third parties such as PayPal, or Cash App. Then there is third party tools to make our finances easier, certainly Zelle rings a bell as the primary way to send a friend money for an Uber or dinner. Investing has been cracked open with applications such as Robinhood or Acorns. These companies possess large teams of developers that create these applications to sell to us, but what about the small startup that wishes to get into the field of application development for mobile, especifically the financial part of the industry? The question becomes:

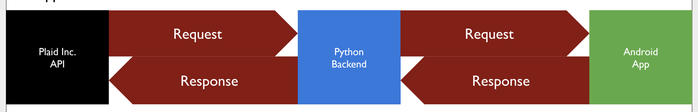
Methodology

The answer isn't as simple as it may seem, but it is possible thanks to companies who expose their Application Programming Interfaces as products. One such company, Plaid Inc. develops and maintains such an API that allows developers to create Financial Technology Applications by using it. This API was chosen for this project because Plaid offers a free developer Sandbox experience that mimics real world data, easy setup and installation. The API will be used to link user bank accounts to our application, and for each bank account the API is used to poll transaction data. And for sake of demonstration we show that this data can be saved locally and used to conduct basic data analysis and allow the user to filter transactions into an expense list.

The application itself will be an Android Application created using the Android Studio IDE, and the Kotlin programming language. Several dependencies are added to the project such as Volley, Cardview, Plaid, and Room to allow the app to do networking, data storage, and have more stylized visuals. [2,3]

The application also has a backend running on the free web hosting platform Heroku. The backend is provided by Plaid Inc. as part of their quickstart to aid the development process, this allows the developers to focus more on the product than the backend. [4]

The application flow is as follows:



The project is implemented bottom up, the android app being last. An account is made at www.plaid.com and the initial setup process is done to by navigating to the appropriate menus to enable the sandbox mode, get the secret keys for the python server, and tell the API that it will receive requests from an Android application. This is all done through a web page with clickable buttons and visual settings so it is a straightforward process. More details can be found on the QR code on the bottom right. [4,2]

The python backend is a reworked version of the Plaid Quickstart version provided on their GitHub page, the original can be found by scanning the QR code on the left, and this projects version can be found by scanning the QR code in the center.

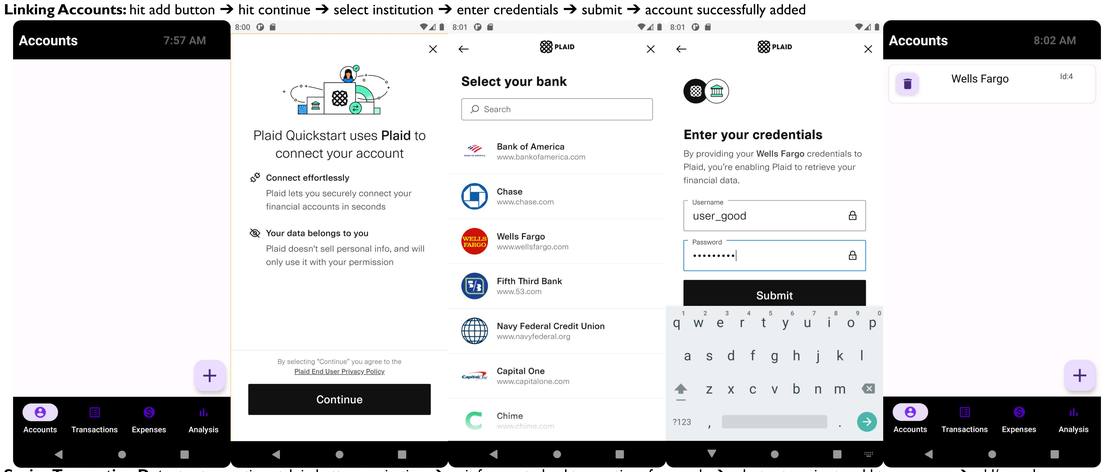
The Android application was constructed using the View ViewModel Model approach using the dependencies mentioned above, and results can be seen in the center focus of the poster.

[PLAID QS][5] [PROJECT] [PLAID][4]

Result

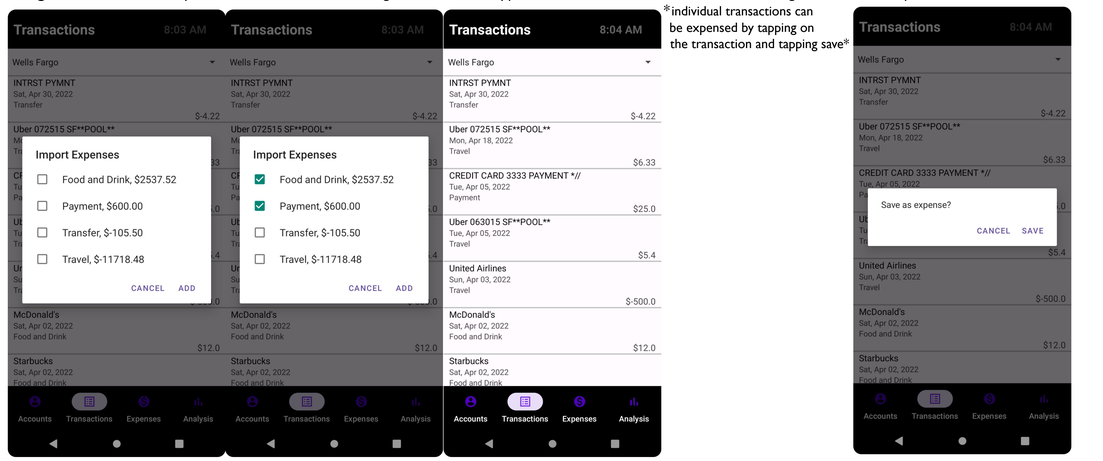
An android application was constructed that allows the user to link bank accounts, poll transaction data, add transactions to an expense list, and visualize their expenses. The functionality is powered in the backend by Python and the Plaid API while the visuals and user experience is rendered by the Android framework producing an example of a Financial Technology Application made at the startup level of knowledge.

Linking Accounts: hit add button ➔ hit continue ➔ select institution ➔ enter credentials ➔ submit ➔ account successfully added



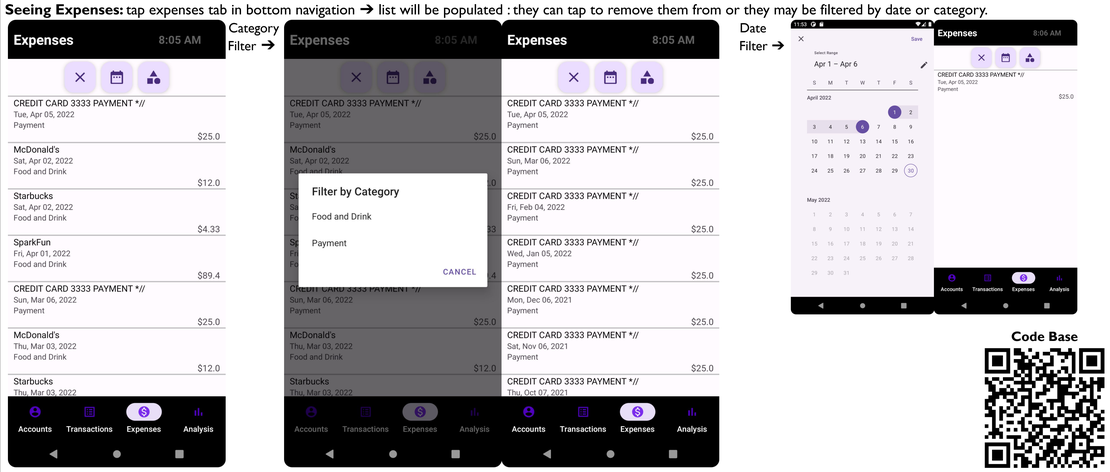
Seeing Transaction Data: tap transactions tab in bottom navigation ➔ wait for app to load transactions from web ➔ select categories to add to expenses ➔ add/cancel

individual transactions can be expensed by tapping on the transaction and tapping save



Seeing Expenses: tap expenses tab in bottom navigation ➔ list will be populated : they can tap to remove them from or they may be filtered by date or category.

Category Filter ➔ Date Filter ➔



Code Base

Conclusion

So can one set out to make a financial technology application on their own? Yes, is it hard? It depends. A developer that just started learning Java will not right off the bat be able to produce this type of application in a short time frame, there is prerequisite knowledge about SQL databases, Android development, and networking to name a few. However, that isn't to say it is not possible, all that is required is the basics in each of these concepts to start and the rest as they say is history. A lot of software is learned by doing after all.

This application was completed in 4 hour stints over the course of two semester, that is thirty weeks 16 hours a week for a total of 480 hours of work which includes coding, design, and learning concepts that were unknown. Condensed to a full time 8 hour day, that is 60 days of full work. In other words, for a lone developer, a product of this kind, a fully polished alpha version could stretch to a 6 month production period.

This small development time is due in part to the heavy lifting for user data being done for us by API providers like Plaid Inc. Coupled with abstraction layers provided by Google like Room to make SQL database work easier, and the Android framework providing an easy to use UI creation tool. It is fair to say that startups with teams of ten can easily produce these applications in mass, and as such it is not just PayPal out there anymore trying to make banking easier. [3,4]

Some things to keep in mind for those on a journey to make such products. Saving data locally as this project has done may not be as secure since the transaction data is saved on device, encryption can be done, but this adds another layer of work for the app to do which could affect performance on retrieval and saving. So it is suggested to instead set up a user account database on the backend and save data there using proper security tools, that way the application only ever has to retrieve data on launch and not do any reading or writing. This effectively would make the application a fancy web page, and the performance could be much better.

Another thing to avoid is re-writing the wheel. Avoid filtering data logically if you can use already optimized libraries to do the work. For instance, in this project expenses are saved as references to a transaction, meaning to retrieve the transactions we could logically find the references in the transaction list. The issue is this becomes slow with large amount of transactions. The issue can be solved easily by simply writing the appropriate SQL query that joins the two lists, and since the engine running SQL in the background has optimizations to do these searches, the performance is much better.

It would also help developers to keep in mind that even this project will become obsolete rather quickly. Developing an Android application natively in Kotlin on Android studio is great for companies that have established products that they support so it may be helpful to write them in their native tools and languages. However, with new tools like Flutter, another framework that uses DART as its programming language offers the developer the ability to write one app, that fits all screens, iOS or Android. For a company that means one less employee, and a possible faster development window. [1]

This work is less of a research project and more of a demonstration. The idea was to showcase the power of the current technologies at the developers disposal to create products in mass in an ever evolving landscape of software. This was shown in the creation of an Android application that allows user to do expense management by linking their bank accounts and filtering expenses from them. It was shown that not a lot of time is required to implement the application, and even a beginner with some effort can learn to utilize the tools available to produce such products to provide value to themselves or their employer.

Tools Used

References

[1] : https://docs.flutter.dev/#new-to-flutter
[2] : https://developer.android.com/docs
[3] : https://developer.android.com/training/data-storage/room
[4] : https://plaid.com/docs/quickstart/
[5] : https://github.com/plaid/quickstart