Testing Exercise

React / React Native application

creatix

We Innovate & Deliver Digital Products

Your goal is to create an application which will fetch remote data, persist data locally and sync data among screens. Created application needs to be composed of three screens. You can choose a platform to be used. It can be web app written in React or iOS/Android app written with help of React Native. Please choose only one app if not advised otherwise.

Hints and general requirements

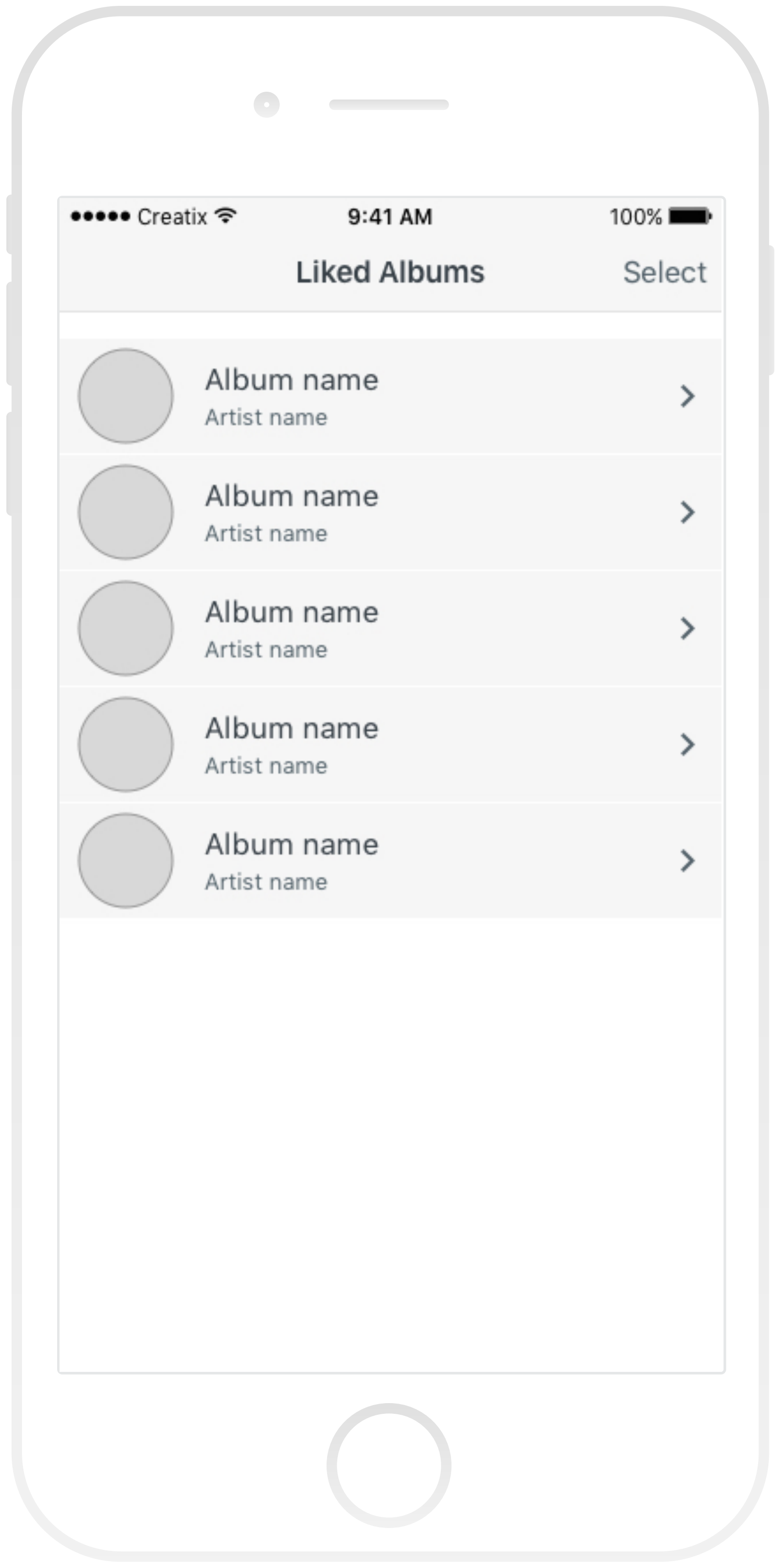
- **JSON** data will be used.
- For web app use **react-router** for navigation.
- For iOS/Android app use **react-navigation** for navigation.
- Create as many reusable "dummy" components as possible.
- App should support portrait and landscape mode.
- App should look good, but does not require "fancy" styling. Keep it clean and simple, use screens as a visual reference. No 3rd party styling packages are necessary.
- Use **axios** for data fetching.
- For async flow it is reccomended to use **redux-saga**. Not mandatory.
- App should handle network or request errors.
  It's okay to use alerts to inform about such errors.
- If using external state management, use **redux**, internal state otherwise.
- Code should not generate any warnings (console warnings).
- Code should not generate any linting warning. We suggest using **eslint-config-react-app.**
- Unified coding style. We suggest to use **prettier** to keep it unified.

Screen #1: liked music albums

This screen displays as a list of liked music albums.

Requirements

- The list of liked albums is persisted locally with no need to sync with a remote server. It is only needed to store ID's of liked albums. Other related data is pulled from the remote server. To load data it's necessary to fetch from: **https://itunes.apple.com/lookup?id={{album 1 id}},{{alubm 2 id}}**
- Tapping on a list item navigates to a screen with album detail
- "Select" button in a navigation bar leads to Screen #2 - Artist Lookup



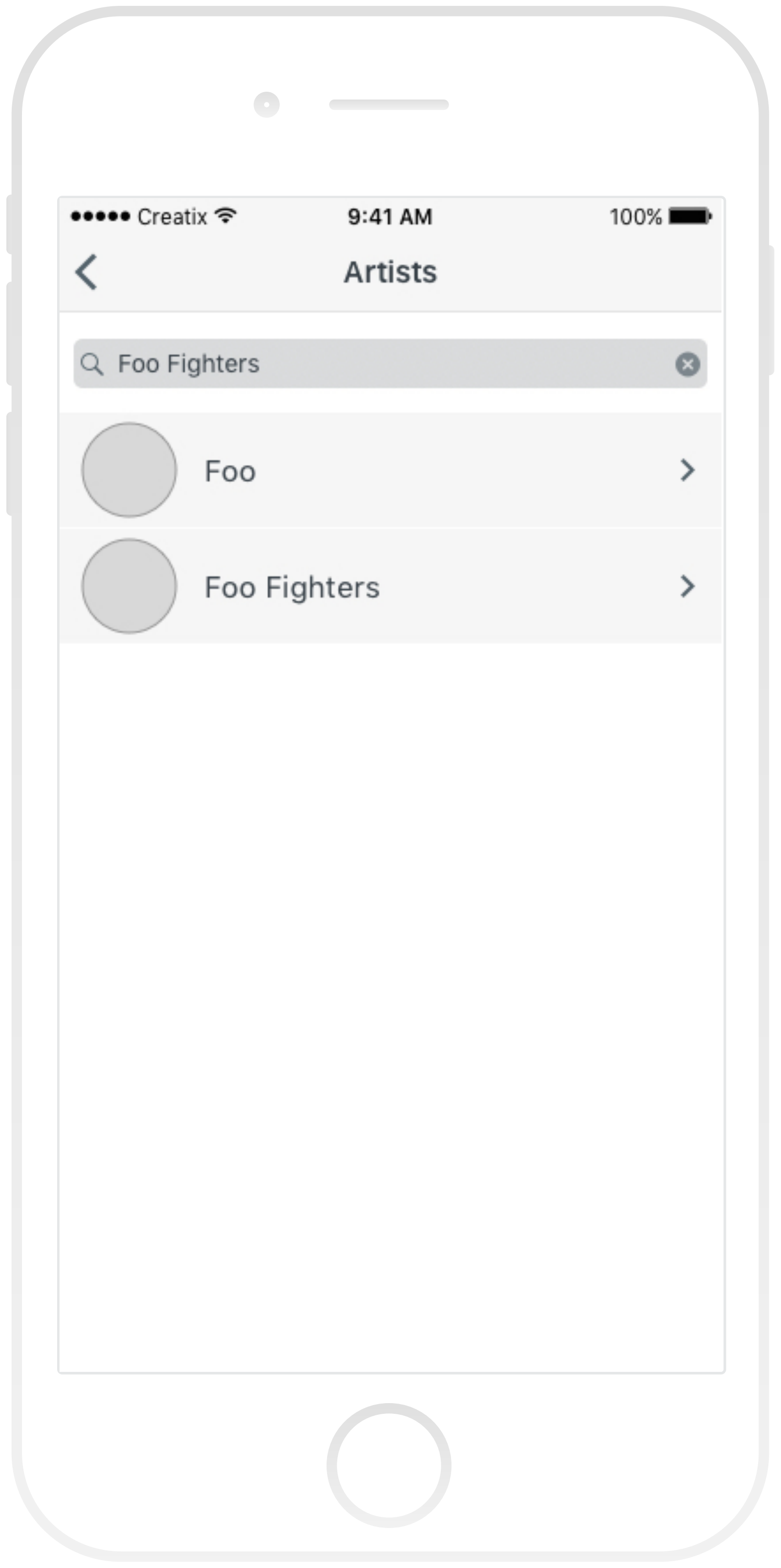
creatix

Screen #2: Artist lookup

This screen contains the artists sorted by a name.

Requirements

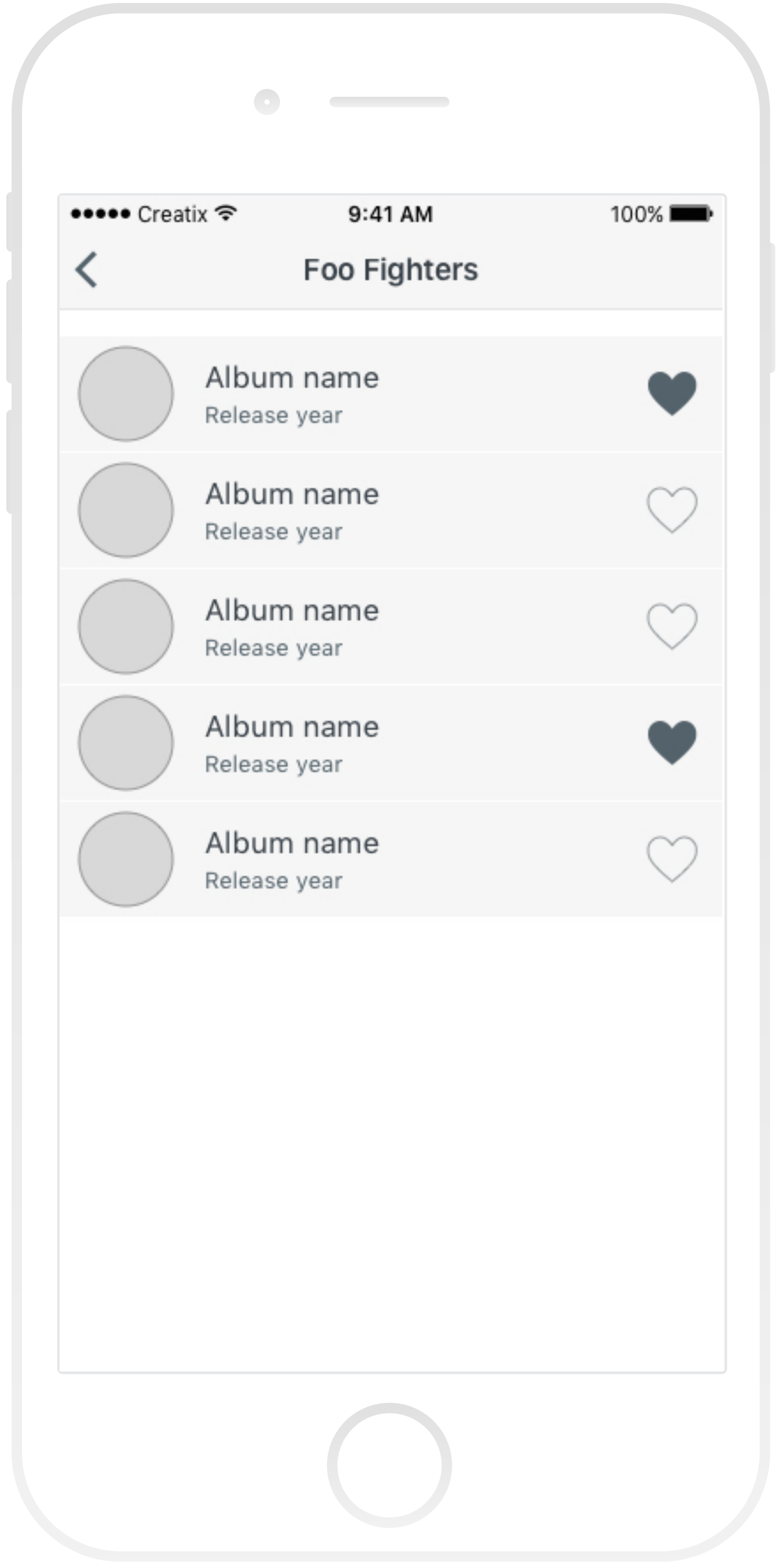
- Create a search field triggering network request on change.
  To load data it's necessary to fetch from:
  https://itunes.apple.com/search?term={{search term}} &entity=musicArtist
- It is needed to render the list of items from data received from the remote server
- Tapping on the list item leads to Screen #3 - Album listing



creatix

Screen #3: Album listing

This screen displays a list of music albums and allows a user to like the albums.



Requirements

- It is required to pass artist's ID and name from Screen #2 to this screen
- Display the artist name in the navigation bar
- Fetch the music albums by given artist and render them
  To load data it's necessary to fetch from:
  https://itunes.apple.com/lookup?id={{artist id}}&entity=album
- Tapping on a heart icon changes a status to like or unlike for a given album
- Any changes have to sync with the data on Screen #1
  (list on Screen #1 needs to show the most recent data when going back)

creatix

How will this challenge be evaluated?

A working solution is a must. It is not required to add extra functions that are not listed in this exercise. It is more important to focus on the approach and execution itself. Keep your code neat because we believe it is essential to be part of a team with an ability to cooperate efficiently.

Deliverables / submission

- Javascipt source code in the case of web application.
- Javascript and native code for React Native application.
- Reasonably documented build process.
- Make sure your source code is on is on Bitbucket or GitHub..
- Along with suggested 3rd party packages, use other 3rd party packages as you see fit, however make sure you document them and explain WHY you used them.
- Please annotate with comments as you see fit.
- Explain your approach and mention any research notes, articles and/or algorithms you have used (if possible, provide links or attach the original material)