React With Redux Certification Training

COURSE OUTLINE MODULE 09

1. Introduction to Web Development and React
2. Components and Styling the Application Layout
3. Handling Navigation with Routes
4. React State Management using Redux
5. Asynchronous Programming with Saga Middleware
6. React Hooks
7. Fetching Data using GraphQL
8. React Application Testing and Deployment
9. Introduction to React Native
10. Building React Native Applications with APIs

Topics

Following are the topics covered in this module:

- Native Applications
- React Native
- React Native Elements
- Expo CLI
- Build a shopping cart mobile application using React Native
- React Native installation and setup
- Working with Styles and Layout

Objectives

After completion of this module you should be able to:

- Learn about Native applications
- Understand React Native Technology
- Write React Native applications using Native elements
- Setup React Native using Expo CLI
- Build a Mobile application using React Native

Native Applications

What Is Native Application?

A ***native application*** is a software program that is developed to be used on a particular platform or device.

Native Applications

01 These devices make use of two types of OS platforms ***Apple's iOS*** and ***Google's Android***

02 Native apps can provide optimized performance and take advantage of the latest technology, such as a ***GPS***

03 They also have access to phone's various devices, such as its ***camera*** and ***address book***. In addition, users can use some apps without an ***internet connection***

04 It provides ***fast performance*** and a ***high degree of reliability***

React Native

What Is React Native?

React Native is a technology that allows us to build real native applications with the help of JavaScript and React Libraries which we can ship to either apple app store or google play store.

- React Native provides you a collection of ***special components***. These components are compiled to ***native widgets*** of iOS and Android
- It provides you access to ***native platform APIs***. example: ***Device camera***
- It provides you certain ***tools*** to connect JavaScript code to Native platform code
- ***React Native features*** are combined with ***React.js*** to compile a JavaScript code to a ***Real Native application*** which can later be shifted to app store and play store

Android applications are developed using JAVA and iOS applications using Swift. Do I need to learn even both of these languages?
No, you only need JavaScript and React Native

React Native Elements

React Native Elements

Here instead of HTML elements you will be writing your application code using Native elements like ***View and Text.***

Example: React +React Native application code

```
const App = props =>{
  return{
  <View>
    <Text> Welcome to Edureka</Text>
  <View>
  };
  }
```

Expo CLI

Expo CLI

Expo CLI is a command line app that is the main interface between a developer and Expo tools.

It is used for variety of tasks, such as:

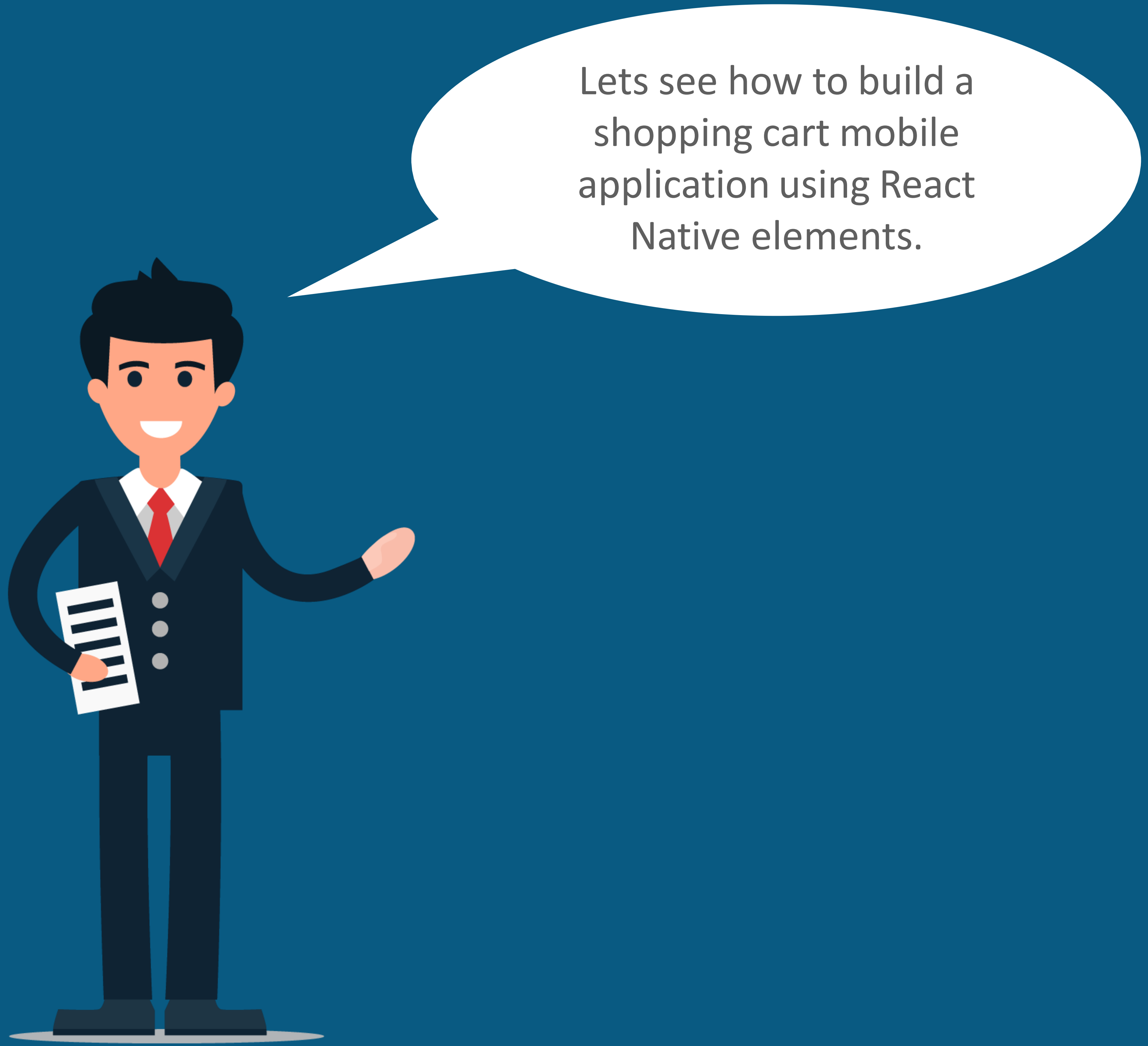
Creating ***new projects***

Developing your app (running the project server, viewing logs, opening your app in a simulator)

Publishing your app, JavaScript and other assets plus managing their release

Building binaries (apk and ipa files) to be uploaded to the App Store and Play Store

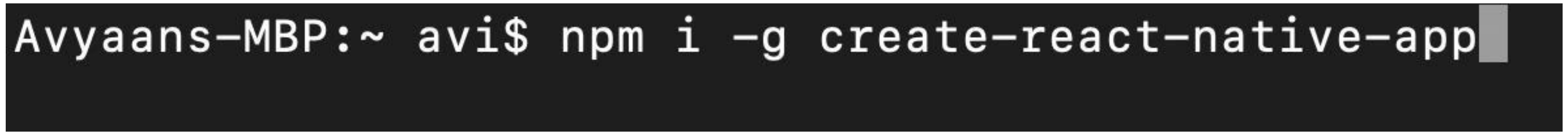
Managing Apple Credentials and Google KeyStore


Lets see how to build a shopping cart mobile application using React Native elements.

Demo: Shopping Cart Application

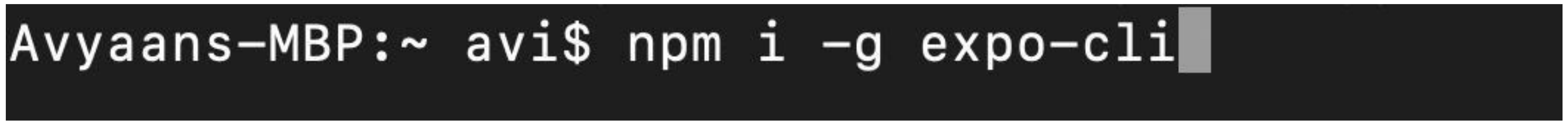
Demo: Setup For Building Mobile Application

Install new CLI for React Native, so using ‘***create-react-native-app***’ create the seed for building native app.



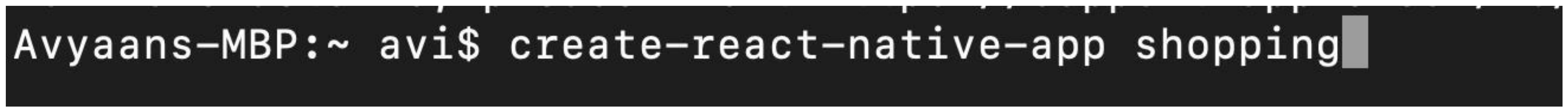
```
Avyaans-MBP:~ avi$ npm i -g create-react-native-app
```

We need a simulator that helps to compile and build the mobile application. So install ***expo-cli***.



```
Avyaans-MBP:~ avi$ npm i -g expo-cli
```

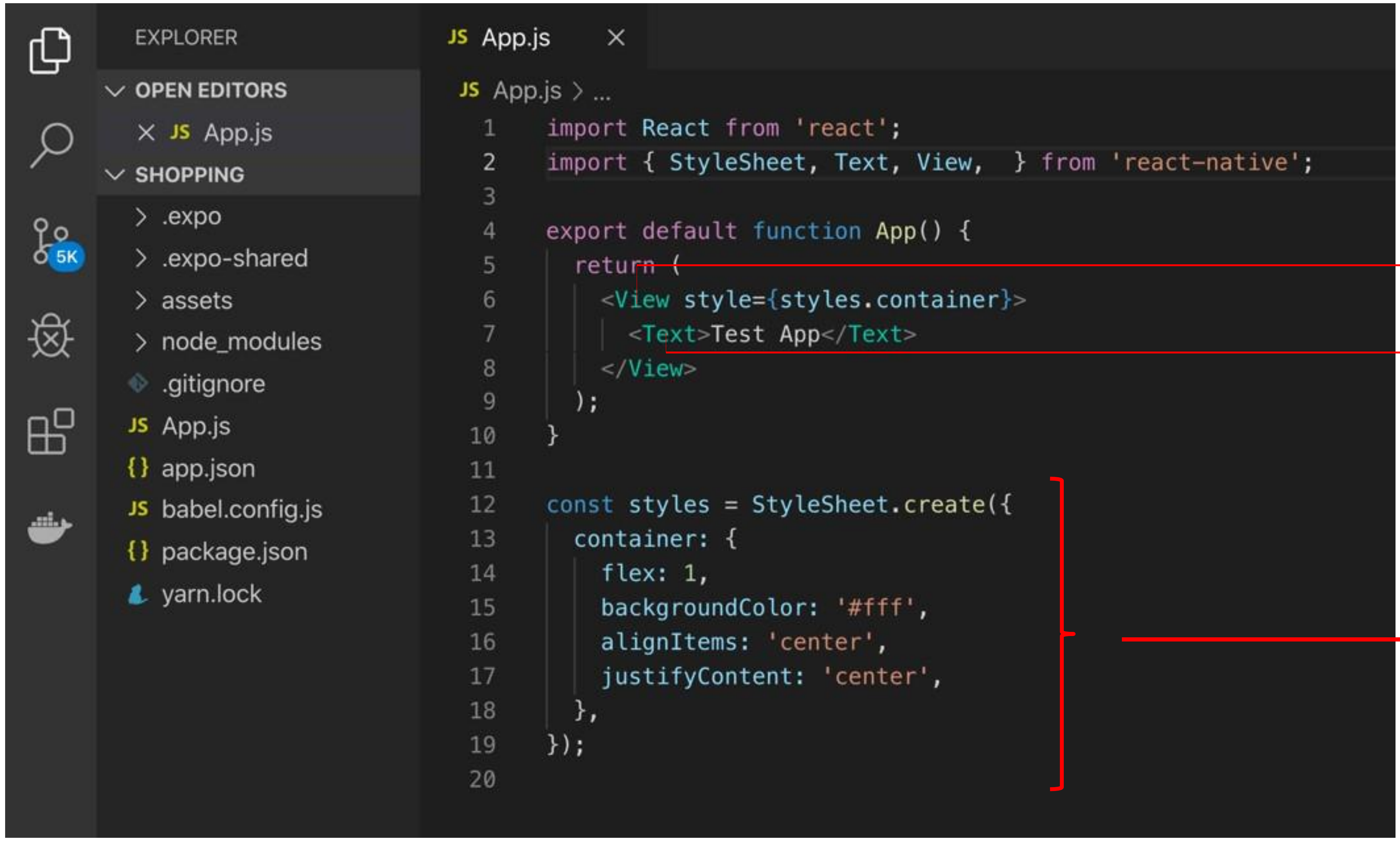
Now to start our application, build a new app within the seed using: ***create-react-native <appname>***



```
Avyaans-MBP:~ avi$ create-react-native-app shopping
```

Demo: App.js

In this seed we will get the basic structure of app folder over which we will start our development. Add the below snippet to App.js file. Application will bootstrap from App.js file.



```
import React from 'react';
import { StyleSheet, Text, View,  } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text>Test App</Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

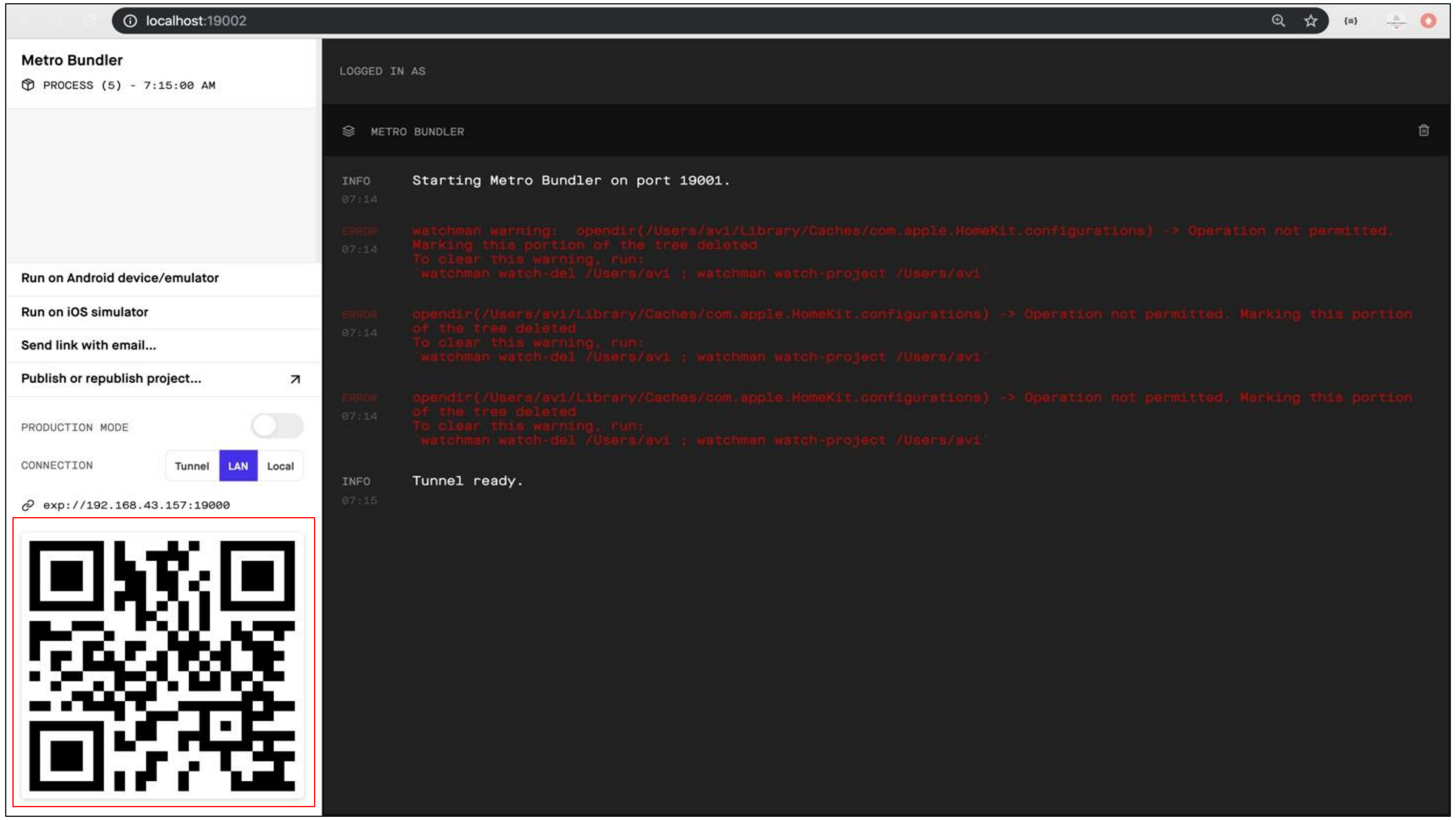
React Native element used to compile the code

React Native element used to display the text

Inline CSS

Demo: Run The Application

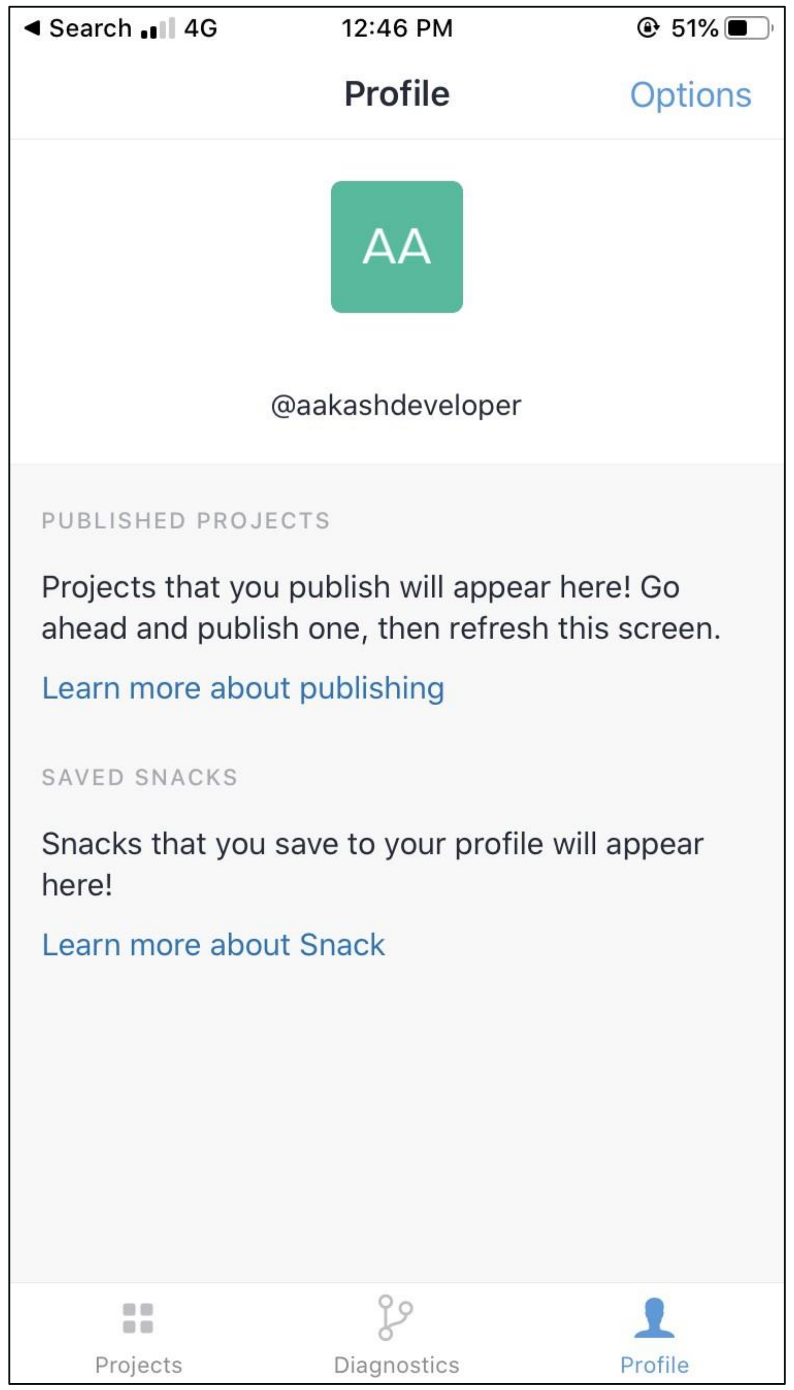
Run your application using ***npm start,*** this will open the panel of React Native on the browser. The panal contains a QR code which is used to start the application in mobile phone.



```
INFO   Starting Metro Bundler on port 19001.
07:14

ERROR  watchman warning:  opendir(/Users/avi/Library/Caches/com.apple.HomeKit.configurations) -> Operation not permitted.
07:14  Marking this portion of the tree deleted
       To clear this warning, run:
       `watchman watch-del /Users/avi ; watchman watch-project /Users/avi`

ERROR  opendir(/Users/avi/Library/Caches/com.apple.HomeKit.configurations) -> Operation not permitted. Marking this portion
07:14  of the tree deleted
       To clear this warning, run:
       `watchman watch-del /Users/avi ; watchman watch-project /Users/avi`

ERROR  opendir(/Users/avi/Library/Caches/com.apple.HomeKit.configurations) -> Operation not permitted. Marking this portion
07:14  of the tree deleted
       To clear this warning, run:
       `watchman watch-del /Users/avi ; watchman watch-project /Users/avi`

INFO   Tunnel ready.
07:15
```

Demo: Install Expo App

Install expo app, here we will see the output of mobile application. On installation the main page of expo app displays as below:


◀ Search 4G 12:46 PM 51%

Profile Options

AA

@aakashdeveloper

PUBLISHED PROJECTS

Projects that you publish will appear here! Go ahead and publish one, then refresh this screen.

Learn more about publishing

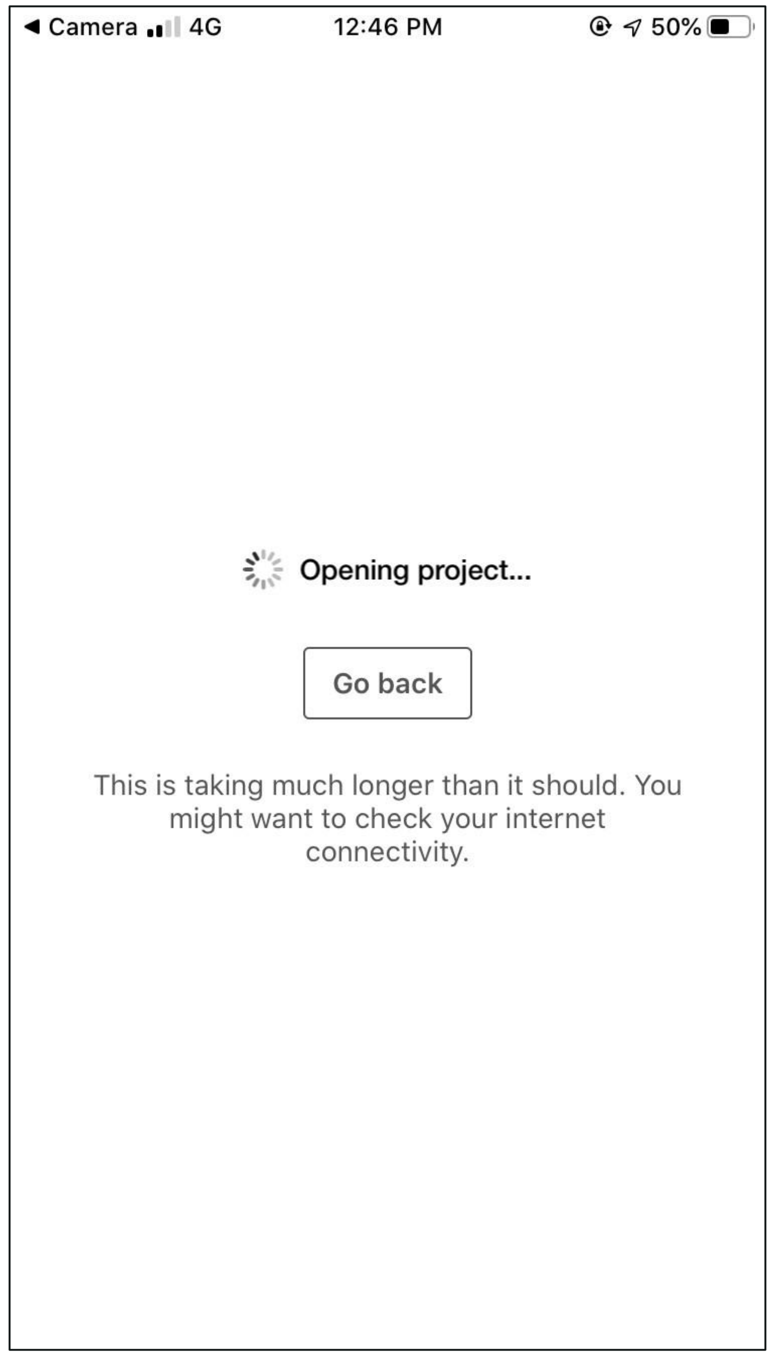
SAVED SNACKS

Snacks that you save to your profile will appear here!

Learn more about Snack

Demo: Setup To Start Application In Mobile (contd.)

Now we have to scan the QR code from the panel which will actually start booting the app in the mobile phone. After scanning you can see "***Opening project***" on screen. Make sure your laptop and mobile are connected to same network.

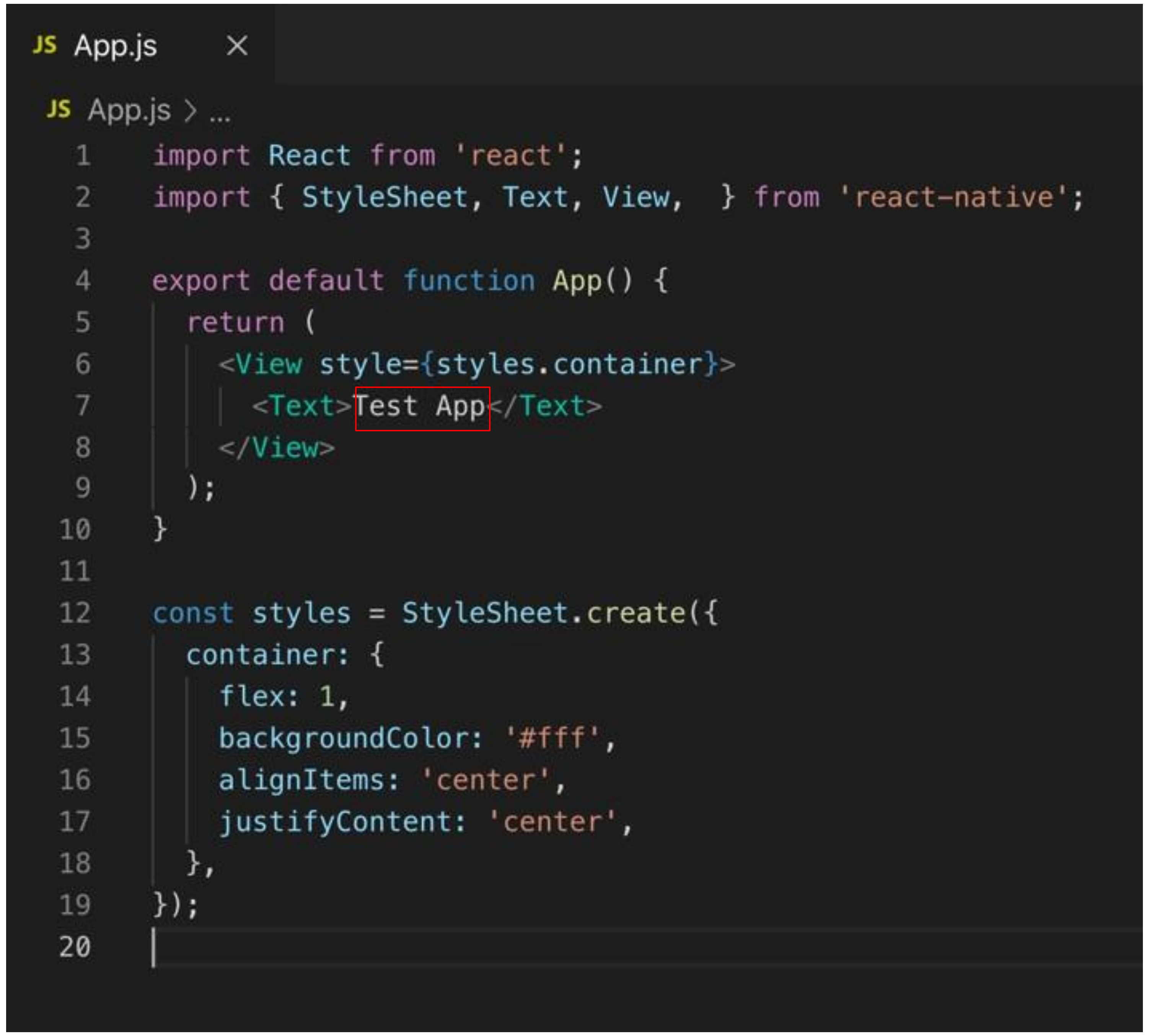


Opening project...

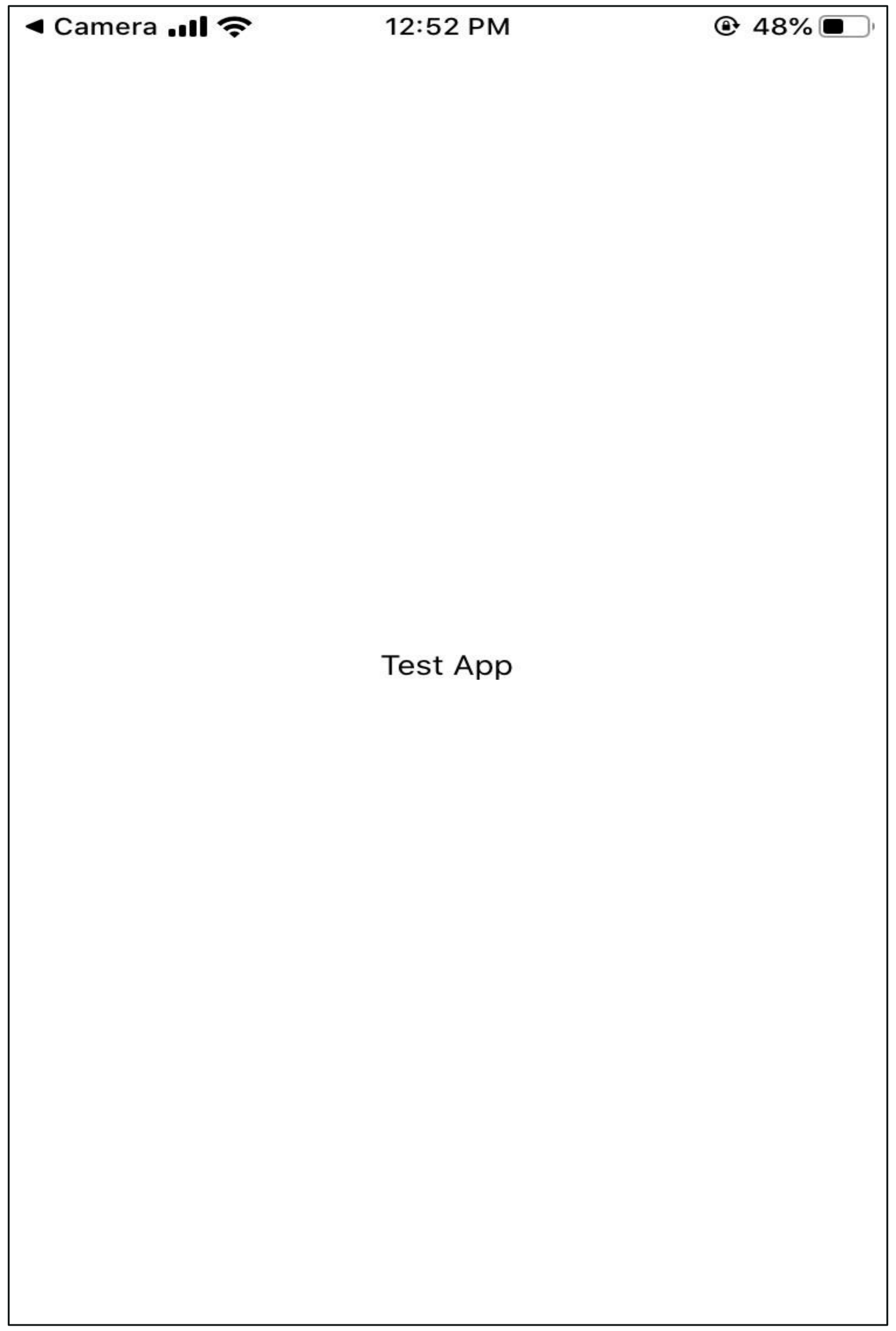
This is taking much longer than it should. You might want to check your internet connectivity.

Demo: Implement Code Changes

Now in App.js file you can edit the data in ***text*** section and the mobile app should display the changes.

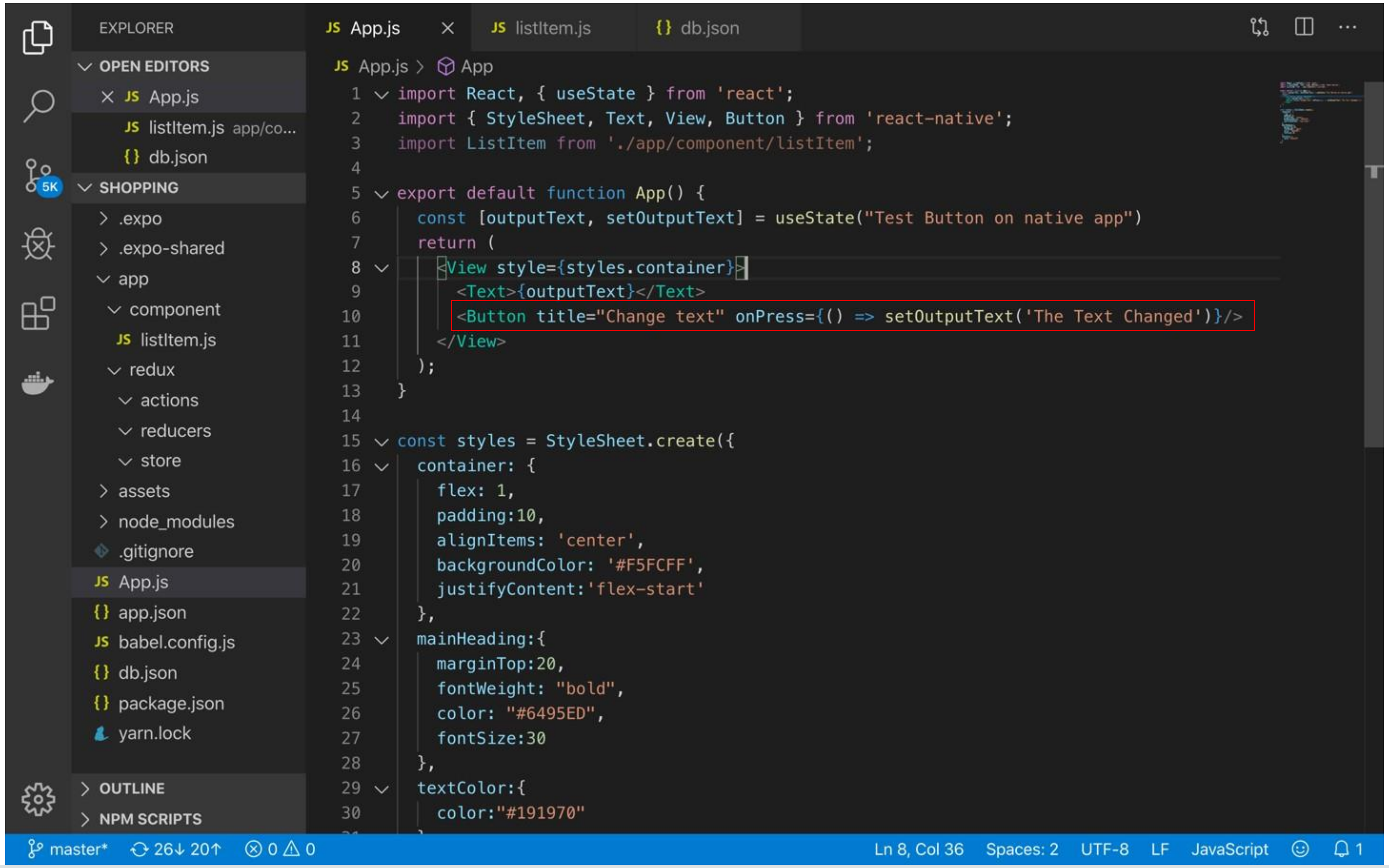


```
import React from 'react';
import { StyleSheet, Text, View,  } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text>Test App</Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```



Demo: Add Click Button To Change The Data

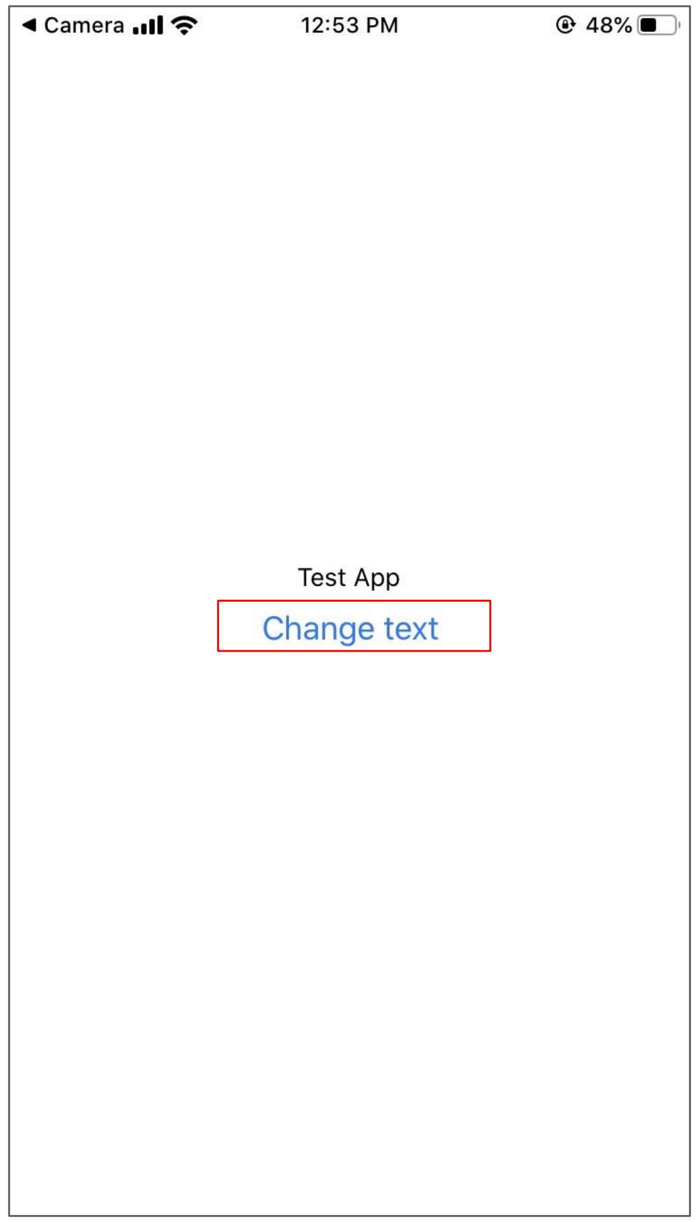
Add a button and make use of React hooks to manage the state change. By this On click of button we will change the text on the screen.



```
import React, { useState } from 'react';
import { StyleSheet, Text, View, Button } from 'react-native';
import ListItem from './app/component/listItem';

export default function App() {
  const [outputText, setOutputText] = useState("Test Button on native app")
  return (
    <View style={styles.container}>
      <Text>{outputText}</Text>
      <Button title="Change text" onPress={() => setOutputText('The Text Changed')}/>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding:10,
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
    justifyContent:'flex-start'
  },
  mainHeading:{
    marginTop:20,
    fontWeight: "bold",
    color: "#6495ED",
    fontSize:30
  },
  textColor:{
    color:"#191970"
```

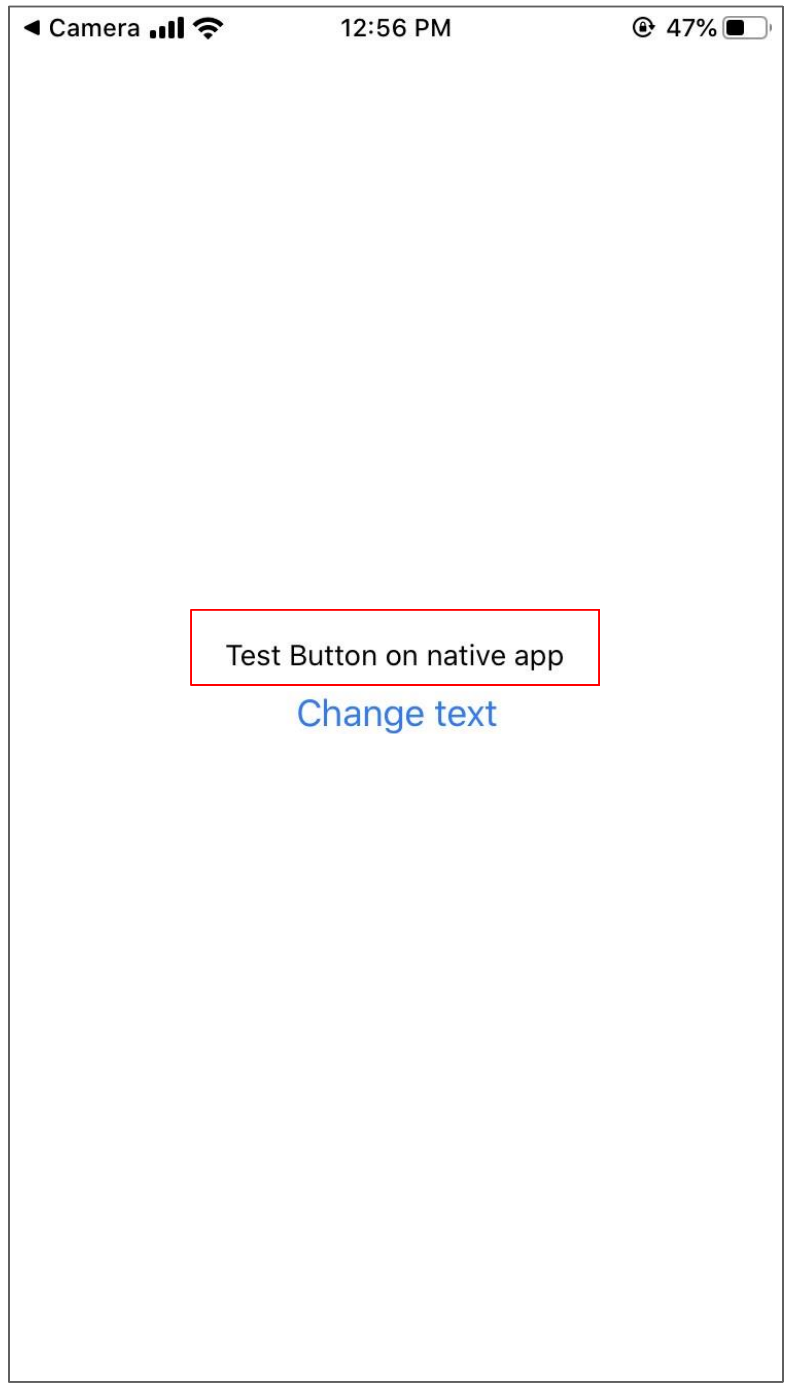
Demo: Check The Working Of Button

You can see a Change text button on screen. Press the button to perform the state change.



Demo: Check The Working Of Button (contd.)

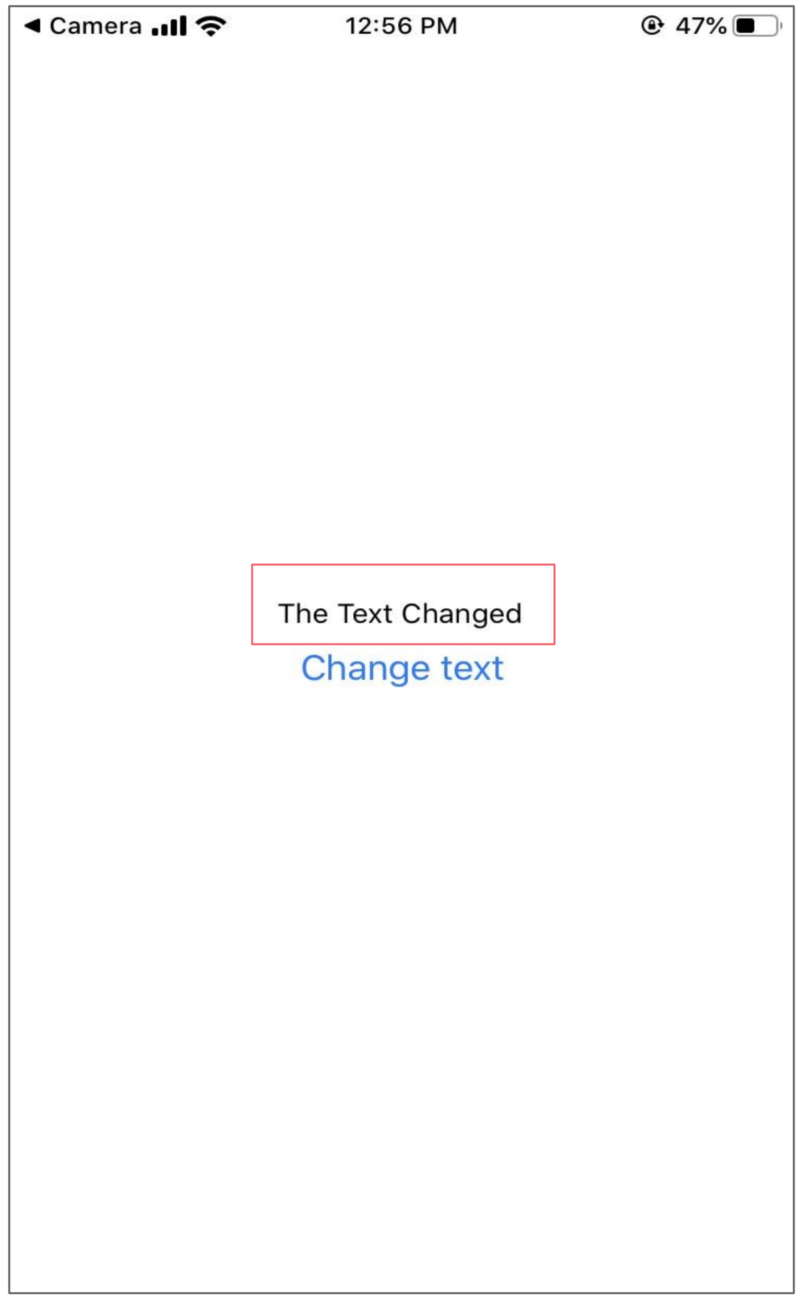
We can see one default text on the screen. “***Test button on the native app***”.



Test Button on native app

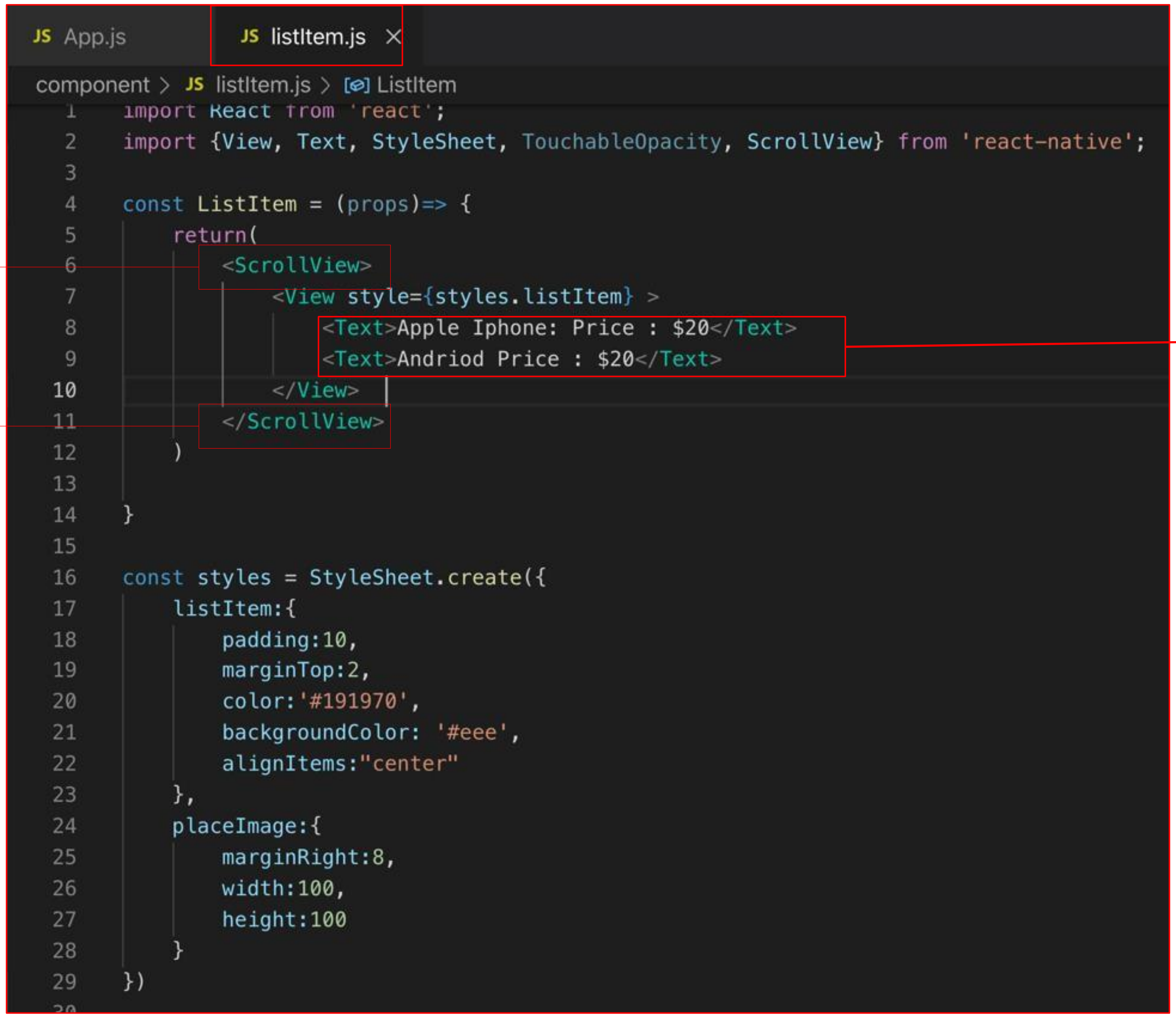
Demo: Check The Working Of Button (contd.)

Now after clicking text changes to " The text changed" this is how we implement ***button click method*** on native app and here instead of onClcik we have to use onPress keyword in application code.



Demo: Implement Scrolling Option

Create a new component listItem.js as shown below. Add the items list (static data). Make use of ***ScrollView*** to implement scrolling effect on screen.



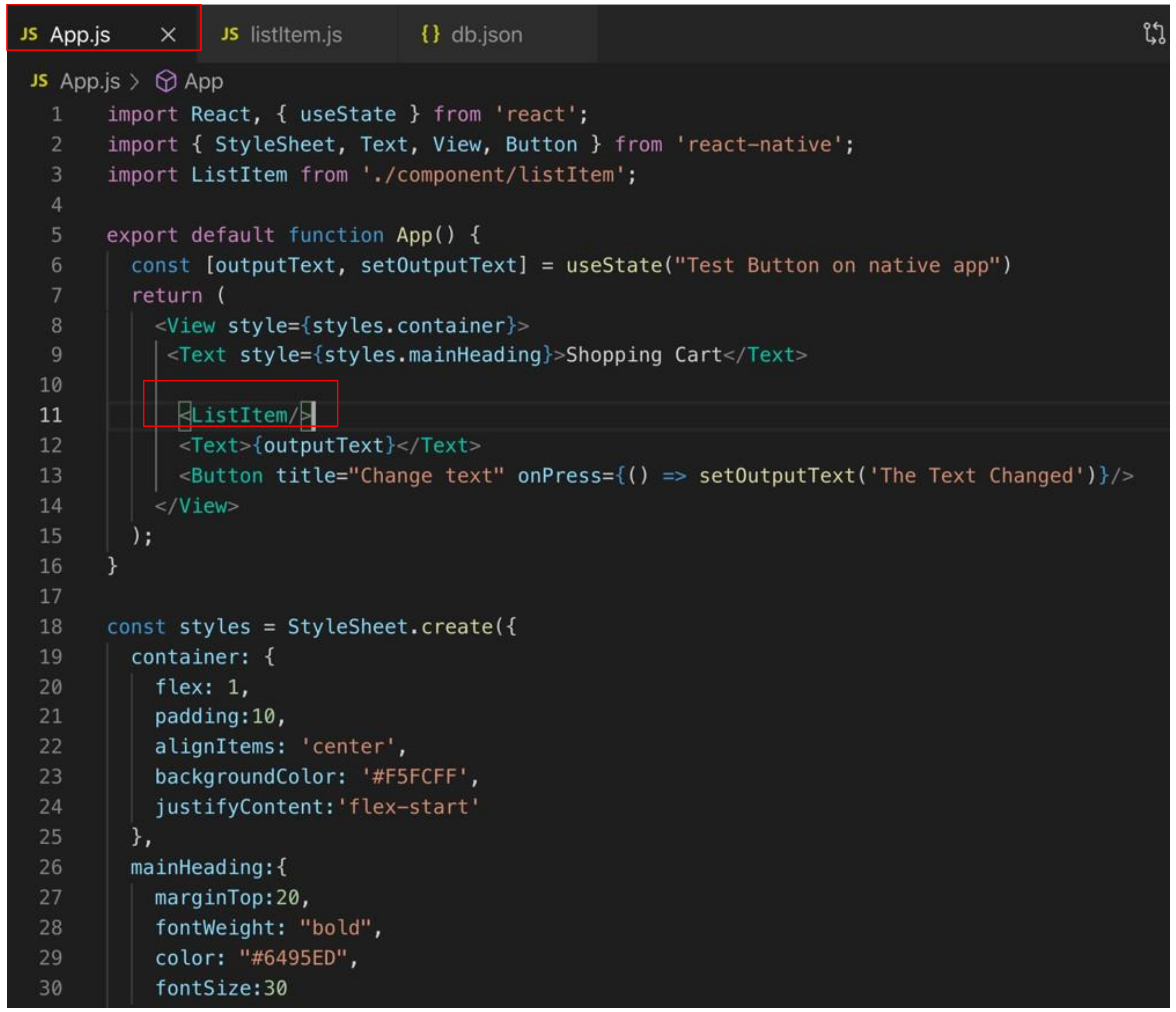
```
import React from 'react';
import {View, Text, StyleSheet, TouchableOpacity, ScrollView} from 'react-native';

const ListItem = (props)=> {
    return(
        <ScrollView>
            <View style={styles.listItem} >
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
            </View>
        </ScrollView>
    )

}

const styles = StyleSheet.create({
    listItem:{
        padding:10,
        marginTop:2,
        color:'#191970',
        backgroundColor: '#eee',
        alignItems:"center"
    },
    placeImage:{
        marginRight:8,
        width:100,
        height:100
    }
})
```

Generates scrolling effect

Items list

Demo: Connecting Components

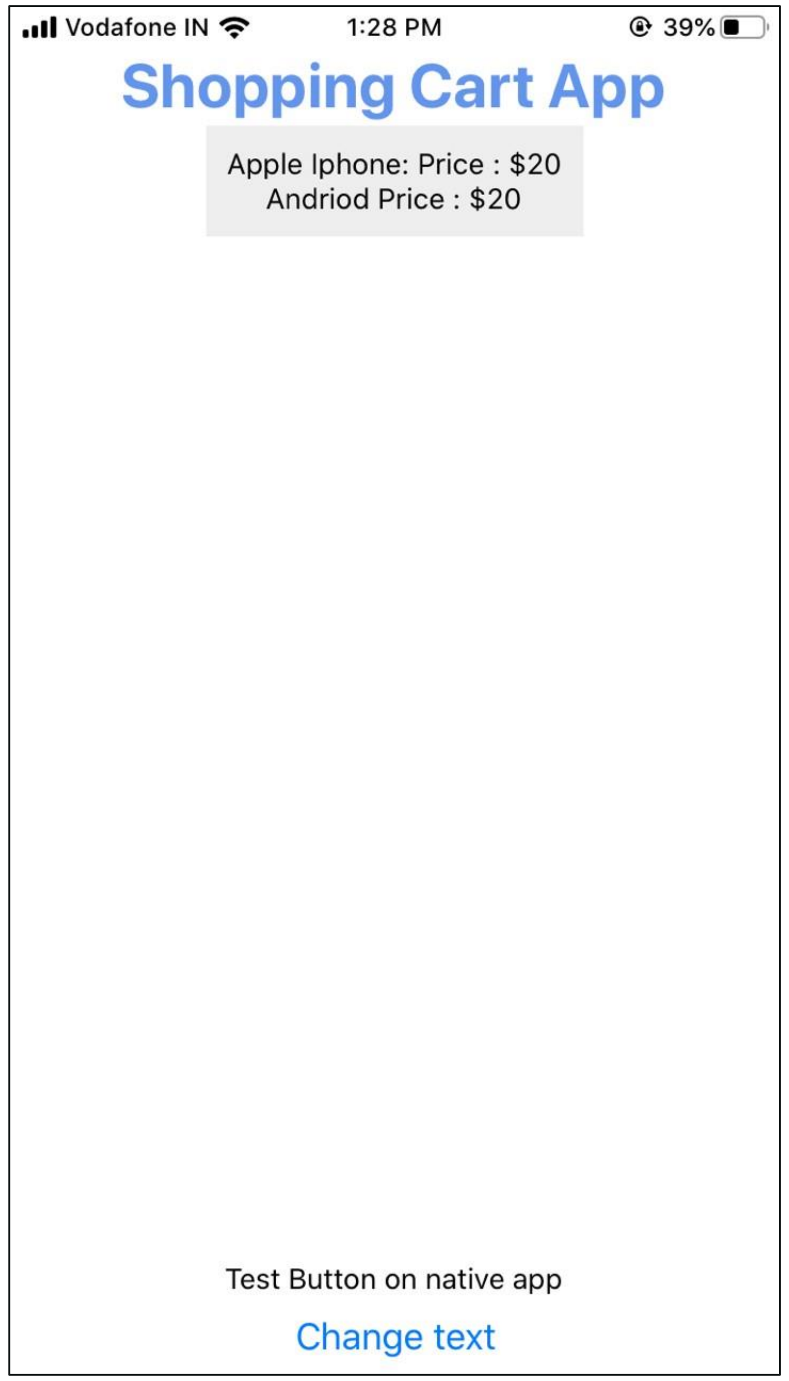
Add the listItem.js component in App.js component.



```
import React, { useState } from 'react';
import { StyleSheet, Text, View, Button } from 'react-native';
import ListItem from './component/listItem';

export default function App() {
  const [outputText, setOutputText] = useState("Test Button on native app")
  return (
    <View style={styles.container}>
     <Text style={styles.mainHeading}>Shopping Cart</Text>

      <ListItem/>
      <Text>{outputText}</Text>
      <Button title="Change text" onPress={() => setOutputText('The Text Changed')}/>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding:10,
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
    justifyContent:'flex-start'
  },
  mainHeading:{
    marginTop:20,
    fontWeight: "bold",
    color: "#6495ED",
    fontSize:30
```

Demo: Check The Output Of ListItems.js In Mobile

On saving the application code following output appears on the mobile screen.



Vodafone IN 1:28 PM 39%

Shopping Cart App

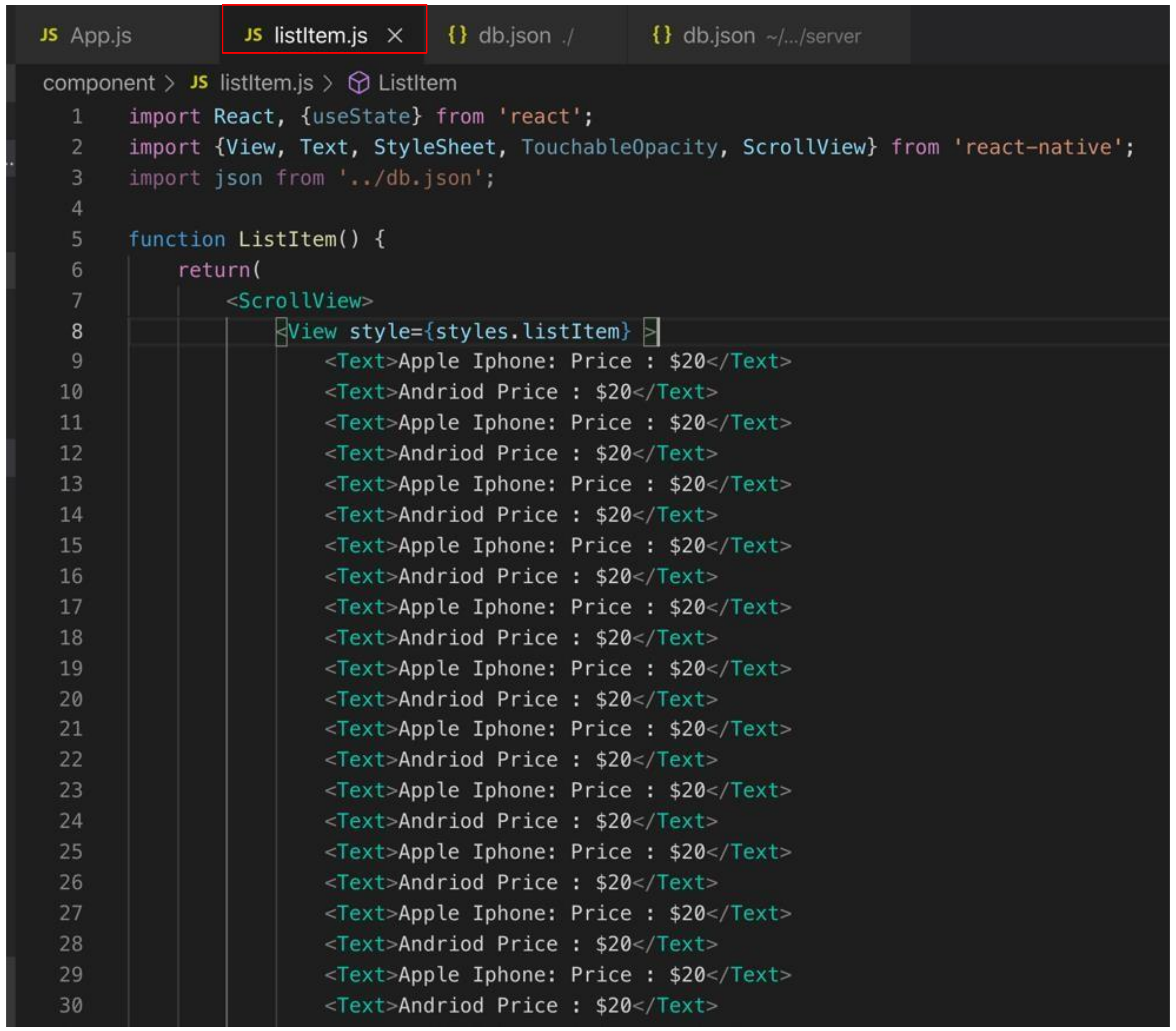
Apple Iphone: Price : $20
Andriod Price : $20

Test Button on native app

Change text

Demo: Additional Text Tags

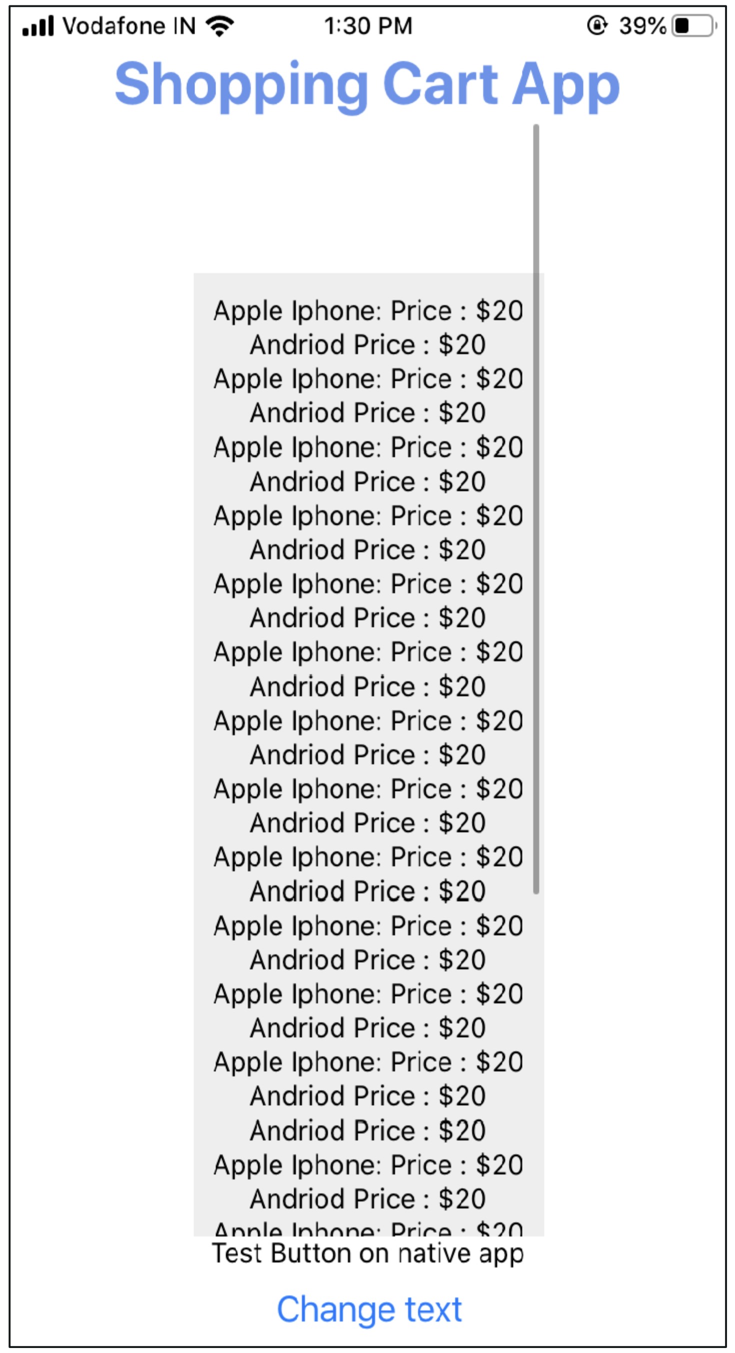
If you will add more text tag on the page, it will keep on adding in the list and we can see the complete list on mobile screen.



```
import React, {useState} from 'react';
import {View, Text, StyleSheet, TouchableOpacity, ScrollView} from 'react-native';
import json from '../db.json';

function ListItem() {
    return(
        <ScrollView>
            <View style={styles.listItem} >
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
                <Text>Apple Iphone: Price : $20</Text>
                <Text>Andriod Price : $20</Text>
```

Demo: Output On Mobile Screen

You can scroll up and down to see the list of available items.



Shopping Cart App

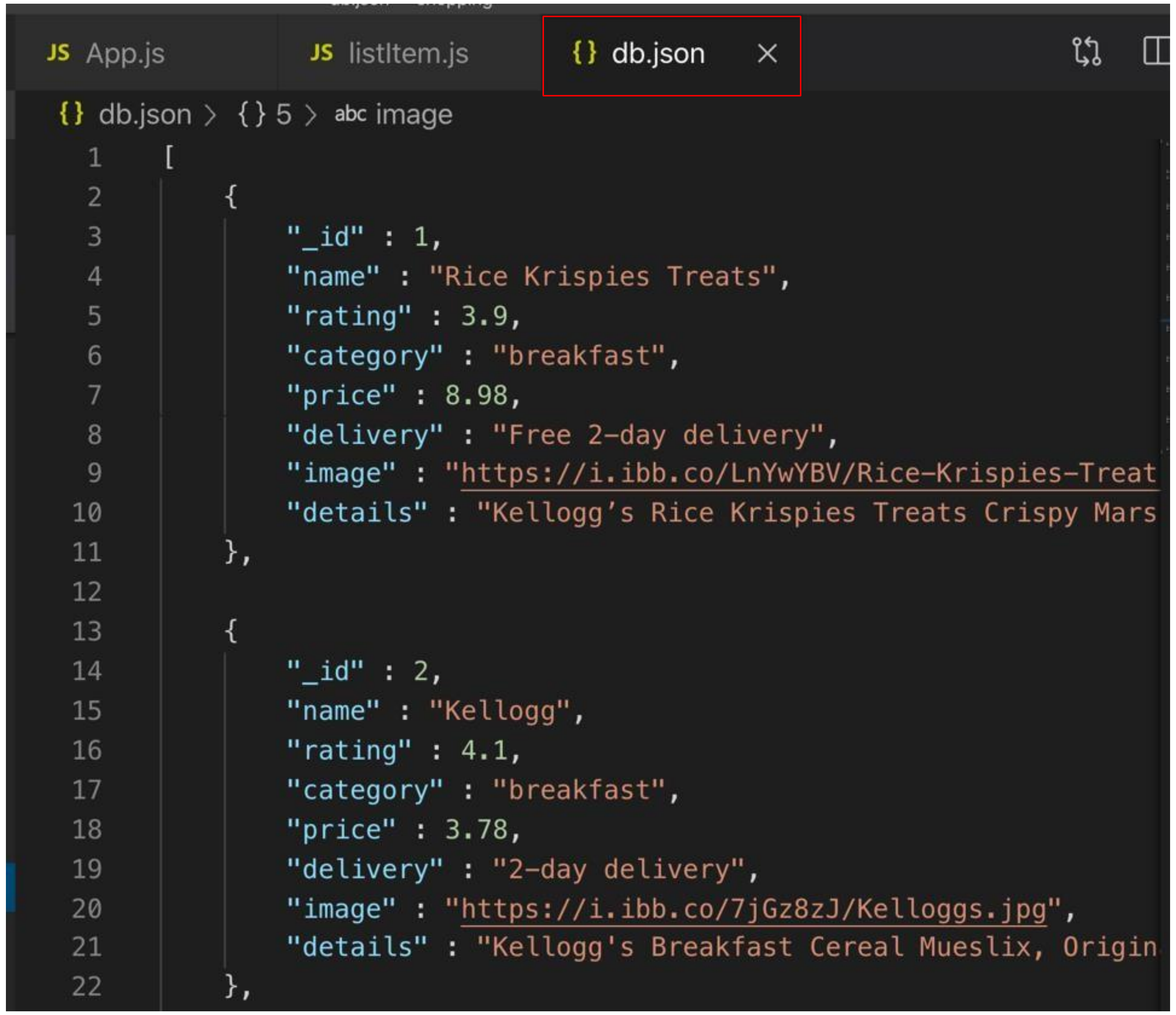
```
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
Andriod Price : $20
Apple Iphone: Price : $20
```

Test Button on native app

Change text

Demo: db.json

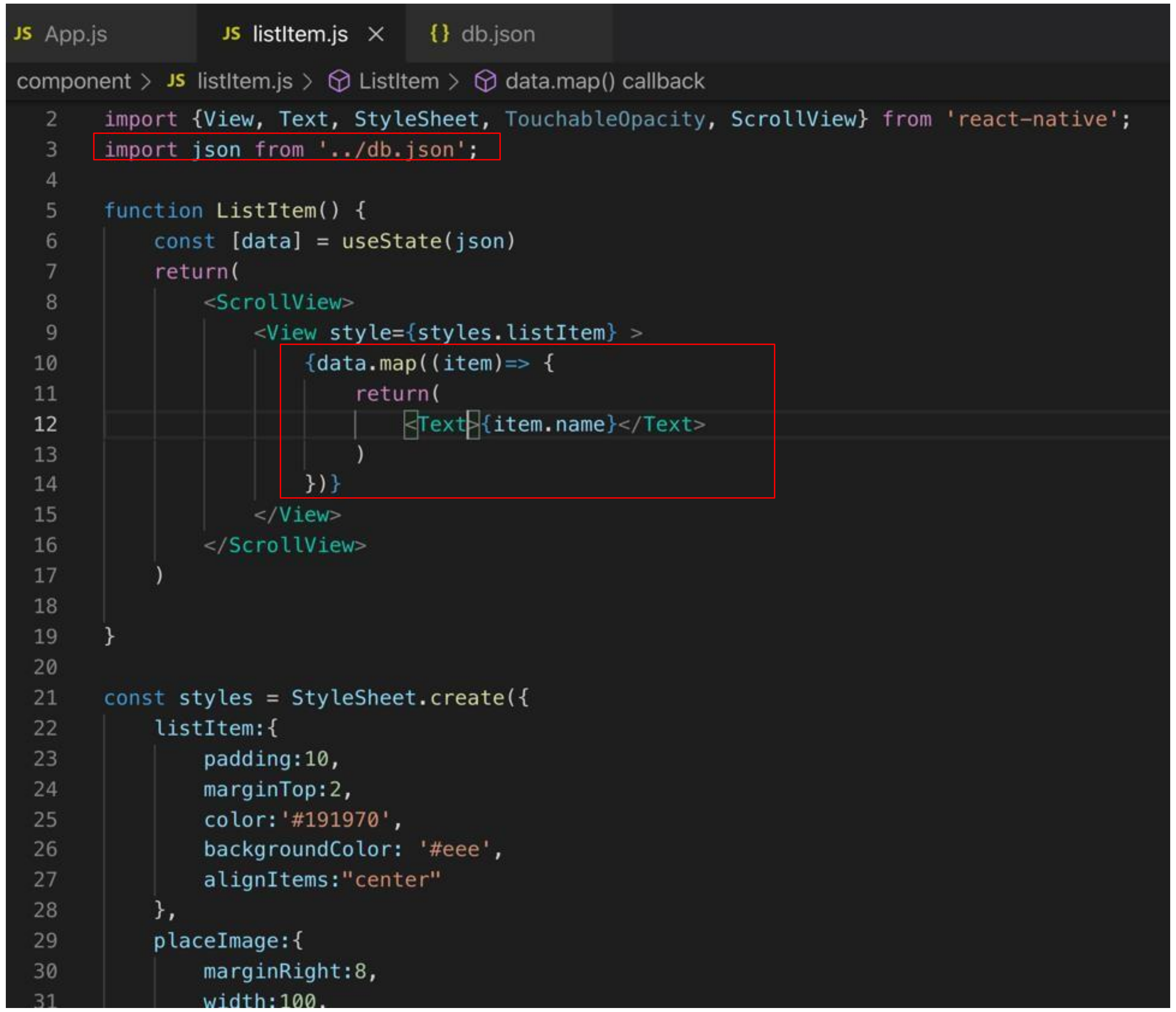
Create a separate file called ***db.json***. Here, add the proper data list to be displayed on screen.



```
[
    {
        "_id" : 1,
        "name" : "Rice Krispies Treats",
        "rating" : 3.9,
        "category" : "breakfast",
        "price" : 8.98,
        "delivery" : "Free 2-day delivery",
        "image" : "https://i.ibb.co/LnYwYBV/Rice-Krispies-Treat
        "details" : "Kellogg’s Rice Krispies Treats Crispy Mars
    },

    {
        "_id" : 2,
        "name" : "Kellogg",
        "rating" : 4.1,
        "category" : "breakfast",
        "price" : 3.78,
        "delivery" : "2-day delivery",
        "image" : "https://i.ibb.co/7jGz8zJ/Kelloggs.jpg",
        "details" : "Kellogg's Breakfast Cereal Mueslix, Origin
    },
```

Demo: Data Binding

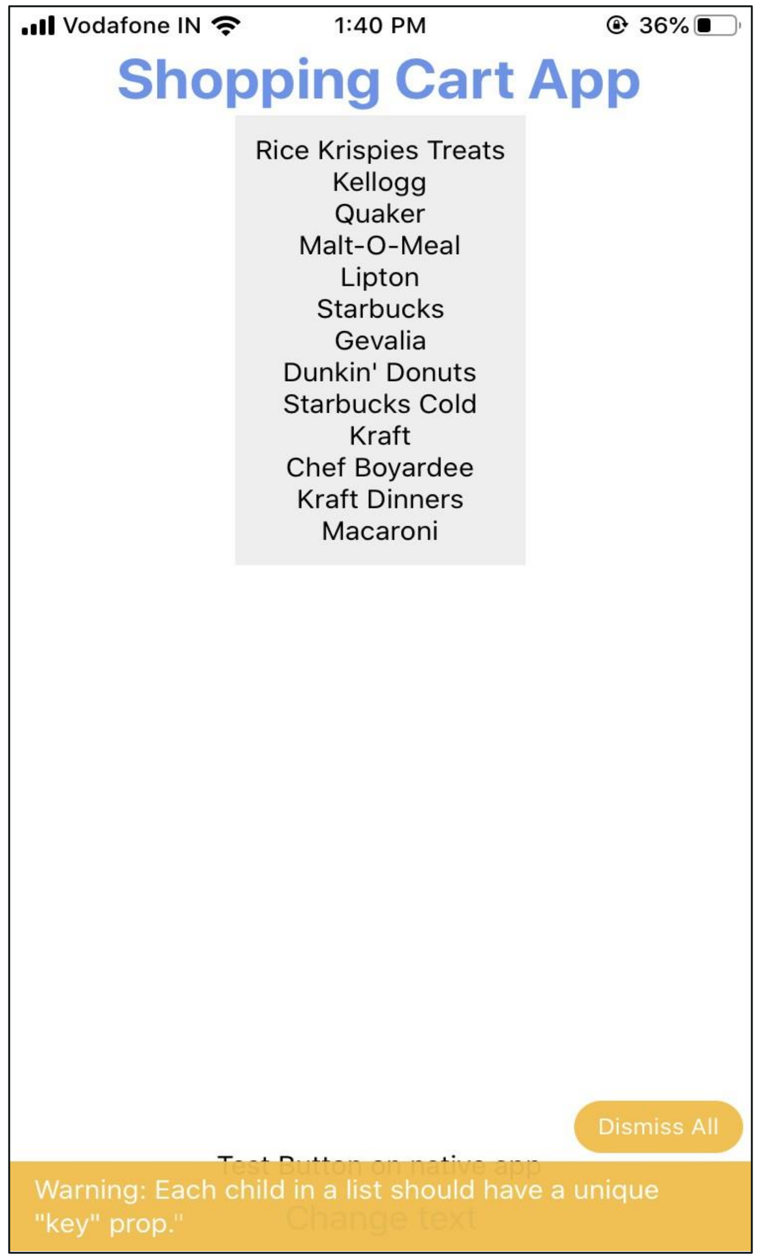
Import the db.json file in listItem.js file and using map operation perform the iteration.



```
import {View, Text, StyleSheet, TouchableOpacity, ScrollView} from 'react-native';
import json from '../db.json';

function ListItem() {
    const [data] = useState(json)
    return(
        <ScrollView>
            <View style={styles.listItem} >
                {data.map((item)=> {
                    return(
                        <Text>{item.name}</Text>
                    )
                })}
            </View>
        </ScrollView>
    )

}

const styles = StyleSheet.create({
    listItem:{
        padding:10,
        marginTop:2,
        color:'#191970',
        backgroundColor: '#eee',
        alignItems:"center"
    },
    placeImage:{
        marginRight:8,
        width:100,
```

Demo: Check Output On Mobile Screen

Now after doing map operation we can see the data is iterated over the list on the mobile screen.



Vodafone IN 1:40 PM 36%

Shopping Cart App

Rice Krispies Treats
Kellogg
Quaker
Malt-O-Meal
Lipton
Starbucks
Gevalia
Dunkin' Donuts
Starbucks Cold
Kraft
Chef Boyardee
Kraft Dinners
Macaroni

Dismiss All

Test Button on native app

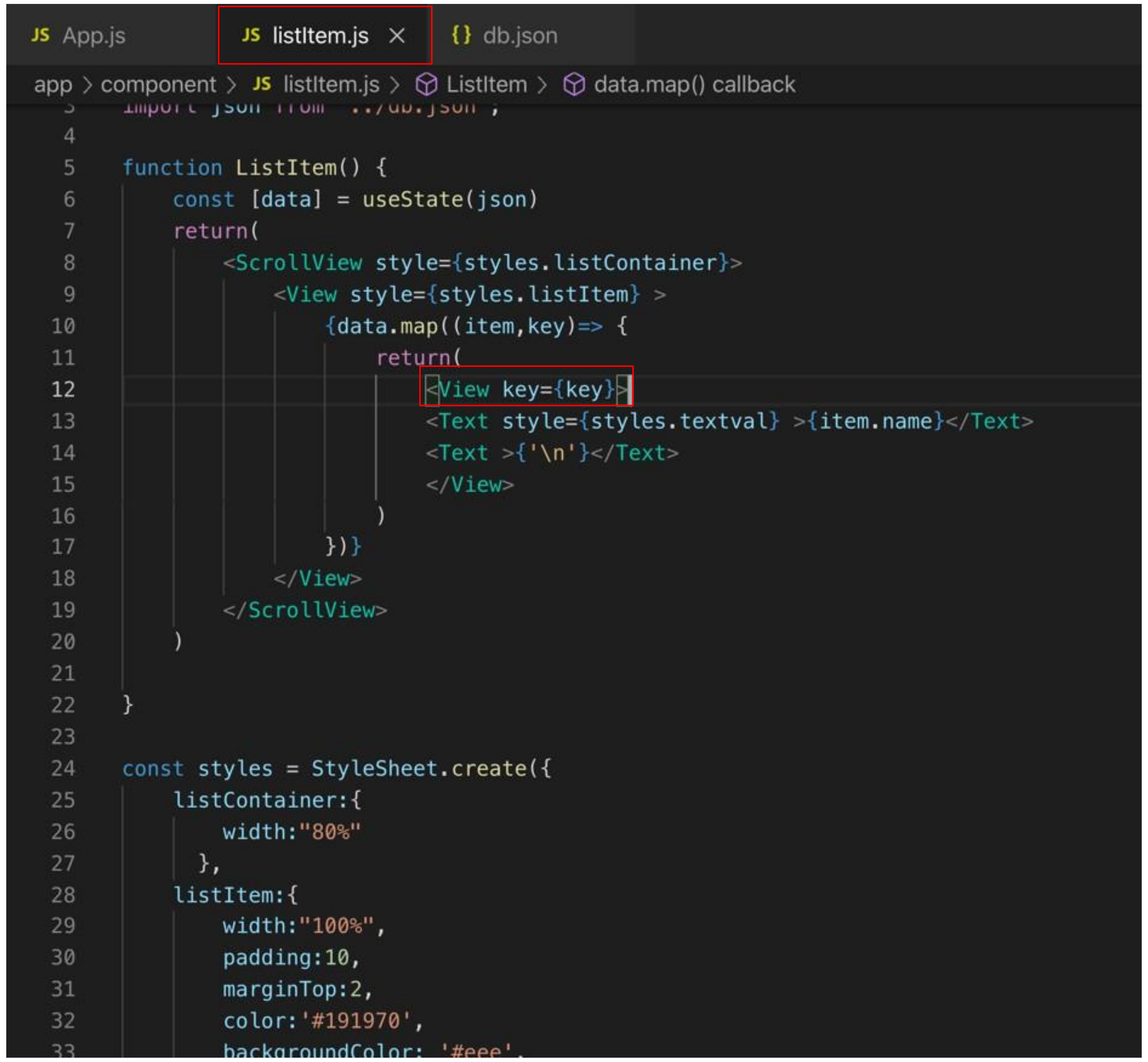
Warning: Each child in a list should have a unique "key" prop."

Change text

To resolve the warning we need to pass the key

Demo: Add Key While Mapping

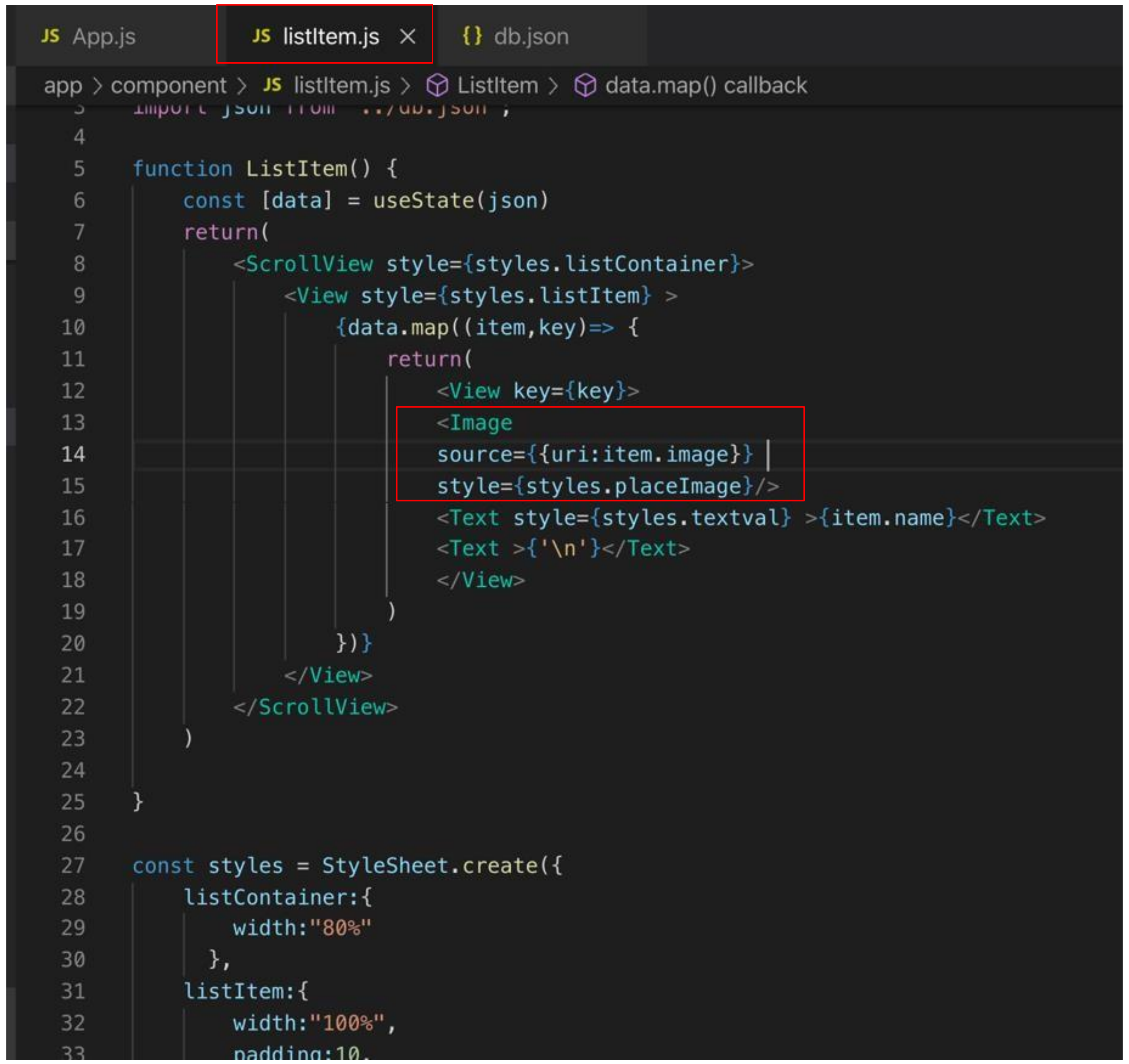
To remove the warning message, we need to pass ***key*** in map operation and bind data, just like we do in the React applications.


App.js
listItem.js
db.json
app > component > listItem.js > ListItem > data.map() callback

```
import json from '../db.json';

function ListItem() {
    const [data] = useState(json)
    return(
        <ScrollView style={styles.listContainer}>
            <View style={styles.listItem} >
                {data.map((item,key)=> {
                    return(
                        <View key={key}>
                        <Text style={styles.textval} >{item.name}</Text>
                        <Text >{'\n'}</Text>
                        </View>
                    )
                })}
            </View>
        </ScrollView>
    )

}

const styles = StyleSheet.create({
    listContainer:{
        width:"80%"
      },
    listItem:{
        width:"100%",
        padding:10,
        marginTop:2,
        color:'#191970',
        backgroundColor: '#eee',
```

Demo: Add Item Image Page

Now at the end add the product items images to the listing page to the application.

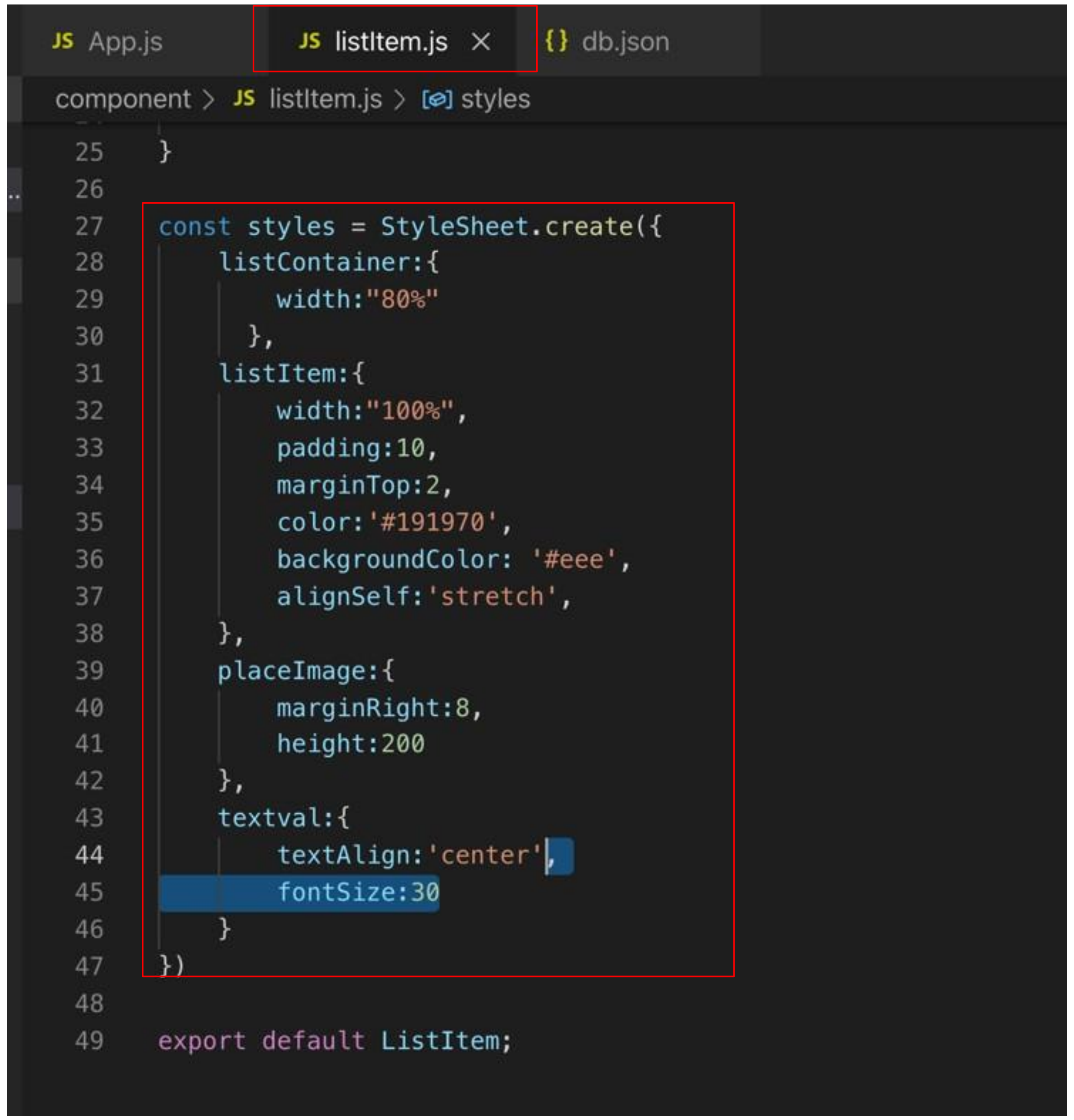


App.js
listItem.js
db.json
app > component > listItem.js > ListItem > data.map() callback

```
import json from '../db.json';

function ListItem() {
    const [data] = useState(json)
    return(
        <ScrollView style={styles.listContainer}>
            <View style={styles.listItem} >
                {data.map((item,key)=> {
                    return(
                        <View key={key}>
                        <Image
                        source={{uri:item.image}}
                        style={styles.placeImage}/>
                        <Text style={styles.textval} >{item.name}</Text>
                        <Text >{'\n'}</Text>
                        </View>
                    )
                })}
            </View>
        </ScrollView>
    )

}

const styles = StyleSheet.create({
    listContainer:{
        width:"80%"
    },
    listItem:{
        width:"100%",
        padding:10,
```

Demo: Styling Application Layout

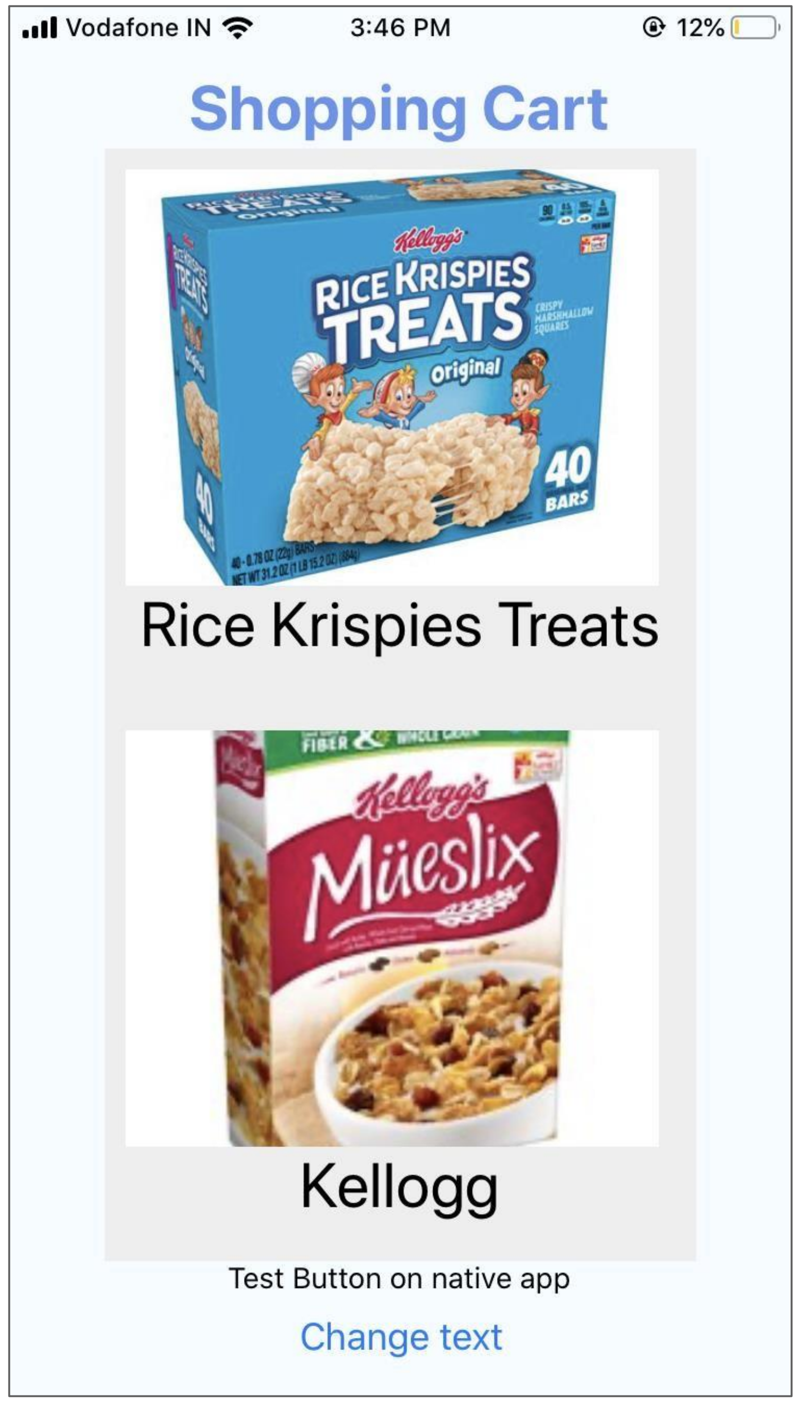
Add the styling to the application.



```
}

const styles = StyleSheet.create({
    listContainer:{
        width:"80%"
      },
    listItem:{
        width:"100%",
        padding:10,
        marginTop:2,
        color:'#191970',
        backgroundColor: '#eee',
        alignSelf:'stretch',
    },
    placeImage:{
        marginRight:8,
        height:200
    },
    textval:{
        textAlign:'center',
        fontSize:30
    }
})

export default ListItem;
```

Demo: Final Output

After adding the style to the component we can see the final output of the application on the mobile screen.



Questions

Comments
Ratings
Suggestions
FEEDBACK
Survey
Likes
Ideas