How to run/view

Attached are images, containing screenshots on various devices, and a video of myself interacting with the app on a simulator.

To run on a simulator, run 'react-native run-ios' or 'react-native run-android' in terminal. You will need to setup Xcode and Android studio to run, the steps can be found at this link: https://facebook.github.io/react-native/docs/running-on-device.

Design

I wanted to design the app to be as device agnostic as possible. To accomplish this, I made the fonts, margins, and paddings based on either the height or width of the device. This allows for components to scale up or down based on the screen size and device type. I also attempted to make as many of the margins equal to the same value, which is something Abridge's Chief Product Officer always emphasizes.

I wanted the UI to be as simple as possible, so I decided that there should be 4 screens: the first to display all tv shows, second to display all seasons for a tv show, third to display all episodes in a season, and a fourth to display information about the tv show. This paradigm allows for components to be reused and rendered using a FlatList component while also making it simple to add additional shows or functionality.

Episode.js is a simple component, which has the episode name as the title and displays all information, as requested in the doc, in an easy to understand format.

Code

App.js, which is the entry point for the app, uses react-navigation to handle all routing between components. I created a stack navigator, with *MainScreen.js* as the initial route, and added some code to handle the appearance of the navigator's header.

The main code structure is fairly simple. *MainScreen.js*, *TelevisionScreen.js* and *Season.js* all follow a similar coding structure of a title display and a FlatList, which has routes to further information. *MainScreen* contains the tv shows hardcoded in, while *TelevisionScreen* and *Season* have values passed in as props and a *processFetchString() method,* which calls the OMDB API to get further information on seasons and episodes.

Unit Tests

I have snapshots tests for all components. Since the only function I call is *processFetchString()*, which essentially is a wrapper for the OMDB API that sets values to state, I decided it was not necessary to write a unit test for it, though I do handle for errors.

Next Steps

If i had more time, I would have explored breaking out the FlatList and title from *MainScreen.js*, *TelevisionScreen.js* and *Season.js* into a separate file. I would then pass the *onPress()* method and Title in as props, thus eliminating the same code being written multiple times. I also would have broken out the OMDB API into a separate *API.js* file, allowing for the search parameters to

be passed in as inputs, and returning the JSON output. Then, the components above could simple use hooks to set those values to state, which would be used in *render()*.