Revolut iOS Test Assignment

The task

The goal of this assignment is to allow you to demonstrate your programming skills and to allow us to understand your approach when developing a mobile application.

Your app should be production ready: In addition to assessing your code quality, we will be trying to break your app by testing for different usability issues that a real user might encounter.

We would like you to implement an exchange rates app.

The app must request and update currency rates every second using an API.

The API endpoint you should use is:

https://europe-west1-revolut-230009.cloudfunctions.net/revolut-ios

To obtain rates you should pass an array of currency code pairs as follows:

https://europe-west1-revolut-230009.cloudfunctions.net/revolut-ios?pairs=USDGBP&pairs=GBPUSD

Users should be able to:

- Add a new currency pair
- Remove a currency pair

The list of currency pairs should be preserved across app launches.

Code

- Please use the latest stable Xcode + Swift
- Deployment target is iOS 12
- Avoid use of third-party libraries such as Rx. We would prefer to see a vanilla approach so we can see you understand the underlying concepts. We do use third party libraries at Revolut of course, but for the purposes of this test, please avoid doing so
- Please ensure your app has good test coverage

Resources

You will find fonts, colours and everything else that may come in handy in Figma project:

https://www.figma.com/file/Vh2zTldjRFFqSk29AmZeVBWy/Exchange-%26-Rates

UI does not have to be exactly the same, the final implementation is up to you.

You can handle currency name, abbreviation and country flag the way that fits you.

Also, in a zip archive you will find the following:

- A video describing the app flow

- A list of currencies supported by the test assignment backend
- An icon

What we are looking for:

- Clarity, elegance, and maintainability of code
- Code consistency. Please refer to a community maintained style guide if you're unsure (eg: Ray Wenderlich: https://github.com/raywenderlich/swift-style-guide).
- A clear architecture and adherence to iOS design patterns
- Knowledge of the iOS human interface guidelines
- Appropriate application of relevant native iOS frameworks (Foundation, UIKit)
- Good code coverage with unit tests

What we don't want to see:

- Overengineering your code
- Lack of unit testing

When you think you're done

We expect to be able to run your code without fuss - so if there are any specifics that are needed to get your code running, please commit them to git repository if possible.

Provide your test app as a zip archive and don't post it on public code sharing services (github, bitbucket).