Clerkie Android Coding Challenge

Congratulations on making it this far in our interview process. We're excited about the opportunity to have you join the team and are very appreciative of you taking the time to complete this coding challenge.

The purpose of this coding exercise is to evaluate and get to know you along 4 dimensions:

1. Android Coding Experience (e.g., how good and familiar you are with Android and relevant libraries)
2. UX (e.g., your ability and passion for beautiful front-end / user experiences)
3. Self-proficiency (e.g., how much you can get done on your own)
4. Collaboration (e.g., your willingness and courage to ask for help)

But, please note that you do not have to excel in all 4 areas to get the job. These dimensions do, however, give us some good data points as to how well you enjoy and excel at Android development.

Deadline

You will have until **Tuesday (8/28)** before midnight (e.g., by 11:59 PM PST) to complete this challenge. Please take your time and don't hesitate to reach out (via email or text message) if you have any questions or concerns.

Submission requirements

Please email your final work to talent@clerkie.io including a link to your github repo (NOTE: please ensure that all dependencies/libraries are included in the repo). In addition, please include screenshots of all relevant screens (either within Github or in a separate document attached to your email submission).

Coding Challenge Part 1 – Creating a Log-in flow

As with any financial app, security is critical. So, it's only fitting for your app to require a **login to access your private financial information**. While we will not require any crazy technology implementation (e.g., facial recognition, iris scan, etc) or even any back-end implementation, we do have 2 very basic requirements:

1. Credentials must be a **Username + Password combination**
    - Username can only be 1) a valid **US Phone** number or 2) a valid **email address**
2. If the user **already has an account**:
    - Valid credentials → Access to the app
    - Invalid credentials → UI prompt indicating that the entered credentials are not valid

3. If the user **already does not have an account** → UI prompt indicating account doesn't exist
4. **Add a signup UX flow and logic to your app** (e.g., if user doesn't have an account, they can create one). The signup screen should be **presented with the following animation →**

http://css3transition.com/wp-content/uploads/2017/08/material-design-login-form-animation.gif

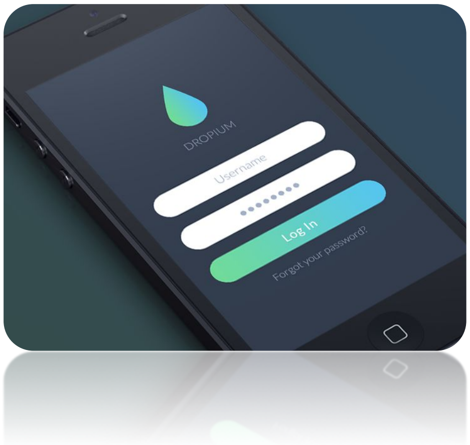
To keep things simple, you can program the app credentials on the App, and submit a list of credential combinations that will give us access to the app.

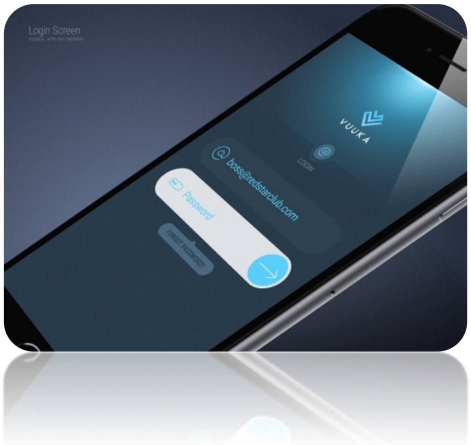
Some of us like to go above and beyond or may just want to commit more time than others to refine their apps. For those kindred spirits, we've laid out a few extra credit ideas that you can choose to implement. **Remember, these extra credits are not required! You can choose to do them all or none of them** 😊

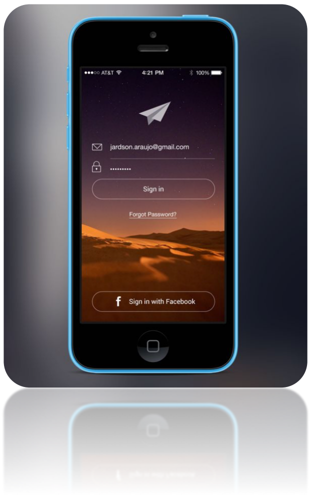
Extra credit #1: Add a forgot/reset password UX flow or logic to your app (e.g., if user entered the wrong credentials or forgot their password, then they can reset their credentials)

Extra credit #2: Connect your app to a back-end API service that can store credentials in a database (e.g., without having to store the credentials locally on the app)

P.S. Make your login screen look good! (Added some images from dribble for inspiration)







Coding Challenge Part 2 – Creating a Chat interface

Conversational interfaces are all around us these days. And, today, these interfaces are by far the easiest way to interact with any AI agent (at least until we, as a society, crack brain-computer-interfaces).

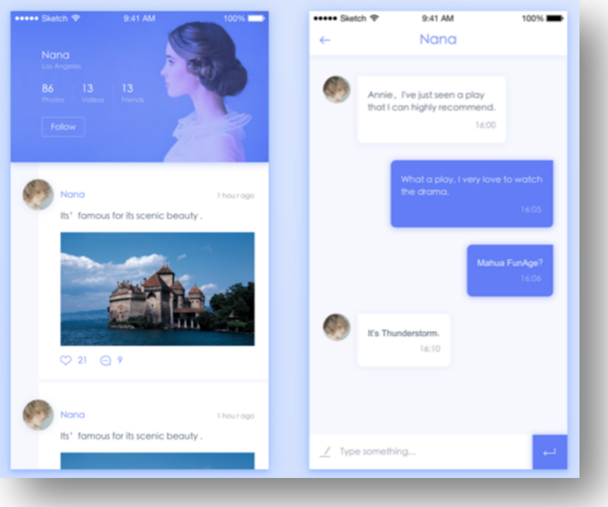
Your next challenge is to create a chat interface for your app. This will be your home screen (e.g., where the user will land after logging in every time). The chat interface is simple and has 6 key requirements:

1. **Keyboard will show up and hide** based on the user's behavior:
    - User wants to start typing (clicks on message box) → keyboard will show up
    - User wants to dismiss keyboard (e.g., clicks outside message box) → keyboard will hide
2. Use **dynamic height of the chat message box** (e.g., where user enters the message he wants to send)
3. User **can copy / paste text** from the chat log into the message box
4. Your **chatroom must be interactive.** That means that when we enter and send text (during testing), we should get a response back. Please note the response doesn't need to be logical and doesn't need to come from a back-end chat service (e.g., you can code an "if-else" or a "random sentence generator" logic locally on the front-end!
5. **Allow users to post videos and photos** in chat using some basic **animation** like the following:
    - https://cdn-images-1.medium.com/max/1600/1*1jF2JulWx9IiK7Kx5PNtSg.gif
    - https://cdn-images-1.medium.com/max/1600/1*UZt78KibwkQUxqjCjxQKpA.gif
6. **Add a chat shortcut button** with various shortcut options and **some basic animations** (e.g., you could add the upload video/photo button here + some good autocomplete financial shortcuts such as: transfer money, borrow money, deposit check… so that the user doesn't always have to type everything) → Here's an easy animation example:
    - https://cdn.dribbble.com/users/226618/screenshots/2340386/shoping_list.gif

For this portion of the challenge, we will only lay out 1 extra credit idea below – which is optional. Feel free to spice it up based on your preferences as well. Enjoy!

Extra credit #1: Connect your chatroom to some back-end of your choice using SocketIO or any tool of your liking (e.g., you could code up a basic if-else logic on the BE that will spit out simple sentences based on the user's input – no NLP or AI coding is required)

P.S. Make sure your chat screen look good! (here are some images from dribble for inspiration)



Its' famous for its scenic beauty .

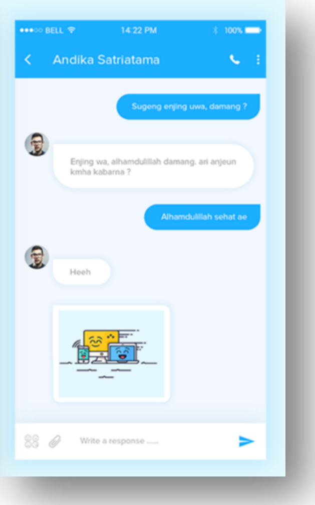
Its' famous for its scenic beauty .

Annie, I've just seen a play that I can highly recommend.
16:00

What a play, I very love to watch the drama.
16:05

Mahua FunAge?
16:06

It's Thunderstorm.
16:10

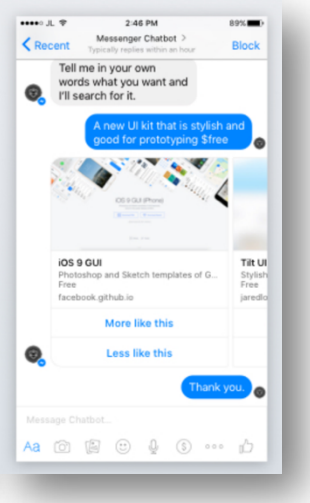


Sugeng enjing uwa, damang ?

Enjing wa, alhamdulillah damang. ari anjeun kmha kabarna ?

Alhamdulillah sehat ae

Heeh



Tell me in your own words what you want and I'll search for it.

A new UI kit that is stylish and good for prototyping $free

iOS 9 GUI
Photoshop and Sketch templates of G...
Free
facebook.github.io

Tilt UI
Stylish
Free
jaredlo

Thank you.

Coding Challenge Part 3 – Creating an analytics dashboard

Being in the FinTech means that you will have to communicate a lot of numbers and financial metrics to our users in a very digestible way. This is why being able to manage and showcase charts is very important.

Your next challenge is to create a simple analytics dashboard for your app. This will be the 2nd screen for your app and will therefore require a navigation bar or menu to allow the user to go back and forth (e.g., between the home screen and the analytics dashboard).

You can use any module of your choice to create the charts – one commonly used module among developers is the the MPAndroidChart chart module for Android by Phil Jay which can be found here: https://github.com/PhilJay/MPAndroidChart

The analytics dashboard only has 2 requirements:

1. As previously mentioned, your app will need a **navigation bar or menu** that allows the user to go back and forth between the 2 screens of your app
2. Your analytics **dashboard must be able to display 5 variety of charts:**
    - **Vertical bar chart**
    - **Horizontal bar chart**
    - **Single line chart**
    - **Duo line chart** (e.g., chart with 2 lines)
    - **Pie chart**

There are only 3 extra credit ideas that come to mind for this screen – as always, all of which are optional. Feel free to again add your personal touch to it. Enjoy!

Extra credit #1: Connect your analytics dashboard to a back-end API service (e.g., this would enable you to show different data sets for different users)

Extra credit #2: Make your charts look good! (e.g., add some view animations or other visual effects, add some color)

Extra credit #3: Add some popover tutorials to your charts (e.g., some popover with information detailing what the numbers or the chart means… the language/content of the explanation doesn’t have to make sense… the goal is to demonstrate that you can code nice tutorials/popovers)