# What is React Native ?

- React Native is an open-source JavaScript framework, designed for building apps on multiple platforms like iOS, Android, and also web applications, utilizing the very same code base.
- React Native is based on React, and it brings all its glory to mobile app development.
- React Native is created by Meta Platforms, Inc.
- To learn React native you need to know basic javascript.

# What is Javascript ?

- JavaScript is the world's most popular programming language.
- JavaScript is easy to learn.
- Basic concept of javascript is given below.
    - Statements
    - Data Types
    - Operators
    - Conditional statements
    - Difference between var , let and const

## 1. Statements

- A computer program is a list of "instructions" to be "executed" by a computer.
- In a programming language, these programming instructions are called statements.
- A JavaScript program is a list of programming statements.
- JavaScript statements are composed of Values, Operators, Expressions, Keywords, and Comments.

## 2. Data Types

- JavaScript variables can hold different data types: numbers, strings, objects and more:
- Data type is a particular kind of data item, as defined by the values it can take, the programming language used, or the operations that can be performed on it.
- In javascript there are many data types.

| Data Type | Used for | Example |
|---|---|---|
| String | Alphanumeric character | “Hello World” , “India” |
| Integer | Whole Number | 1452 , 4569 |
| Float | Number with decimal point | 3.15,2.25 |
| Character | Encoding text numerically | 97 (in ASCII 97 is lowercase ‘a’ ) |
| Boolean | Representing logical values | TRUE , FALSE |

## 3. Operators

- JavaScript operators are symbols that are used to perform operations on operands.
- There are following types of operators in JavaScript.
    1. Arithmetic Operators
    2. Comparison (Relational) Operators
    3. Bitwise Operators
    4. Logical Operators
    5. Assignment Operators
    6. Special Operators

## 4. Conditional statements

- In JavaScript we have the following conditional statements:
    - Use if to specify a block of code to be executed, if a specified condition is true.
    - Use else to specify a block of code to be executed, if the same condition is false.
    - Use else if to specify a new condition to test, if the first condition is false.
    - Use switch to specify many alternative blocks of code to be executed.

# Different between var, let  and const

| var | let | const |
|---|---|---|
| It can be declared without initialization. | It can be declared without initialization | It cannot be declared without initialization. |
| It can be updated and re-declared into the scope | It can be updated but cannot be redeclared into the scope. | It cannot be updated or re-declared into the scope |
| The scope of a *var* variable is functional scope. | The scope of a *let* variable is block scope. | The scope of a *const* variable is block scope. |
| It can be accessed without initialization as its default value is “undefined”. | It cannot be accessed without initialization, as it returns an error. | It cannot be accessed without initialization, as it cannot be declared without initialization. |

# What are React Native Core Components?

- React Native has many Core Components for everything from form controls to activity indicators.

| React Native Component | Description |
|---|---|
| <Views> | It is building a user interface. |
| <Text> | It is displaying text. |
| <Image> | The Image component is used to display the image on the screen. |
| <ScrollView> | The ScrollView is a generic scrolling container . |
| <TextInput> | Allow the user to enter text. |

## Example:

- In this program I am trying to display a simple view which includes <Text> <Image> , <TextInput> and also add <ScrollView> .

```
import React from 'react';
import { View, Text, Image, ScrollView,
TextInput } from 'react-native';
const App = () => {
  return (
    <ScrollView>
      <Text>Hello</Text>
      <View>
        <Text>This is my first Program</Text>
        <Image
          source={{
            uri: ' IMAGE PATH ',
          }}
          style={{ width: 200, height: 200 }}   />
      </View>
      <TextInput
        style={{
          height: 40,
          borderColor: 'gray',
          borderWidth: 1
        }}
        defaultValue="You can type in me"
// placeholder />
    </ScrollView>
  );
}
export default App;
```

OUTPUT

Hello
This is my first Program

IMG

One textbox show
On this text box which user write anything

# Difference between Function Components & Class Components.

| Function Components | Class Components |
| --- | --- |
| There is no render method used in functional components. | It must have the render() method returning JSX( JSX means Javascript XML. which is syntactically similar to HTML) |
| A functional component is just a plain JavaScript pure function that accepts props as an argument and returns a React element. | A class component requires you to extend from React. Component and create a render function which returns a React element. |
| Also known as Stateless components which means you can't use this. | Also known as Stateful components. |
| React lifecycle methods cannot be used in functional components. | React lifecycle methods can be used in Class components. |
| Hooks can be easily used in functional components. | It requires different syntax inside a class component to implement hooks. |
| Constructors are not used. | Constructors are used. |
| import React from 'react';<br>import { Text, View } from 'react-native';<br><br>const HelloWorldApp = () => {<br>return (<br><View style={{<br>flex: 1,<br>justifyContent: 'center',<br>alignItems: 'center'<br>}}><br><Text>Hello, world!</Text><br></View><br>);<br>}<br><br>export default HelloWorldApp; | import React, { Component } from 'react';<br>import { Text, View } from 'react-native';<br><br>class HelloWorldApp extends Component {<br>render() { // render method Component class<br>return (<br><View style={{<br>flex: 1,<br>justifyContent: "center",<br>alignItems: "center"}}><br><Text>Hello, world!</Text><br></View><br>);<br>}<br>}<br><br>export default HelloWorldApp; |

# React Native Installation

- When you install React Native you will need Node js , JDK , React Native command line interface, and Android Studio.

### 1. Node

- Follow the installation instructions to install Node 14 or newer.
- Download and install the latest version of nodeJs.

### 2. JDK

- React Native also requires Java SE Development Kit (JDK).
- Download and install the latest version of JDK and set the environment path.

### 3. Android Studio

- Follow the below steps to download and install Android Studio.
  - **Install Android Studio**
    - Download and install Android Studio.
    - Android Studio Download always **Intel Chip** If you use mac OS.
    - While on Android Studio installation wizard, make sure the boxes next to all of the following items are checked:
      - Android SDK
      - Android SDK Platform
      - Android Virtual Device
  - **Install the Android SDK**
    - Android Studio installs the latest Android SDK by default
    - Building a React Native app with native code, however, requires the Android 11 (R) SDK in particular.
    - Additional Android SDKs can be installed through the SDK Manager in Android Studio.

    - To do that, open Android Studio, click on the "Configure" button and select "SDK Manager".
    - Select the "SDK Platforms" tab from within the SDK Manager, then check the box next to "Show Package Details" in the bottom right corner.
    - Look for and expand the Android 11 (R) entry, then make sure the following items are checked:
        - Android SDK Platform 30
        - Intel x86 Atom_64 System Image or Google APIs Intel x86 Atom System Image
    - Next, select the "SDK Tools" tab and check the box next to "Show Package Details" here as well.
    - Look for and expand the "Android SDK Build-Tools" entry, then make sure that 30.0.2 is selected .
    - Finally, click "Apply" to download and install the Android SDK and related build tools.

- **Configure the ANDROID_HOME environment variable**
    - The React Native tools require some environment variables to be set up in order to build apps with native code.
        - Open the Windows Control Panel.
        - Click on User Accounts, then click User Accounts again
        - Click on Change my environment variables
        - Click on New... to create a new ANDROID_HOME user variable that points to the path to your Android SDK.

## The commands are given below

| |
|---|
| export ANDROID_SDK_ROOT=$HOME/Library/Android/sdk |
| export PATH=$PATH:$ANDROID_SDK_ROOT/emulator |
| export PATH=$PATH:$ANDROID_SDK_ROOT/platform-tools |

# Creating a new application

- React Native has a built-in command line interface, which you can use to generate a new project.
- Let's create a new React Native project called "FirstProject" . The command given below.

```
npx react-native init FirstProject
```

- After some process your project is created.

# Running your React Native application

### Step 1: Start Metro

- First, you will need to start Metro, the JavaScript bundler that ships with React Native.
- To start Metro, run npx react-native start inside your React Native project folder:

```
npx react-native start
```

### Step 2: Start your application

- Let Metro Bundler run in its own terminal. Open a new terminal inside your React Native project folder. Run the following:

```
npx react-native run-android
```

- After the complete process your application runs.
- Thus all are the steps to perform the React native application on your device.

# React native Basic Command :

## Starting a new project

- There are different ways you can bootstrap a react native application.
- You can use **Expo** or create-react-native-app (which in turns uses Expo-Cli) to start your new project, but with this method you are in more control of what happened in your project and can communicate, tweak and write your own modules with native libraries for iOS and Android mobile platform.
- The commanond are given below.

| react-native init [ Project-Name ] |
| --- |
| cd [ Project-Name ] |

## Run app in Android emulator

- This command is self explanatory and as it says it will start the Android emulator and install the app you just created.
- You need to be in the root of the project to run this command.
- The command is given below.

| react -native run-android |
| --- |

## Run app in Android emulator

- This command does exactly the same as react-native run-android but instead of the Android emulator, it opens the iPhone simulator.

| react-native run-ios |
| --- |

## Link dependencies to native project

- Some libraries have dependencies that need to be linked in the native code generated for React Native. If something doesn't work after you installed a new library, maybe it is because you skipped this step.
- The command is given below.

| react -native link [Library-Name] |
|---|

## Clear Bundle

- If something doesn't run as expected, maybe you need to clear and create a new bundle with this command.
- The command is given below.

| Watchman watch-del-all |
|---|

## Support decorators

- JSX doesn't support decorators by default so you need to install the **Babel** plugin to make it work.
- The commands are as follows.

| npm install babel-plugin-transform-decorators-legacy –save |
|---|
| Npm install babel-plugin-transform-class-properties –save |

# Export APK to run in device

- With the following commands you will have an unsigned apk so you can install and share with your colleagues for testing purposes.
- Just remember that this apk is not ready to upload to the App Store or production.
- You will find your fresh apk in android/app/build/outputs/apk/app-debug.apk.

### 1. Bundle debug build

- react-native bundle --dev false --platform android --entry-file index.android.js --bundle-output ./android/app/build/intermediates/assets/debug/index.android.bundle --assets-dest ./android/app/build/intermediates/res/merged/debug

### 2. Create debug build

- cd android
- ./gradlew assembleDebug

# React-native First Application :

- Let's build our first React Native application on Windows as a development operating system and Android as a target operating system.

### Steps to create React Native application :

1. First, you have to start your emulator (Android Emulator) and make it live.
2. Create a directory (ReactNative) in your any drive.
3. Open "Command Prompt" and go to your ReactNative directory.
4. Write a command react-native init FirstApp to initialize your app "FirstApp".

5. Go to your directory location "FirstApp" and run the command **react-native run-android**. It will start the Node server and launch your application in a virtual device emulator.

**Note :** Before run the application you must check the environment path propar defined.

## View Code of React Native Application

- Open your one of the favorite JavaScript supportable IDE and open **App.js** file inside your FirstApp application.

## Create a simple React Native "Hello World" application

- Create a simple "Hello World" app by editing the App.js file of FirstApp.
- Save the application and reload by simply pressing "R".
- The code is given below.

App.js

```
import React from 'react';
import { Text, View } from 'react-native';
const HelloWorldApp = () => {
  return (
    <View
      style={{
        flex: 1,
        justifyContent: "center",
        alignItems: "center"
      }}>
      <Text>Hello, world!</Text>
    </View>
  )
}
export default HelloWorldApp;
```

## React Native Code Explanation

- **import React, {Component} from 'react':** imports the library and other components from the react module and assigns them to variable React.
- **const instruction:** sets to display the platform-specific message.
- **export default class App extends Component:** defines the classes that extend the React Component. The export default class modifier makes the class "public". This block of code defines the components that represent the user interface.

- **render():** a function that returns a React element.
- **return():** returns the result of layout and UI components.
- **View:** a container that supports the layout accessibility controls. It is a fundamental component for building the UI.
- **Text:** a React component for displaying text.
- **style:** a property used for styling the components using StyleSheet.
- **styles:** used to design individual components.
    - {styles.instructions}>{instructions}:
- **const styles = StyleSheet.create({}):** The **StyleSheet** class creates the style object that controls the layout and appearance of components. It is similar to Cascading Style Sheets (CSS) used on the Web.

# React Native View

- The **View** is the fundamental component of React Native for building a user interface.
- It is a container that supports layout with flexbox, style, touch handling, and accessibility controls.
- It maps directly to the native view similar to whatever platform the React Native app is running on.
- It displays the components regardless of **UIView, , android.view,** etc.
- View components can be nested, containing other views inside it.
- It can contain 0 to many children of any type.

## Props of View

| | | |
|---|---|---|
| onStartShouldSetResponder | accessibilityLabel | accessibilityHint |
| nativeID | onAccessibilityTap | onLayout |
| onMoveShouldSetResponder | onMoveShouldSetResponderCapture | onResponderGrant |
| onResponderReject | onResponderRelease | style |
| accessible | onStartShouldSetResponderCapture | pointerEvents |
| onResponderTerminate | accessibilityComponentType | testID |
| accessibilityRole | accessibilityStates | accessibilityTraits |
| accessibilityElementsHidden | accessibilityIgnoresInvertColors | shouldRasterizeIOS |

## Example

- In this example, we create a View component that contains two colored boxes and a text component in a row with height and width.

## App.js

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


export default class SwitchExample extends Component {
```

```
   render() {
      return (
         <View style={styles.container}>
            <View style={{backgroundColor: 'blue', flex: 0.3}} />
            <View style={{backgroundColor: 'red', flex: 0.5}} />
                  <Text style={{fontSize: 18}}>View Example</Text>
         </View>
      );  }   }
const styles = StyleSheet.create ({
    container: {
       flex: 1,
       flexDirection: 'row',
       height: 100,   width: "80%",
       backgroundColor:"#5ead97"
    }
 })
```

# React Native Button

- Most users interact with mobile through touches. There are combinations of gestures that work on it, such as tapping on the button, zooming the map, scrolling a list, etc.
- A button is one of the components that work on its click.
- React Native Button is a basic component that works by clicking on it. It imports the Button class of react-native.

## Props of View

| Prop | Type | Description |
|---|---|---|
| onPress | function | Call the handler when the user clicks the button. |
| title | string | Display the text inside the button. |
| accessibilityLabel | string | Display the text for blindness accessibility features. |
| color | Color | Set the background color of the Android button or set the color of iOS text. |
| disabled | bool | It disables all interactions for this component, if true |
| textID | string | Used to locate this view in end-to-end tests. |
| hasTVPreferredFocus | bool | It preferred TV focus work only for Apple TV. |

## Example

```
import React, { Component } from 'react';
import { Alert, AppRegistry, Button, StyleSheet, View } from 'react-native';

export default class ButtonBasics extends Component {
        onPressButton() {
            Alert.alert('You clicked the button!')
        }
        render() {
            return (
                <View style={styles.container}>
                    <View style={styles.buttonContainer}>
                        <Button  onPress={this.onPressButton} title="Press Me" />
                    </View>
                    <View style={styles.buttonContainer}>
                        <Button onPress={this.onPressButton} title="Press Me"
                            color="#009933"  />
                    </View>
                    <View style={styles.multiButtonContainer}>
                        <Button onPress={this.onPressButton} title="A disabled button"
                            disabled={true}/>
```

```
                    <Button onPress={this.onPressButton}  title="OK!"
                        color="#009933"    />
                </View>
            </View>
        );    } }
const styles = StyleSheet.create({
    container: {
        flex: 1,
        justifyContent: 'center',  },
    buttonContainer: {
        margin: 20  },
    multiButtonContainer: {
        margin: 20,
        flexDirection: 'row',
        justifyContent: 'space-between'  }
})
```

# React Native ScrollView

- The ScrollView is a generic scrollable container, which scrolls multiple child components and views inside it.
- In the ScrollView, we can scroll the components in both directions vertically and horizontally.
- By default, the ScrollView container scrolls its components and views vertically.
- To scroll its components horizontally, it uses the props horizontal: true (default, horizontal: false).

# Example

```
import React from 'react';
import { StyleSheet, Text, SafeAreaView, ScrollView, StatusBar } from 'react-native';

const App = () => {
  return (
    <SafeAreaView style={styles.container}>
      <ScrollView style={styles.scrollView}>
        <Text style={styles.text}>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id
est laborum.
        </Text>
      </ScrollView>
    </SafeAreaView>
  );
}
const styles = StyleSheet.create({
  container: {
    flex: 1,
    },
  scrollView: {
    backgroundColor: 'pink',
    marginHorizontal: 20,
  },
  text: {
    fontSize: 42,
  },
});
export default App;
```

## Props of ScrollView

| | |
|---|---|
| StickyHeaderComponent | alwaysBounceHorizontal |
| alwaysBounceVertical | automaticallyAdjustContentInsets |
| automaticallyAdjustKeyboardInsets | automaticallyAdjustsScrollIndicatorInsets |
| bounces | bouncesZoom |
| canCancelContentTouches | centerContent |
| indicatorStyle | onScroll |
| onScrollEndDrag | pagingEnabled |
| scrollsToTop | snapToInterval |

# React Native FlatList

- The FlatList component displays the similar structured data in a scrollable list.
- It works well for large lists of data where the number of list items might change over time.
- The FlatList shows only those rendered elements which are currently displaying on the screen, not all the elements of the list at once.
- The FlatList component takes two required props: data and renderItem.
- The data is the source of elements for the list, and renderItem takes one item from the source and returns a formatted component to render.
- To implement the FlatList component, we need to import FlatList from the 'react-native' library.

## Example

```
import React, { useState } from "react";
import { FlatList, SafeAreaView, StatusBar, StyleSheet, Text, TouchableOpacity } from
"react-native";

const DATA = [
  {
    id: 101,
```

```
    title: "First Item",
  },
  {
    id: 102,
    title: "Second Item",
  },
  {
    id: 103,
    title: "Third Item",
  },
];

const Item = ({ item, onPress, backgroundColor, textColor }) => (
  <TouchableOpacity onPress={onPress} style={[styles.item, backgroundColor]}>
    <Text style={[styles.title, textColor]}>{item.title}</Text>
  </TouchableOpacity>
);

const App = () => {
  const [selectedId, setSelectedId] = useState(null);

  const renderItem = ({ item }) => {
    const backgroundColor = item.id === selectedId ? "#6e3b6e" : "#f9c2ff";
    const color = item.id === selectedId ? 'white' : 'black';

    return (
      <Item
        item={item}
        onPress={() => setSelectedId(item.id)}
        backgroundColor={{ backgroundColor }}
        textColor={{ color }}
      />
    );
  };
  return (
    <SafeAreaView style={styles.container}>
      <FlatList  data={DATA}    renderItem={renderItem}    keyExtractor={(item) =>
item.id}
        extraData={selectedId}   />
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
```

```
    marginTop: StatusBar.currentHeight || 0,
  },
  item: {
    padding: 20,
    marginVertical: 8,
    marginHorizontal: 16,
  },
  title: {
    fontSize: 32,
  },
});
export default App;
```

## Props of FlatList

| | |
|---|---|
| renderItem | data |
| ItemSeparatorComponent | ListEmptyComponent |
| ListFooterComponent | extraData |
| getItemLayout | horizontal |
| inverted | numColumns |
| onRefresh | progressViewOffset |

# React Native Text Input

- TextInput is the fundamental component to input text. .
- It has several props which configure the different features, such as onChangeText that takes a function and calls it whenever the text changes.
- The onSubmitEditing prop takes a function, which is called when the text is submitted.
- TextInput has by default a border at the bottom of its view.

- This border has its padding set by the background image provided by the system, and it cannot be changed.

# Example

```
import React from "react";
import { SafeAreaView, StyleSheet, TextInput } from "react-native";

const textinput = () => {
  return (
    <SafeAreaView>
      <TextInput
        style={styles.input}
       value=”Simple text ” />
      <TextInput
        style={styles.input}
        onChangeText={onChangeNumber}
        value={number}
        placeholder="with  placeholder"
        keyboardType="numeric"
      />
    </SafeAreaView>
  );   };
const styles = StyleSheet.create({
  input: {
    height: 40,
    margin: 12,
    borderWidth: 1,
    padding: 10,
  },  });
export default textinput;
```

# Props of TextInput

| | |
|---|---|
| allowFontScaling | autoFocus |
| defaultValue | keyboardType |
| maxLength | multiline |
| textAlign | selection |
| selectionColor | style |
| value | placeholder |

| | |
|---|---|
| onPress | onLayout |
| placeholderTextColor | textalign |

# React Native Image

- The Image component is used to display the image on the screen.
- The image can be loaded from different sources such as static resources, temporary local files, local disc, network images, etc.

## Static Image Resources

❖ The static images are added in app by placing it in somewhere in the source code directory and provide its path as:

```
< Image source={require('./icon_name.png')} />
```

❖ In the above syntax, the packager will look for icon_name.png in the same folder as the component that requires it.

❖ To load the image on a specific platform, it should be named by specifying the platform as extensions such as icon_name.ios.png and icon_name.android.png.

## Uri Data Images

❖ The encoded image data uses the "data:" uri scheme in the image component. The data image is also required to specify the dimension of the image.

# Props of Image

| | |
|---|---|
| style | onPress |
| accessible | blurRadius |
| defaultSource | loadingIndicatorSource |
| onLoad | resizeMode |

# Example

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

const styles = StyleSheet.create({
  container: {
    paddingTop: 50,
  },
  tinyLogo: {
    width: 50,
    height: 50,
  },
  logo: {
    width: 66,
    height: 58,
  },
});

const DisplayAnImage = () => {
  return (
    <View style={styles.container}>
      <Image
        style={styles.tinyLogo}
        source={require('@expo/snack-static/react-native-logo.png')}
      />
      <Image
        style={styles.tinyLogo}
        source={{
          uri: 'https://reactnative.dev/img/tiny_logo.png',
        }}
      />
      <Image
```

```
        style={styles.logo}
        source={{
          uri: ' image link online ',
        }}
      />
    </View>
  );
}

export default DisplayAnImage;
```

# React Native Modal

- The **React Native Modal** is a type of View component which is used to present the content above an enclosing view.
- There are three different types of options (slide, fade and none) available in a modal that decides how the modal will show inside the react native app.
- The Modal show above the screen covers all the application areas.
- To use the **Modal** component in our application, we need to import **Modal** from the **react-native** library.

## Modal Props

| | |
|---|---|
| visible | onRequestClose |
| onShow | transparent |
| animationType | hardwareAccelerated |
| onDismiss | onOrientationChange |
| presentationStyle | animated |

# Example

```
import React, { useState } from "react";
import { Alert, Modal, StyleSheet, Text, Pressable, View } from "react-native";

const App = () => {
  const [modalVisible, setModalVisible] = useState(false);
  return (
    <View style={styles.centeredView}>
      <Modal
        animationType="slide"
        transparent={true}
        visible={modalVisible}
        onRequestClose={() => {
          Alert.alert("Modal has been closed.");
          setModalVisible(!modalVisible);
        }} >
        <View style={styles.centeredView}>
          <View style={styles.modalView}>
            <Text style={styles.modalText}>Hello World!</Text>
            <Pressable
              style={[styles.button, styles.buttonClose]}
              onPress={() => setModalVisible(!modalVisible)} >
              <Text style={styles.textStyle}>Hide Modal</Text>
            </Pressable>
          </View>
        </View>
      </Modal>
      <Pressable
        style={[styles.button, styles.buttonOpen]}
        onPress={() => setModalVisible(true)} >
        <Text style={styles.textStyle}>Show Modal</Text>
      </Pressable>
    </View>
  );
};
const styles = StyleSheet.create({
  centeredView: {
    flex: 1,
    justifyContent: "center",
    alignItems: "center",
    marginTop: 22},
  modalView: {
    margin: 20,
    backgroundColor: "white",
```

```
    borderRadius: 20,
    padding: 35,
    alignItems: "center",
  },
  button: {
    borderRadius: 20,
    padding: 10,
    elevation: 2
  },
  buttonOpen: {
    backgroundColor: "#F194FF",
  },
  buttonClose: {
    backgroundColor: "#2196F3",
  },
  textStyle: {
    color: "white",
    fontWeight: "bold",
    textAlign: "center"
  },

  modalText: {
    marginBottom: 15,
    textAlign: "center"
  }
});
export default App;
```

# React-native Navigation

- React Native Navigation is used for managing the presentation, and transition between multiple screens.
- There are two types of navigation built in mobile applications.
    - These are stack navigation and tabbed navigation patterns.

## Installation and setup

- First, you need to install them in your project:

```
npm install @react-navigation/native @react-navigation/native-stack
```

- If any type of error occurs then the following command is used .

```
npm install react-native-reanimated react-native-gesture-handler react-native-screens
react-native-safe-area-context @react-native-community/masked-view
```

- After the installation successfully then following command.
- After successfully installing the library you can easily use the navigation components.

# Example

```
import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';

const Stack = createNativeStackNavigator();

const MyStack = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen
          name="Home"
          component={HomeScreen}
          options={{ title: 'Welcome' }}
        />
        <Stack.Screen name="Profile" component={ProfileScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
};
```

# React Native Dropdown (Picker)

- React Native Picker is a component which is used to select an item from multiple choices. This is the same as a Dropdown option.
- Picker is used when we need to provide an alternative to choose from multiple options.
- It is used by importing the Picker component from the react-native library.

### Library Install :

```
npm install --save react-native-picker/picke
```

## Props of Picker :

| Sr.No | Props & Description |
|---|---|
| 1 | **enabled** : It takes a boolean value. The picker will be disabled if set to false and the user will not be able to select the item. |
| 2 | **itemStyleStyling** : to be applied for the items. |
| 3 | **mode :** This property decides how to display the items of the picker. The options available are : dialog and dropdown. If dialog mode the picker items will be displayed in a modal dialog. If dropdown it will display like normal dropdown mode. |
| 4 | **onValueChange :** The callback function that will get called when the item from the picker is selected. The parameters available are itemValue i.e the actual value selected and itemPosition i.e the index position of the item. |
| 5 | **selectedValue :** The selected value from the picker. |

# Example

```
import { Picker } from "@react-native-picker/picker";
import React, { useState } from "react";
import { Alert, Image, Modal, Text, TextInput, View, Pressable, ScrollView } from
"react-native";

const ModelPicker = () => {

   const [modalVisible, setModalVisible] = useState(false);
   const [Enable, setEnable] = useState("Commercial");

      return (
      <View style={{ flex: 1, justifyContent: "center", alignItems: "center", marginTop:
22 }}>
         <Modal animationType="slide" transparent={false} visible={modalVisible}
onRequestClose={() => { Alert.alert("Modal has been closed.");
setModalVisible(!modalVisible); }}>
            {/* Modal Form  Start*/}
            <View>
               <ScrollView >
                  <Pressable onPress={() => setModalVisible(!modalVisible)} >
                     <Image source={require('./assests1/close.png')} style={{ height: 50,
width: 50, left: 320 }} />
                  </Pressable>
                  <TextInput style={{ borderColor: 'gray', fontSize: 25, borderWidth: 1,
width: 300, left: 25, top: 10, margin: 15, padding: 15, height: 60, borderRadius: 10 }}
placeholderTextColor="gray" placeholder="Property Name *" />
                  <View style={{ borderWidth: 1, fontSize: 25, borderColor: 'gray',
fontSize: 25, width: 300, left: 25, margin: 15, height: 60, borderRadius: 10 }}>
                     <Picker selectedValue={Enable} onValueChange={(itemValue) =>
setEnable(itemValue)}>
                        <Picker.Item label="Commercial" value="Commercial" />
                        <Picker.Item label="Residential" value="Residential" />
                        <Picker.Item label="Mixed use" value="Mixed use" />
                        <Picker.Item label="Industrial" value="Industrial" />
                        <Picker.Item label="Agriculture" value="Agriculture" />
                        <Picker.Item label="Special purpose" value="Special purpose" />
                     </Picker>
                  </View>
               </ScrollView>
            </View>
         </Modal>
         {/*  Modal Form end */}
         <Pressable
```

```
            style={{ borderRadius: 20, padding: 10, elevation: 2, backgroundColor:
"#2196F3", }}
            onPress={() => setModalVisible(true)} >
            <Text style={{ fontSize: 25 }}>Open to Modal</Text>
          </Pressable>
        </View>
    );
}
export default ModelPicker;
```

# React Native Drawer Navigation

- React Native Drawer Navigation is an UI panel which displays the app's navigation menu. By default it is hidden when not in use, but it appears when the user swipes a finger from the edge of the screen or when the user touches at the top of the drawer icon added at the app bar.
- React component that wraps the platform DrawerLayout (Android only). The Drawer (typically used for navigation) is rendered with renderNavigationView and direct children are the main view (where your content goes).
- React Native Drawer Navigation imports the **createDrawerNavigator** from the **react-navigation** library.

```
import { createDrawerNavigator } from 'react-navigation'
```

## Library Install :

```
npm install --save react-native-drawer
```

# Props of Drawer :

| | |
|---|---|
| drawerBackgroundColor | drawerLockMode |
| drawerPosition | keyboardDismissMode |
| onDrawerClose | onDrawerOpen |
| onDrawerSlide | onDrawerStateChanged |
| onDrawerStateChanged | statusBarBackgroundColor |

# Example :

```
import * as React from 'react';
import { createDrawerNavigator } from '@react-navigation/drawer';
import { NavigationContainer } from '@react-navigation/native';
import ScreenFirst from './ScreenFirst';
import SignUp from './SignUp';

const DrawerDemo = () => {

   const Drawer = createDrawerNavigator();

   return (
      <NavigationContainer independent={true}>
         <Drawer.Navigator  initialRouteName='ScreenFirst' dr>
            <Drawer.Screen name="ScreenFirst" component={ScreenFirst} />
            <Drawer.Screen name="SignUp" component={SignUp} />
         </Drawer.Navigator>
      </NavigationContainer>
   )
}
export default DrawerDemo;
```

# React-native API

- API stands for **Application programming interface**.
- In react native there are 2 methods of **API**.
    - GET method
    - POST method

## ☐ Making requests :

In order to fetch content from an arbitrary URL, you can pass the URL to fetch:

### Example :

```
fetch('https://mywebsite.com/mydata.json');
```

### POST Method :

- In the post method we give requests and pass a parameter to this api.
- In this example red fonts pass the parameters.

### Request :

```
fetch('https://mywebsite.com/endpoint/', {
  method: 'POST',
  headers: {
    Accept: 'application/json',
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    firstParam: 'yourValue',
    secondParam: 'yourOtherValue'
  })
});
```

### GET Method :

- In this get method we pass the api link and give a response.

**Request** :

```
const getMoviesFromApi = () => {
  return fetch('https://reactnative.dev/movies.json')
    .then((response) => response.json())
    .then((json) => {
      return json.movies;
    })
    .catch((error) => {
      console.error(error);
    });
};
```

# Async Storage :

- Async Storage can only store string data, so in order to store object data you need to serialize it first.
- For data that can be serialized to JSON you can use JSON.stringify() when saving the data and JSON.parse() when loading the data

## Installation

```
npm install @react-native-async-storage/async-storage
```

### Storing data

- setItem() is used both to add a new data item (when no data for a given key exists), and to modify an existing item (when previous data for a given key exists).

### Example :

```
const storeData = async (value) => {
   try{
            await AsyncStorage.setItem('@storage_Key', value)
    } catch (e) {
             console.log(e)
    }
  }
}
```

## Reading data

- getItem returns a promise that either resolves to stored value when data is found for a given key, or returns null otherwise.

### Example :

```
const getData = async () => {
  try {
    const value = await AsyncStorage.getItem('@storage_Key')
    if(value !== null) {
      Alert.alert(“ Success ”);
    }
  } catch(e) {
    console.log(e);
  }
}
```

# React-native Paper Library Installation Step :

1. Open a Terminal in your project's folder and run:

```
npm install react-native-paper
```

2. After sometime the process completes then your library is installed successfully.

# ActivityIndicator (Loading)

- Activity indicator is used to present progress of some activity in the app. It can be used as a drop-in for the ActivityIndicator shipped with React Native.

```
import React, { useState } from 'react';
import {  View, } from 'react-native';
import {  ActivityIndicator } from 'react-native-paper';

const ReactnativePaperDemo = () => {
    return (
        <View style={{ top: 25, }}>
          <ActivityIndicator animating={true} color={'red'} size={'large'} />
        </View>
    );
};
export default ReactnativePaperDemo;
```

# Badge

- Badges are small status descriptors for UI elements.
- A badge consists of a small circle, typically containing a number or other short set of characters, that appears in proximity to another object.

```
import * as React from 'react';
import { Badge } from 'react-native-paper';

const ReactnativePaperDemo = () => (
  <Badge size={40}>3</Badge>
);

export default ReactnativePaperDemo;
```

# Banner

- Banner displays a prominent message and related actions.

```
import React, { useState } from 'react';
import { View, } from 'react-native';
import { Image } from 'react-native';
import { Banner, Button } from 'react-native-paper';

const ReactnativePaperDemo = () => {
   const [visible, setVisible] = useState(false);

   return (
      <View style={{ top: 15 }}>
         <Banner visible={visible}
            actions={[{
               label: 'Fix it',
               onPress: () => setVisible(false),
            },
            {
               label: 'Learn more',
               onPress: () => setVisible(false),
            },]}
            icon={({ size }) => (<Image source={{ uri:
'https://avatars3.githubusercontent.com/u/17571969?s=400&v=4', }} style={{
width: size, height: size, }} />)}>
            It is a banner.
            It was created by Darshan Makhecha.
         </Banner>
         <View>
            <Button onPress={() => setVisible(true)} style={{ height: 50, width:
400, top: 250, }}>SHow Banner  </Button>
         </View>
      </View>
   );
}
export default ReactnativePaperDemo;
```

# BottomNavigation

- Bottom navigation provides quick navigation between top-level views of an app with a bottom navigation bar.
- It is primarily designed for use on mobile.

```
import React, { useState } from 'react';
import { BottomNavigation, Text } from 'react-native-paper';

const MusicRoute = () => <Text>Music</Text>;
const AlbumsRoute = () => <Text>Albums</Text>;
const RecentsRoute = () => <Text>Recents</Text>;

const ReactnativePaperDemo = () => {
  const [index, setIndex] = useState(0);
  const [routes] =useState([
    { key: 'music', title: 'Favorites', focusedIcon: 'heart', unfocusedIcon: 'heart-outline'},
    { key: 'albums', title: 'Albums', focusedIcon: 'album' },
    { key: 'recents', title: 'Recents', focusedIcon: 'history' },
     ]);

  const renderScene = BottomNavigation.SceneMap({
    music: MusicRoute,
    albums: AlbumsRoute,
    recents: RecentsRoute,
  });

  return (
    <BottomNavigation
      navigationState={{ index, routes }}
      onIndexChange={setIndex}
      renderScene={renderScene}
    />
  );
};

export default ReactnativePaperDemo;
```

# Button

- A button is a component that the user can press to trigger an action.

```
import React, { useState } from 'react';
import { View, } from 'react-native';
import { Button } from 'react-native-paper';

const ReactnativePaperDemo = () => {
    return (
        <View style={{ top: 15 }}>
            <Button mode="contained" onPress={() => console.log('Pressed')}> Press me
</Button>
        </View>
    );
}
export default ReactnativePaperDemo;
```

# Card

- A card is a sheet of material that serves as an entry point to more detailed information.

```
import React, { useState } from 'react';
import { View, } from 'react-native';
import { Button,Card, Paragraph, Title } from 'react-native-paper';

const ReactnativePaperDemo = () => {
    return (
        <View style={{ top: 15 }}>
          <Card>
              <Card.Content>
                 <Title>ABC Group</Title>
                 <Paragraph>This is card view {'\n'} </Paragraph>
              </Card.Content>
              <Card.Cover source={{ uri:
'https://i.picsum.photos/id/871/700/700.jpg?hmac=9Li72HkQHg9-fSOQXTzUpSDjxQa7
4ubTG5O8PAkieyY' }} />
              <Card.Actions>
              <Paragraph>This is a card view. This is a card view. This is a card view. This
is a card view. This is a. card view. This is a card view. This is card view.{'\n'}
</Paragraph>
```

```
            </Card.Actions>
            <View style={{flexDirection:'row'}}>
            <Button onPress={() => console.log("Ok Pressed")}>Ok</Button>
            <Button onPress={() => console.log("Cancel Pressed")}>Cancel</Button>
            </View>
          </Card>
        </View>
    );
}
export default ReactnativePaperDemo;
```

# Chip

- Chips can be used to display entities in small blocks.

```
import React, { useState } from 'react';
import { View, } from 'react-native';
import { Chip, } from 'react-native-paper';
const ReactnativePaperDemo = () => {
    return (
        <View style={{ top: 15 }}>
            <Chip icon="information" onPress={() => console.log('Pressed 1')}
style={{ top: 25 }} >Example Chip 1</Chip>
            <Chip icon="information" onPress={() => console.log('Pressed2')}
style={{ top: 35 }} >Example Chip 2</Chip>
            <Chip icon="information" onPress={() => console.log('Pressed3')}
style={{ top: 45 }} >Example Chip 3</Chip>
            <Chip icon="information" onPress={() => console.log('Pressed4')}
style={{ top: 55 }} >Example Chip 4</Chip>
            <Chip icon="information" onPress={() => console.log('Pressed5')}
style={{ top: 65 }} >Example Chip 5</Chip>
            <Chip icon="information" onPress={() => console.log('Pressed6')}
style={{ top: 75 }} >Example Chip 6</Chip>
            <Chip icon="information" onPress={() => console.log('Pressed7')}
style={{ top: 85 }} >Example Chip 7</Chip>
        </View>
    );
}
export default ReactnativePaperDemo;
```

# Dialog

- Dialogs inform users about a specific task and may contain critical information, require decisions, or involve multiple tasks. To render the Dialog above other components, you'll need to wrap it with the Portal component.

```
import React, { useState } from 'react';
import { View, } from 'react-native';
import { Button, Paragraph, Dialog, Portal, Provider } from 'react-native-paper';

const ReactnativePaperDemo = () => {
   const [visible, setVisible] = React.useState(false);

   const showDialog = () => setVisible(true);

   const hideDialog = () => setVisible(false);

   return (
     <Provider>
       <View>
         <Button onPress={showDialog}>Show Dialog</Button>
         <Portal>
           <Dialog visible={visible} onDismiss={hideDialog}>
             <Dialog.Title>Alert Box</Dialog.Title>
             <Dialog.Content>
               <Paragraph> This is a simple dialog. This is a simple dialog.
                           This is a simple dialog. This is a simple dialog.
               </Paragraph>
             </Dialog.Content>
             <Dialog.Actions>
               <Button onPress={hideDialog}>Done</Button>
             </Dialog.Actions>
           </Dialog>
         </Portal>
       </View>
     </Provider>
   );
}
export default ReactnativePaperDemo;
```

# Drawer

- A component to group content inside a navigation drawer.

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

```
import { Drawer } from 'react-native-paper';

const ReactnativePaperDemo = () => {
   const [active, setActive] = React.useState('');
   return (
      <View style={{padding:20}}>
       <Drawer.Section title="ABC DASHBOARD" >
    <Drawer.Item
      label="First Item"
      active={active === 'first'}
      onPress={() => setActive('first')}
    />
    <Drawer.Item
      label="Second Item"
      active={active === 'second'}
      onPress={() => setActive('second')}
    />
    <Drawer.Item
      label="Third Item"
      active={active === 'third'}
      onPress={() => setActive('third')}
    />
    <Drawer.Item
      label="Four Item"
      active={active === 'Four'}
      onPress={() => setActive('Four')}
    />
   </Drawer.Section>
      </View>
   );
}
export default ReactnativePaperDemo;
```

# FAB Group

- A floating action button represents the primary action in an application.

- A component to display a stack of FABs with related actions in a speed dial. To render the group above other components, you'll need to wrap it with the Portal component.

```
import React, { useState } from 'react';
import { FAB, Portal, Provider } from 'react-native-paper';

const ReactnativePaperDemo = () => {
   const [state, setState] = useState({ open: false });

  const onStateChange = ({ open }) => setState({ open });

  const { open } = state;

  return (
    <Provider>
      <Portal>
        <FAB.Group
          open={open}
          icon={open ? 'calendar-today' : 'plus'}
          actions={[
            {   icon: 'plus',
            label: 'Add',
                onPress: () => console.log('Pressed add') },
            {
              icon: 'star',
              label: 'Star',
              onPress: () => console.log('Pressed star'),
            },
            {
              icon: 'email',
              label: 'Email',
              onPress: () => console.log('Pressed email'),
            },
            {
              icon: 'bell',
              label: 'Remind',
              onPress: () => console.log('Pressed notifications'),
            },
          ]}
          onStateChange={onStateChange}
        />
      </Portal>
    </Provider>
     );
```

```
    };
export default ReactnativePaperDemo;
```

# HelperText

- Helper text is used in conjunction with input elements to provide additional hints for the user.

```
import React, { useState } from 'react';
import { View } from 'react-native';
import { HelperText, TextInput } from 'react-native-paper';

const ReactnativePaperDemo = () => {
   const [text, setText] =useState('');

   const onChangeText = text => setText(text);

  const check = () => {
    return !text.includes('@gmail.com');
  //  It means text ma @gmail.com include nathi check kare
  };

 return (
   <View>
     <TextInput label="Email" value={text} onChangeText={onChangeText} />
     <HelperText type="error" visible={check()}>
       Email address is invalid!
     </HelperText>
   </View>
  );
};
export default ReactnativePaperDemo;
```

# List

- A component used to display an expandable list item.

```
import React, { useState } from 'react';
import { View } from 'react-native';
import { Text } from 'react-native';
import { List } from 'react-native-paper';

const ReactnativePaperDemo = () => {
   const [expanded, setExpanded] = useState(true);

   return (
      <View style={{ backgroundColor: 'lightgray', flex: 1 }} >
         <Text style={{ fontSize: 25, textAlign: 'center' }}>LIST DEMO</Text>
         <List.Section>
            <List.Accordion title="FIRST LIST">
               <List.Item title="First item" />
               <List.Item title="Second item" />
            </List.Accordion>

            <List.Accordion title="Second LIST">
               <List.Item title="First item" />
               <List.Item title="Second item" />
            </List.Accordion>
         </List.Section>
      </View>
   );
};

export default ReactnativePaperDemo;
```

# Menu

- Menus display a list of choices on temporary elevated surfaces.
- Their placement varies based on the element that opens them.

```
import React, { useState } from 'react';
import { View } from 'react-native';
import { Button, Menu, Divider, Provider } from 'react-native-paper';
```

```
const ReactnativePaperDemo = () => {
 const [visible, setVisible] = useState(false);

 const openMenu = () => setVisible(true);
 const closeMenu = () => setVisible(false);

 return (
   <Provider>
     <View
       style={{
         paddingTop: 50,
         flexDirection: 'row',
         justifyContent: 'center',
       }}>
       <Menu
         visible={visible}
         onDismiss={closeMenu}
         anchor={<Button onPress={openMenu}>Show menu</Button>}>
         <Menu.Item onPress={() => {console.log('Item 1')}} title="Item 1" />
         <Menu.Item onPress={() => {console.log('Item 2')}} title="Item 2" />
         <Divider />
         <Menu.Item onPress={() => {console.log('Item 3')}} title="Item 3" />
       </Menu>
     </View>
   </Provider>
 );
};
export default ReactnativePaperDemo;
```

# Modal

- The Modal component is a simple way to present content above an enclosing view. To render the Modal above other components, you'll need to wrap it with the Portal component.

```
import React, { useState } from 'react';
import { View } from 'react-native';
import { Modal, Portal, Text, Button, Provider } from 'react-native-paper';
```

```
const ReactnativePaperDemo = () => {
 const [visible, setVisible] = React.useState(false);

  const showModal = () => setVisible(true);
  const hideModal = () => setVisible(false);
  const containerStyle = {backgroundColor: 'white', padding: 20};

  return (
    <Provider>
      <Portal>
        <Modal visible={visible} onDismiss={hideModal}
contentContainerStyle={containerStyle}>
          <Text>Example Modal. {'\n'}{'\n'}Click outside this area to dismiss.</Text>
        </Modal>
      </Portal>
      <Button style={{marginTop: 30}} onPress={showModal}>
        Show Modal
      </Button>
    </Provider>
  );
};
export default ReactnativePaperDemo;
```

# ProgressBar

- Progress bar is an indicator used to present progress of some activity in the app.

```
import * as React from 'react';
import { ProgressBar } from 'react-native-paper';
const MyComponent = () => (
 <ProgressBar progress={0.5} color={'red'} />
);
export default MyComponent;
```

# Portal.Host

- Portal host renders all of its children Portal elements. For example, you can wrap a screen in Portal.Host to render items above the screen.
- If you're using the Provider component, it already includes Portal.Host.

```
import React, { useState } from 'react';
import { View,Text } from 'react-native';
import { Portal } from 'react-native-paper';

const ReactnativePaperDemo = () => {

<Portal.Host>
   <Text>Content of the app</Text>
 </Portal.Host>

};
export default ReactnativePaperDemo;
```

# Radio-Button

- Radio buttons allow the selection of a single option from a set.

```
import React, { useState } from 'react';
import { View,Text } from 'react-native';
import { RadioButton } from 'react-native-paper';

const ReactnativePaperDemo = () => {
 const [checked, setChecked] = useState('first');
return (
    <View>
      <RadioButton
        value="first"
        status={ checked === 'first' ? 'checked' : 'unchecked' }
        onPress={() => setChecked('first')} />
      <RadioButton
        value="second"
        status={ checked === 'second' ? 'checked' : 'unchecked' }
        onPress={() => setChecked('second')} />
    </View>
)    };
export default ReactnativePaperDemo;
```

# Searchbar

- Searchbar is a simple input box where users can type search queries.

```
import React, { useState } from 'react';
import { View,Text } from 'react-native';
import { Searchbar } from 'react-native-paper';

const ReactnativePaperDemo = () => {

 const [searchQuery, setSearchQuery] =useState('');
 const onChangeSearch = query => setSearchQuery(query);

  return (
    <Searchbar
      placeholder="Search"
      onChangeText={onChangeSearch}
      value={searchQuery}
    />
  );
}
export default ReactnativePaperDemo;
```

# Snackbar

- Snackbars provide brief feedback about an operation through a message at the bottom of the screen.
- Snackbar by default uses onSurface color from the theme.

```
import React, { useState } from 'react';
import { View, StyleSheet } from 'react-native';
import { Button, Snackbar } from 'react-native-paper';
const ReactnativePaperDemo = () => {

  const [visible, setVisible] = React.useState(false);
  const onToggleSnackBar = () => setVisible(!visible);
  const onDismissSnackBar = () => setVisible(false);

  return (
    <View style={styles.container}>
      <Button onPress={onToggleSnackBar}>{visible ? 'Hide' : 'Show'}</Button>
      <Snackbar
```

```
      visible={visible}
      onDismiss={onDismissSnackBar}
      action={{
        label: 'Undo',
        onPress: () => {
          console.log('Snackbar activate');
        },
      }}>
      Hey there! I'm a Snackbar.
    </Snackbar>
  </View>
);
}
export default ReactnativePaperDemo;
```

# Surface

- Surface is a basic container that can give depth to an element with elevation shadow.
- On a dark theme with adaptive mode, the surface is constructed by also placing a semi-transparent white overlay over a component surface.

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

const ReactnativePaperDemo = () => {
return(
 <Surface style={styles.surface} elevation={4}>
     <Text>Surface</Text>
  </Surface>
)}
export default ReactnativePaperDemo;
const styles = StyleSheet.create({
  surface: {
    padding: 8,
    height: 80,
    width: 80,
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

# Switch

- Switch is a visual toggle between two mutually exclusive states — on and off.

```
import React, { useState } from 'react';
import { View, StyleSheet } from 'react-native';
import { Switch } from 'react-native-paper';

const ReactnativePaperDemo = () => {
 const [isSwitchOn, setIsSwitchOn] = React.useState(false);
 const onToggleSwitch = () => setIsSwitchOn(!isSwitchOn);
 return <Switch value={isSwitchOn} onValueChange={onToggleSwitch} />;
}
export default ReactnativePaperDemo;
```

# TextInput

- A component to allow users to input text.

```
import React, { useState } from 'react';
import { View, StyleSheet } from 'react-native';
import { TextInput } from 'react-native-paper';

const ReactnativePaperDemo = () => {

const [text, setText] = useState("");
  return (
    <TextInput
      label="Email"
      value={text}
      onChangeText={text => setText(text)}
    />
  );
}
export default ReactnativePaperDemo;
```

# ToggleButton

- Toggle buttons can be used to group related options. To emphasize groups of related toggle buttons, a group should share a common container.

```
import React, { useState } from 'react';
import { View } from 'react-native';
import { ToggleButton } from 'react-native-paper';

const ReactnativePaperDemo = () => {
   const [status, setStatus] =useState('checked');
   const [status1, setStatus1] =useState();

   const onButtonToggle = value => {
      setStatus(status == 'checked' ? 'unchecked' : 'checked');
      setStatus1(status1 == 'checked' ? 'unchecked' : 'checked');
   };
   return (
      <View>
         <View style={{ height: 500, width: 800, left: 52 }}>
            <ToggleButton icon="bluetooth" value="bluetooth" status={status}
onPress={onButtonToggle} size={40}  />

         </View>
         <View style={{ height: 500, width: 800, left: 52 }}>
            <ToggleButton icon="camera" value="camera" status={status1}
onPress={onButtonToggle} size={40} />
         </View>
      </View>
   );
};

export default ReactnativePaperDemo;
```

# TouchableRipple

- A wrapper for views that should respond to touches

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


const ReactnativePaperDemo = () => {
```

```
return(
<TouchableRipple
   onPress={() => console.log('Pressed')}
   rippleColor="rgba(0, 0, 0, .32)">
   <Text>Press the text</Text>
 </TouchableRipple>
  );
}
export default ReactnativePaperDemo;
```

## Text

- Typography component showing styles complied with passed variant prop and supported by the type system.

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


const ReactnativePaperDemo = () => {
 return(
   <Text variant="displayLarge">Display Large</Text>
   <Text variant="displayMedium">Display Medium</Text>
   <Text variant="displaySmall">Display small</Text>
 );
}
export default ReactnativePaperDemo;
```

# Android Platform :

- Operating system
- Based Linux (Open source freely)
- Latest version : Android 12 (API 32)
- Different device with different resolution and custom UI
- **Always provide height and width of any view in percentage**
- Example => Activity, Fragment, Service, Broadcast Receiver , Notification, Database, Bluetooth, Wifi, Internet, etc.
- It is developed by Google and later the OHA (Open Handset Alliance).
- OHA was established on 5th November, 2007, led by Google.
- OHA is a consortium of 84 companies such as Google, Samsung, AKM, KDDI, Garmin, Teleca, Ebay, Intel etc.

## Activity :

- ➔ Screen / Page / UI / Main screen
- ➔ An activity represents a single screen with a user interface just like a window or frame of Java.
- ➔ The Activity class defines the following call backs i.e. events.

## Fragment :

- ➔ The fragment is the part of Activity which represents a portion of User Interface(UI) on the screen.
- ➔ Example : Sub screen of activity (Whatsapp tab screens)

## Service :

- ➔ Service is special component that facilitates an application to run in the background in order to perform long-running operation tasks
- ➔ The prime aim of a service is to ensure that the application remains active in the background so that the user can operate multiple applications at the same time.

## Broadcast Receiver :

- ➔ Data Receive from API
- ➔ Alarm Manager

## **Notification** :

- ➔ A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app.
- ➔ Users can tap the notification to open your app or take an action directly from the notification.

## **Database** :

- ➔ Android comes in with built in SQLite database implementation.
- ➔ SQLite is an open source SQL database that stores data to a text file on a device.
- ➔ It is embedded in android by default. So, there is no need to perform any database setup or administration task.

## **Permissions** :

- ➔ App permissions help support user privacy by protecting access to the following: Restricted data, such as system state and a user's contact information.
- ➔ Restricted actions, such as connecting to a paired device and recording audio
- ➔ Example :
    - ◆ Internet access check if internet is not there that return false

**<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />**

## **R.java file** :

- ➔ Android R. java is an auto-generated file by aapt that contains resource IDs for all the resources of res/ directory.
- ➔ AAPT stands for Android Asset Packaging Tool .
- ➔ If you create any component in the activity_main. xml file, id for the corresponding component is automatically created in this file.
- ➔ R. java is used to provide access to resources defined in XML files.

### **Manifest File** :

- ➔ The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.
- ➔ It is used by various Windows technologies for configuring and deploying software, including ClickOnce and the Common Language Runtime (CLR).
- ➔ All permission is written here.
- ➔ This file is a very required file.

# Image-Picker Library Installation Step :

- A React Native module that allows you to select a photo/video from the device library or camera.

```
npm i react-native-image-picker
```

- No permissions required (saveToPhotos requires permission).
- Import the library in the top.

```
import { launchCamera, launchImageLibrary} from 'react-native-image-picker';
```

- In android studio Manifest file gives Permissions for launch camera.

```
<uses-permission android:name="android.permission.CAMERA"/>
```

- Next the permission giving  Open App info in Mobile and giving camera permission in your Phone.
- After this code is executed

```
import React, { useCallback, useState } from 'react';
import { Alert, Button, Image, Pressable, Text, Touchable, View } from 'react-native';
import * as ImagePicker from 'react-native-image-picker';

const App = () => {
  const [response, setResponse] = useState();

  const type = 'capture2';
  const options = {
    saveToPhotos: true,
    mediaType: 'photo',
    includeBase64: false,
  }

  const btnClick = ((type, options) => {

    if (type == 'capture') {
      ImagePicker.launchCamera(options, setResponse);
    }else {
      ImagePicker.launchImageLibrary(options, setResponse);
      console.log("else " + JSON.stringify(response, null, 2));
    }
  });
  return (
    <View>
      <Text style={{ fontSize: 32, textAlign: 'center' }}> Image Picker Example</Text>
       <Pressable onPress={()=> {setResponse(undefined)}}>
        <Image source={require('./assets/close.png')}
style={{height:50,width:50,left:300,top:25}} />
      </Pressable>
      <View>
        <Image style={{ width: 200, height: 500, marginLeft: 130, height: 100, width: 100,
borderRadius:40,top:80 }}
         source={(response != undefined && response.assets!=undefined) ? { uri:
response.assets[0].uri } :require('./assets/defaultImage.png')} />
```

```
    </View>
    <View style={{ top: 150, height:50,width:150,left:100}}>
      <Button onPress={() => btnClick(type, options)} title="Choose Image" ></Button>
    </View>

  </View>
 )
}
export default App;
```

# Custom Listing :

## .map() :

- The map method technically gives you more flexibility to customize your list the way you want with the React Native ScrollView component but it loads all the items every time which can be expensive.
- map() calls a function once for each element in an array.
- map() does not execute the function for empty elements.
- map() does not change the original array.

### Example

```
import React from "react";
import { Text, View, StyleSheet, ScrollView } from "react-native";
import { Divider } from "react-native-paper";

const persons = [
   {
      id: "1",
      name: "Ernest Green",
   },
   {
      id: "2",
      name: "Winston Orn",
   },
   {
      id: "3",
      name: "Carlton Collins",
   },
   {
```

```
        id: "4",
        name: "Malcolm Labadie",
    },
    {
        id: "5",
        name: "Michelle Dare",
    },
    {
        id: "6",
        name: "Carlton Zieme",
    },
    {
        id: "7",
        name: "Jessie Dickinson",
    },
    {
        id: "8",
        name: "Julian Gulgowski",
    },
]
const Custom_map = () => {
    return (
        <View style={styles.container}>
            <ScrollView showsHorizontalScrollIndicator={false}
showsVerticalScrollIndicator={false}>
                <View>
                    <View style={{flexDirection:'row'}}>
                        <Text style={{ fontWeight: 'bold' ,fontSize:25,left:20}}>ID</Text>
                        <Text style={{ fontWeight: 'bold' ,fontSize:25,left:80}}>Name</Text>
                        <Divider />
                        </View>
                        {persons.map((person) => {
                            return (
                                <View style={{ flexDirection: "row" }}>
                                    <Text style={styles.item}>{person.id}</Text>
                                    <Text style={styles.item}>{person.name}</Text>
                                </View>
                            );
                        })}
                </View>
            </ScrollView>
        </View>
    );
}
export default Custom_map;
const styles = StyleSheet.create({
```

```
    container: {
        padding: 50,
    },
    item: {
        padding: 20,
        fontSize: 20,
        marginTop: 5,
        color: 'blue',
    }
});
```

# Android Components :

## Android Permission

- PermissionsAndroid provides access to Android M's new permissions model.
- The so-called "normal" permissions are granted by default when the application is installed as long as they appear in AndroidManifest.xml.
- However, "dangerous" permissions require a dialog prompt.
- You should use this module for those permissions.

1. Go to manifest file which is located your android/app/src/main give Permission to access your camera and rebuild gradle.
   Permission code :
   <uses-permission android:name="android.permission.CAMERA"/>
2. Go to your file and code apply which is given below.

```
const granted = await PermissionsAndroid.request(
PermissionsAndroid.PERMISSIONS.CAMERA, {
      title: " App camera Permission",
      message: " App needs access to your camera " ,
      buttonNeutral: "Ask Me Later",
      buttonNegative: "Cancel",
      buttonPositive: "OK"
} );
```

**Example** :

```
import React from "react";
import { Button, PermissionsAndroid, StyleSheet, Text, View } from "react-native";

const requestContact = () => {
  try {
    const granted = await PermissionsAndroid.request(
      PermissionsAndroid.PERMISSIONS.READ_CONTACTS,
      {
        title: " App Contacts Permission",
        message:
          " App needs access to your Contacts " +
          "so you can take remainder.",
        buttonNeutral: "Ask Me Later",
        buttonNegative: "Cancel",
        buttonPositive: "OK"
      }
    );
    if (granted == PermissionsAndroid.RESULTS.GRANTED) {
      console.log("You can use the calendar");
    } else {
      console.log("calendar permission denied");
    }
  } catch (error) {
    console.log(error);
  }
};

const Permission = () => (
  <View style={styles.container}>
    <Text style={styles.item}>Try permissions</Text>
    <Button title="contact" onPress={requestContact()} />
  </View>
);

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: "center",
        backgroundColor: "#ecf0f1",
    padding: 8
  },
  item: {
    margin: 24,
```

```
    fontSize: 18,
    fontWeight: "bold",
    textAlign: "center"
  }
});

export default Permission;
```

## Android Toast

- A toast provides simple feedback about an operation in a small popup.
- It only fills the amount of space required for the message and the current activity remains visible and interactive.
- Toasts automatically disappear after a timeout.

## Example :

```
import React from "react";
import { View, StyleSheet, ToastAndroid, Button } from "react-native";

const App = () => {
  const showToast = () => {
    ToastAndroid.show("This is Toast ", ToastAndroid.SHORT);
  };

  return (
    <View style={styles.container}>
      <Button title="Toggle Toast" onPress={() => showToast()} />
    </View>

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: "center",
    backgroundColor: "#888888",
    padding: 8
  }
});

export default App;
```

# Android DrawerLayout

- Android Navigation Drawer is a sliding left menu that is used to display the important links in the application.
- Navigation drawer makes it easy to navigate to and fro between those links. It's not visible by default and it needs to open either by sliding from left or clicking its icon in the ActionBar.

## Example :

```
import React, { useRef, useState } from 'react';
import { Button, DrawerLayoutAndroid, Text, View } from 'react-native';

const DrawerAndroid = () => {
  const drawer = useRef(null);
  const [drawerPosition, setDrawerPosition] = useState("left");

  const changeDrawerPosition = () => {
    if (drawerPosition == "left") {
      setDrawerPosition("right");
    }
    else {
      setDrawerPosition("left");
    }
  };

  const navigationView = () => {
    return (
      <View style={{ flex: 1, height: '100%', width: '100%', alignItems: "center",
justifyContent: "center", padding: 16, backgroundColor: "#ecf0f1" }}>
        <Text style={{ padding: 16, fontSize: 15, textAlign: "center", color: 'red' }}> I am
drawer </Text>
        <Button title="Close drawer" onPress={() => drawer.current.closeDrawer()} />
      </View>
    )
  }
  return (
    <DrawerLayoutAndroid
      ref={drawer}
      drawerWidth={300}
      drawerPosition={drawerPosition}
      renderNavigationView={navigationView}>
      <View style={{
        flex: 1,
```

```
        alignItems: "center",
        justifyContent: "center",
        padding: 16, }}>
        <Text style={{
          padding: 16,
          fontSize: 15,
          textAlign: "center" }}>
          Drawer on the {drawerPosition} !
        </Text>
        <Button
          title="Change Drawer Position"
          onPress={() => changeDrawerPosition()}
        />
        <Text style={{ padding: 16, fontSize: 15, textAlign: "center" }}>
          Swipe from the side or press the button below to see it!
        </Text>
        <Button
          title="Open drawer"
          onPress={() => drawer.current.openDrawer()}
        />
      </View>
    </DrawerLayoutAndroid>
  )
}
export default DrawerAndroid;
```

## Android Statusbar

- Status bar (or notification bar) is an interface element at the top of the screen on Android devices that displays the notification icons, minimized notifications, battery information, device time, and other system status details.

**Example** :

```
import React, { useState } from "react";
import { Button, SafeAreaView, StatusBar, Text, View } from 'react-native';

const StatusBarDemo = () => {

    const styles = ['dark-content', 'light-content'];
    const [hidden, setHidden] = useState(true);
    const [statusBarStyle, setStatusBarStyle] = useState(styles[0]);
```

```
    const changeStatusBarVisibility = () => setHidden(!hidden);

    const changeStatusBarStyle = () => {
        const styleId = styles.indexOf(statusBarStyle) + 1;
        if (styleId == styles.length) {
            setStatusBarStyle(styles[0]);
        } else {
            setStatusBarStyle(styles[styleId]);
        }
    };
    return (
        <SafeAreaView style={{ flex: 1, justifyContent: 'center', backgroundColor:
'#ECF0F1' }}>
            <StatusBar animated={false} backgroundColor="#ECF0F1"
barStyle={statusBarStyle} hidden={hidden} />
            <Text style={{ textAlign: 'center', marginBottom: 8 }}> StatusBar Visibility:{'\n'}
{hidden ? 'Hidden' : 'Visible'} </Text>
            <Text style={{ textAlign: 'center', marginBottom: 8 }}> StatusBar Style:{'\n'}
{statusBarStyle} </Text>
            <View style={{ padding: 10 }}>
                <Button title="Toggle StatusBar" onPress={changeStatusBarVisibility} />
                <Button title="Change StatusBar Style" onPress={changeStatusBarStyle} />
            </View>
        </SafeAreaView>
    )
}
export default StatusBarDemo;
```

## Android RefreshControl

- This component is used inside a ScrollView or ListView to add pull to refresh functionality.

```
import React, {useCallback, useState}from "react";
import { RefreshControl, SafeAreaView, ScrollView, StyleSheet, Text } from
'react-native';

const wait = (timeout) => {
    return new Promise(resolve => setTimeout(resolve, timeout));
}
const RefreshPage = () => {
    const [refreshing, setRefreshing] = useState(false);
```

```
    const onRefresh = useCallback(() => {
        setRefreshing(true);
        wait(2000).then(() => setRefreshing(false));
    }, []);
    return (
        <SafeAreaView style={styles.container}>
            <ScrollView contentContainerStyle={styles.scrollView}
                    refreshControl={ <RefreshControl refreshing={refreshing}
onRefresh={onRefresh} /> } >
                <Text>Pull down to see RefreshControl indicator</Text>
            </ScrollView>
        </SafeAreaView>
    )
}
const styles = StyleSheet.create({
    container: {
      flex: 1,
    },
    scrollView: {
      flex: 1,
      backgroundColor: 'pink',
      alignItems: 'center',
      justifyContent: 'center',
    },
  });
export default RefreshPage;
```

# Animation :

- Animations are very important to create a great user experience. Stationary objects must overcome inertia as they start moving.
- Objects in motion have momentum and rarely come to a stop immediately.
- Animations allow you to convey physically believable motion in your interface.
- React Native provides two complementary animation systems: **Animated** for granular and interactive control of specific values, and **LayoutAnimation** for animated global layout transactions.

## Overview

- There are two value types you can use with Animated:
  - Animated.Value() for single values.
  - Animated.ValueXY() for vectors.
- Animated.Value can bind to style properties or other props, and can be interpolated as well.
- A single Animated.Value can drive any number of properties.
- Animated.decay() starts with an initial velocity and gradually slows to a stop.
- Animated.spring() provides a basic spring physics model.
- Animated.timing() animates a value over time using easing functions.

## Working with animations

- Animations are started by calling start() on your animation.
- **start()** takes a completion callback that will be called when the animation is done.
- If the animation finishes running normally, the completion callback will be invoked with **{finished: true}**.
- If the animation is done because stop() was called on it before it could finish

**Example :**

```
Animated.timing({

        // Code here

}).start();
```

## Animatable components

- Animated.Image
- Animated.ScrollView
- Animated.Text
- Animated.View
- Animated.FlatList
- Animated.SectionList

## Animated.timing()

- Timing animation refers to how long an action takes from beginning to end.
- The functions of timing are to create movement that obeys the laws of physics, and to add interest to your animations.
- Timing can be implemented by applying weight, scaling properties, and emotion.
- Config is an object that may have the following options:

➔ **duration:** Length of animation (milliseconds). Default 500.
➔ **easing:** Easing function to define curve. Default is Easing.inOut.
➔ **delay:** Start the animation after delay (milliseconds). Default 0.
➔ **isInteraction:** Whether or not this animation creates an "interaction handle" on the InteractionManager. Default true.
➔ **useNativeDriver:** Uses the native driver when true. Default false.

**Example :**

static timing(value, config);

## Animated.spring()

- Animated. spring **defines a way to transition an Animated**.
- Value based on tension and friction of a spring.
- The tension defines how much energy the spring has, and the friction defines how quickly that energy will dissipate.
- The friction/tension or bounciness/speed options match the spring model in Facebook Pop, Rebound, and Origami.

➔ **friction**: Controls "bounciness"/overshoot. Default 7.
➔ **tension**: Controls speed. Default 40.
➔ **speed**: Controls speed of the animation. Default 12.
➔ **bounciness**: Controls bounciness. Default 8.

**Example :**

static spring(value, config);

## Animated.Loop()

- Looping an animation causes it to repeat.
- Loops a given animation continuously, so that each time it reaches the end, it resets and begins again from the start.
- This is used when an animation needs to keep repeating.

- Config is an object that may have the following options:
  - ➔ **iterations**: Number of times the animation should loop.
    - Default -1 (infinite).

## Animated.sequence()

- Starts an array of animations in order, waiting for each to complete before starting the next.
- If the current running animation is stopped, no following animations will be started.

## Example

```
import React, { useRef,useEffect } from "react";
import { Text, View, StyleSheet, Animated, Image, ImageBackground, } from
"react-native";

const AnimationScreen = () => {

  useEffect(() => {
    _start();
  });
  useEffect(() => {
    Animated.spring(
      animation,
      {
        toValue: 100,
        speed: 0.1,
        bounciness: 1
      }
    ).start();
  });
  useEffect(() => {
    spinValue.setValue(0);
    Animated.timing(
      spinValue, {
      toValue: 1,
      duration: 8000,
    }
    ).start();
  });
    // plane
  useEffect(() => {
    Animated.loop(
      Animated.sequence([
        Animated.timing(anim.current, {
          toValue: -5,
          duration: 150,
        }),
        Animated.timing(anim.current, {
          toValue: 2,
          duration: 150,
```

```
      }),
    ]),
    { iterations: 5 }
  ).start();
}, []);
const slideUpValue = new Animated.Value(0);
const animation = new Animated.Value(0);
const spinValue = new Animated.Value(0);
const anim = useRef(new Animated.Value(0));

const _start = () => {
  Animated.timing(slideUpValue, {
    toValue: 1,
    duration: 5000,
  }).start();
};

const translateX = slideUpValue.interpolate({
  inputRange: [0, 1],
  outputRange: [320, 80]
});
const spin = spinValue.interpolate({
  inputRange: [0, 1],
  outputRange: ['0deg', '360deg']
})
return (
  <View style={styles.container}>

    <ImageBackground source={require('./assets/CartoonGarden.webp')}
style={{ flex: 1 }}>
    <Animated.View style={{ transform: [{ translateX: anim.current }] }}>
      <Image source={require('./assets/plane.png')} style={{ height: 100, width:
100, left: 250 }} />
      </Animated.View>
      <Animated.Image style={{ width: 120, height: 100, top: 200, left: 20,
transform: [{ rotate: spin }] }}
        source={{ uri: 'https://freepngimg.com/download/sun/23611-7-sun.png' }}
/>
      <Animated.View style={{
        top: 40, left: 100, borderRadius: 20, backgroundColor: 'orange', width:
150, height: 100,
        transform: [{ translateY: animation }] }}>
```

```
          <Text style={{ fontSize: 22, color: 'white', top: 25, alignSelf: 'center'
}}>SpellHero App</Text>
        </Animated.View>
        <Image source={require('./assets/play.png')} style={{ height: 80, width: 80,
left: 160, top: 250 }} />
        <Animated.Image
          style={{ width: 60, height: 60, top: 300,
            transform: [{
              translateX: translateX
            }, { rotate: spin }],
          }}
          source={require('./assets/ball.png')}
        />
        <Image source={require('./assets/dog.png')} style={{ height: 100, width:
100, top: 220, }} />
      </ImageBackground>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#FFF",
  },
  btn: {
    backgroundColor: "#480032",
    width: 100,
    height: 40,
    padding: 3,
    justifyContent: "center",
    alignSelf: 'center',
    borderRadius: 6,
    marginTop: 29
  },
  text: {
    fontSize: 20,
    color: "#fff",
    fontWeight: "bold",
    textAlign: "center"
  },
  textBtn: {
```

```
    color: "#f4f4f4",
    fontWeight: "bold",
    textAlign: "center"
  }
});

export default AnimationScreen;
```

# SQL Database :

- React-native uses database is SQLite Database.
- SQLite is the most used database engine in the world.
- SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day.
- Install react-native-sqlite-storage dependency to use SQLite

```
npm i react-native-sqlite-storage
```

- After the installation of this library You just have to import the library like this:

```
import { openDatabase } from 'react-native-sqlite-storage';
```

- and open the database using

```
var db = openDatabase({ name: 'UserDatabase.db' });
```

- Now, whenever you need to make some database calls you can use the db variable to execute the database query.

## Example :

```
db.transaction(function(txn)
{
        txn.executeSql(
                query,  //Query to execute as prepared statement
                argsToBePassed[...],  //Argument to pass for the prepared statement
                function(tx, res) {...}  //Callback function to handle the result
        );
});
```

In this example, I will make a **HomeScreen** with the option to go to other screens like

- **RegisterUser**: To Register the User. (Create/Insert)
- **ViewAllUser**: To View All Users. (Read)
- **ViewUser**: To View Single Users By Id. (Read)
- **UpdateUser**: To Update the User.(Update)
- **DeleteUser**: To Delete the User.

## Example

### App.js

```
import * as React from 'react';
import { Button, View, Text } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';

import HomeScreen from 'HomeScreen';
import RegisterUser from 'RegisterUser';
import UpdateUser from 'UpdateUser';
import ViewUser from 'ViewUser';
import ViewAllUser from 'ViewAllUser';
import DeleteUser from 'DeleteUser';

const Stack = createStackNavigator();

const App = () => {
```

```
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="HomeScreen" screenOptions={{ headerShown:
false }}>
        <Stack.Screen name="HomeScreen"component={HomeScreen} />
        <Stack.Screen name="'RegisterUser'"component={'RegisterUser'} />
        <Stack.Screen name="'UpdateUser'"component={'UpdateUser'} />
        <Stack.Screen name="'ViewUser'"component={'ViewUser'} />
        <Stack.Screen name="'ViewAllUser'"component={'ViewAllUser'} />
        <Stack.Screen name="'DeleteUser'"component={'DeleteUser'} />
      </Stack.Navigator>
    </NavigationContainer>
  );
};

export default App;
```

# HomeScreen.js

```
import React, { useEffect } from 'react';
import { View, Text, SafeAreaView } from 'react-native';
import { openDatabase } from 'react-native-sqlite-storage';

var db = openDatabase({ name: 'UserDatabase.db' });

const HomeScreen = ({ navigation }) => {
  useEffect(() => {
    db.transaction(function (txn) {
  txn.executeSql(
            'CREATE TABLE IF NOT EXISTS table_user(user_id INTEGER PRIMARY
KEY AUTOINCREMENT, user_name VARCHAR(20), user_contact INT(10),
user_address VARCHAR(255))',
            []
          );
        );
    });
  }, []);

return (
    <SafeAreaView style={{ flex: 1 }}>
      <View style={{ flex: 1, backgroundColor: 'white' }}>
 <ImageBackground source={{ uri:
"https://i.pinimg.com/736x/db/89/07/db8907208b387dedf183982486f262f8.jpg" }}
```

```
resizeMode="cover" style={{ flex: 1, justifyContent: "center" }}>
                <Text style={{ fontSize: 25, textAlign: 'center', color: 'black', fontWeight:
"bold" }}>
                    Example of SQLite Database
                </Text>
                <Text style={styles.txt} text="SQLite Example" />
                <TouchableOpacity style={styles.button} onPress={() =>
navigation.navigate('RegisterUser')}>
                    <Text style={{ color: 'white', fontSize: 20 }}> Register </Text>
                </TouchableOpacity>
                <TouchableOpacity style={styles.button} onPress={() =>
navigation.navigate('UpdateUser')}>
                    <Text style={{ color: 'white', fontSize: 20 }}> Update </Text>
                </TouchableOpacity>
                <TouchableOpacity style={styles.button} onPress={() =>
navigation.navigate('ViewUser')}>
                    <Text style={{ color: 'white', fontSize: 20 }}> User Details </Text>
                </TouchableOpacity>
                <TouchableOpacity style={styles.button} onPress={() =>
navigation.navigate('ViewAllUser')}>
                    <Text style={{ color: 'white', fontSize: 20 }}> All Users </Text>
                </TouchableOpacity>
                <TouchableOpacity style={styles.button} onPress={() =>
navigation.navigate('DeleteUser')}>
                    <Text style={{ color: 'white', fontSize: 20 }}> Delete </Text>
                </TouchableOpacity>
            </ImageBackground>
        </View>
    </SafeAreaView>
  )
}
const styles = StyleSheet.create({
  button: {
    alignItems: 'center',
    backgroundColor: '#f96487',
    padding: 10,
    marginTop: 16,
    marginLeft: 35,
    marginRight: 35,
  },
  txt: {
    color: '#111825',
    fontSize: 18,
    marginTop: 16,
    marginLeft: 35,
    marginRight: 35,
```

```
    }
});

export default HomeScreen;
```

# RegisterUser.js

```
import React, { useState } from 'react';
import { View, ScrollView, TextInput, Alert, SafeAreaView, Text, Pressable, StyleSheet,
} from 'react-native';
import { openDatabase } from 'react-native-sqlite-storage';

var db = openDatabase({ name: 'UserDatabase.db' });

const RegisterUser = ({ navigation }) => {
    const [userName, setUserName] = useState("");
    const [userContact, setUserContact] = useState("");
    const [userAddress, setUserAddress] = useState("");
    const register_user = () => {
        console.log("Details username, contact , Address " + userName, userContact,
userAddress);
        if (userName.length == 0) {
            Alert.alert("Please enter username");
            return;
        }
        else if (userContact.length != 10) {
            Alert.alert("Please enter mobile number");
            return;
        }
        else if (userAddress.length == 10) {
            Alert.alert("Please enter User Address");
            return;
        }
        db.transaction(function (tx) {
            tx.executeSql(
                'INSERT INTO table_user (user_name, user_contact, user_address)
VALUES (?,?,?)',
                [userName, userContact, userAddress],
                (results) => {
                    console.log('Row(s) Affected ', results.rowsAffected);
                    if (results.rowsAffected > 0) {
                        Alert.alert(
```

```
                        'Success',
                        'You are Registered Successfully',
                        [
                            {
                                text: 'Ok',
                                onPress: () => navigation.navigate('HomeScreen'),
                            },
                        ],
                        { cancelable: false }
                    );
                } else alert('Registration Failed');
            }
        );
    });
};

return (
    <View>
        <ScrollView>
            <View style={{}}>
                <Text style={{ fontSize: 25, fontWeight: 'bold', textAlign: "center"
}}>Register User</Text>
                <TextInput style={{ borderColor: 'black', fontSize: 25, borderWidth: 1,
borderRadius: 20, margin: 15, padding: 15, }} value={userName}
onChangeText={setUserName} placeholderTextColor="#003f5c" placeholder="Enter
Last Name" />
                <TextInput style={{ borderColor: 'black', fontSize: 25, borderWidth: 1,
borderRadius: 25, margin: 15, padding: 15, }} value={userContact}
onChangeText={setUserContact} placeholderTextColor="#003f5c"
keyboardType='numeric' placeholder="Enter Mobile" />
                <TextInput style={{ borderColor: 'black', fontSize: 25, textAlignVertical:
'top', padding: 10, borderWidth: 1, borderRadius: 25, margin: 15, padding: 15, }}
value={userAddress} onChangeText={setUserAddress}
placeholderTextColor="#003f5c" placeholder="Enter Address" maxLength={225}
numberOfLines={5} multiline={true} />
                <Pressable style={commonStyles.loginBtn1} onPress={() => {
register_user() }}>
                    <Text style={commonStyles.loginText1} >Submit</Text>
                </Pressable>
            </View>
        </ScrollView>
    </View>
)
}
export const commonStyles = StyleSheet.create({
    loginBtn1: {
```

```
        width: "90%",
        borderRadius: 50,
        height: 50,
        alignItems: "center",
        justifyContent: "center",
        marginTop: 20,
        marginLeft: 20,
        backgroundColor: "blue",
    },
    loginText1: {
        color: "white",
        fontSize: 20,
    }
})
export default RegisterUser;
```

# UpdateUser.js

```
import React, { useState } from 'react';
import { View, ScrollView,TextInput, Alert, Text, StyleSheet, Pressable, } from
'react-native';
import { openDatabase } from 'react-native-sqlite-storage';

var db = openDatabase({ name: 'UserDatabase.db' });
const UpdateUser = ({ navigation }) => {
    const [inputUserId, setInputUserId] = useState('');
    const [userName, setUserName] = useState('');
    const [userContact, setUserContact] = useState('');
    const [userAddress, setUserAddress] = useState('');

    let updateAllStates = (name, contact, address) => {
        setUserName(name);
        setUserContact(contact);
        setUserAddress(address);
    };
    let searchUser = () => {
        console.log(inputUserId);
        db.transaction((tx) => {
            tx.executeSql(
                'SELECT * FROM table_user where user_id = ?',
                [inputUserId],
                (results) => {
                    var len = results.rows.length;
```

```
                if (len > 0) {
                    let res = results.rows.item(0);
                    updateAllStates(
                        res.user_name,
                        res.user_contact,
                        res.user_address
                    );
                } else {
                    alert('No user found');
                    updateAllStates('', '', '');
                }
            }
        );
    });
};
let updateUser = () => {
    console.log(inputUserId, userName, userContact, userAddress);

    if (!inputUserId) {
        alert('Please fill User id');
        return;
    }
    if (!userName) {
        alert('Please fill name');
        return;
    }
    if (!userContact) {
        alert('Please fill Contact Number');
        return;
    }
    if (!userAddress) {
        alert('Please fill Address');
        return;
    }

    db.transaction((tx) => {
        tx.executeSql(
            'UPDATE table_user set user_name=?, user_contact=? , user_address=?
where user_id=?',
            [userName, userContact, userAddress, inputUserId],
            (results) => {
                console.log('Results', results.rowsAffected);
                if (results.rowsAffected > 0) {
                    Alert.alert(
                        'Success',
                        'User updated successfully',
```

```
                        [
                            {
                                text: 'Ok',
                                onPress: () => navigation.navigate('HomeScreen'),
                            },
                        ],
                        { cancelable: false }
                    );
                } else alert('Updation Failed');
            }
        );
    });
};
return (
    <View style={{ flex: 1 }}>
        <Text style={{ fontSize: 25, fontWeight: 'bold', textAlign: "center" }}>Update
User Page</Text>
        <ScrollView>
            <View>
                <TextInput style={{ borderColor: 'black', fontSize: 25, borderWidth: 1,
borderRadius: 20, margin: 15, padding: 15, }} value={inputUserId}
onChangeText={setInputUserId} placeholderTextColor="#003f5c" placeholder="Enter
User ID" />

                <Pressable style={commonStyles.loginBtn1} onPress={() => {
searchUser() }}>
                    <Text style={commonStyles.loginText1} >Search User</Text>
                </Pressable>
                <TextInput style={{ borderColor: 'black', fontSize: 25, borderWidth: 1,
borderRadius: 20, margin: 15, padding: 15, }}
                    value={userName}
                    onChangeText={
                        (userName) => setUserName(userName)
                    }
                    placeholderTextColor="#003f5c"
                    placeholder="Enter UserName" />
                <TextInput style={{ borderColor: 'black', fontSize: 25, borderWidth: 1,
borderRadius: 20, margin: 15, padding: 15, }}
                    value={'' + userContact}
                    onChangeText={
                        (userContact) => setUserContact(userContact)
                    } maxLength={10}
                    keyboardType="numeric"
                    placeholderTextColor="#003f5c"
                    placeholder="Enter Mobile Number" />
                <TextInput style={{ borderColor: 'black', textAlignVertical: 'top', padding:
```

```
10, fontSize: 25, borderWidth: 1, borderRadius: 20, margin: 15, padding: 15, }}
                    value={userAddress}
                    placeholder="Enter Address"
                    onChangeText={
                        (userAddress) => setUserAddress(userAddress)
                    }
                    maxLength={225}
                    numberOfLines={5}
                    multiline={true}
                    placeholderTextColor="#003f5c"
                />
                <Pressable style={commonStyles.loginBtn1} onPress={() => {
updateUser() }}>
                    <Text style={commonStyles.loginText1}> Update User</Text>
                </Pressable>
            </View>
        </ScrollView>
    </View>
  )
}
export const commonStyles = StyleSheet.create({

  loginBtn1: {
    width: "90%",
    borderRadius: 50,
    height: 50,
    alignItems: "center",
    justifyContent: "center",
    marginTop: 20,
    marginLeft: 20,
    backgroundColor: "red",
  },
  loginText1: {
    color: "white",
    fontSize: 20,
  }
 })
export default UpdateUser;
```

ViewUser.js

```
import React, { useState } from 'react';
import { Text, View, TextInput, Pressable, ScrollView, StyleSheet } from 'react-native';
import { openDatabase } from 'react-native-sqlite-storage';

var db = openDatabase({ name: 'UserDatabase.db' });
const ViewUser = () => {
   let [inputUserId, setInputUserId] = useState('');
   let [userData, setUserData] = useState({});

   let searchUser = () => {
      console.log(inputUserId);
      setUserData({});
      db.transaction((tx) => {
         tx.executeSql(
            'SELECT * FROM table_user where user_id = ?',
            [inputUserId],
            (tx, results) => {
               var len = results.rows.length;
               console.log('len', len);
               if (len > 0) {
                  setUserData(results.rows.item(0));
               } else {
                  alert('No user found');
               }
            }
         );
      });
   };
   return (
      <View style={{ flex: 1, backgroundColor: 'white' }}>
         <Text style={{ fontSize: 25, fontWeight: 'bold', textAlign: "center" }}> User
Details</Text>
         <ScrollView>
            <View>
               <TextInput style={{ borderColor: 'black', fontSize: 25, borderWidth: 1,
borderRadius: 20, margin: 15, padding: 15, }}
                  value={inputUserId}
                  onChangeText={setInputUserId}
                  placeholderTextColor="#003f5c"
                  placeholder="Enter User ID" />
               <Pressable style={commonStyles.loginBtn1} onPress={() => {
searchUser() }}>
                  <Text style={commonStyles.loginText1}> Search User</Text>
               </Pressable>
               <View style={{ marginLeft: 35, marginRight: 35, marginTop: 10 }}>
```

```
                    <Text>User Id: {userData.user_id}</Text>
                    <Text>User Name: {userData.user_name}</Text>
                    <Text>User Contact: {userData.user_contact}</Text>
                    <Text>User Address: {userData.user_address}</Text>
                </View>
            </View>
        </ScrollView>
    </View>
  )
}
export const commonStyles = StyleSheet.create({

  loginBtn1: {
    width: "90%",
    borderRadius: 50,
    height: 50,
    alignItems: "center",
    justifyContent: "center",
    marginTop: 20,
    marginLeft: 20,
    backgroundColor: "red",
  },
  loginText1: {
    color: "white",
    fontSize: 20,
  }
})
export default ViewUser;
```

## ViewAllUser.js

```
import React, { useState, useEffect } from 'react';
import { FlatList, Text, View, ScrollView } from 'react-native';
import { openDatabase } from 'react-native-sqlite-storage';

var db = openDatabase({ name: 'UserDatabase.db' });

const ViewAllUser = ({ navigation }) => {
  const [flatListItems, setFlatListItems] = useState([]);

  useEffect(() => {
    db.transaction((tx) => {
      tx.executeSql(
```

```
            'SELECT * FROM table_user',
            [],
            (results) => {
              var temp = [];
              for (let i = 0; i < results.rows.length; ++i)
                temp.push(results.rows.item(i));
              setFlatListItems(temp);
            }
          );
        });
    }, []);
    let listItemView = (item) => {
      return (
        <View
          key={item.user_id}
          style={{ backgroundColor: 'white', padding: 20 }}>
          <Text>Id: {item.user_id}</Text>
          <Text>Name: {item.user_name}</Text>
          <Text>Contact: {item.user_contact}</Text>
          <Text>Address: {item.user_address}</Text>
        </View>
      );
    };
    return (
      <View style={{ flex: 1, backgroundColor: 'white' }}>
        <Text style={{ fontSize: 25, fontWeight: 'bold', textAlign: "center" }}>All User
Details</Text>
        <ScrollView>
          <View>
            <FlatList
              data={flatListItems}
              keyExtractor={(index) => index.toString()}
              renderItem={({ item }) => listItemView(item)}
            />
          </View>
        </ScrollView>
      </View>
    )
}
export default ViewAllUser;
```

DeleteUser.js

```
import React, { useState } from 'react';
import { Text, View, Alert, ScrollView, TextInput, Pressable, StyleSheet } from 'react-native';
import { openDatabase } from 'react-native-sqlite-storage';

var db = openDatabase({ name: 'UserDatabase.db' });

const DeleteUser = ({ navigation }) => {
   const [inputUserId, setInputUserId] = useState("");

   let deleteUser = () => {
      db.transaction((tx) => {
         tx.executeSql(
            'DELETE FROM  table_user where user_id=?',
            [inputUserId],
            (results) => {
               console.log('Results', results.rowsAffected);
               if (results.rowsAffected > 0) {
                  Alert.alert(
                     'Success',
                     'User deleted successfully',
                     [
                        {
                           text: 'Ok',
                           onPress: () => navigation.navigate('HomeScreen'),
                        },
                     ],
                     { cancelable: false }
                  );
               } else {
                  alert('Please insert a valid User Id');
               }  }
         );
      });
 };
   return (
      <View style={{ flex: 1, backgroundColor: 'white' }}>
         <Text style={{ fontSize: 25, fontWeight: 'bold', textAlign: "center" }}>Delete User
</Text>
         <ScrollView>
            <View>
               <TextInput style={{ borderColor: 'black', fontSize: 25, borderWidth: 1,
borderRadius: 20, margin: 15, padding: 15, }} value={inputUserId}
onChangeText={setInputUserId} placeholderTextColor="#003f5c" placeholder="Enter User
ID" keyboardType='numeric' />
               <Pressable style={commonStyles.loginBtn1} onPress={() => { deleteUser() }}>
                  <Text style={commonStyles.loginText1} >Delete User</Text>
```

```
                </Pressable>
            </View>
        </ScrollView>
    </View>
  )}
export const commonStyles = StyleSheet.create({
  loginBtn1: {
    width: "90%",
    borderRadius: 50,
    height: 50,
    alignItems: "center",
    justifyContent: "center",
    marginTop: 20,
    marginLeft: 20,
    backgroundColor: "red",
  },
  loginText1: {
    color: "white",
    fontSize: 20,
  }
})
export default DeleteUser;
```

## Database listing with Network conflict

- In this program First of all Data shows API.
- The whole process of the program given below.
  - First open the application then check if your phone is connected to the internet or not.
  - If the internet connection is turned on then this alert display which says your phone is connected to the internet after pressing or clicking OK then you show some data which renders the API.
  - If the internet connection is turned off then this alert display which says your phone is Disconnected to the internet after pressing or clicking OK then you show data which displays local Database(SQLite).
  - When First time open the application then Insert record from API to local database then after all time update record or modify data.
  - If in any case some data is deleted from API then also delete local database but make sure your phone is connected to internet then it is possible to.

➔ How to check if your phone is connected to the Internet or not ?

- To check if your application is connected to the internet or not for that we download dependency.

```
npm i @react-native-community/netinfo
```

- After successfully downloading the dependency then import the library.

```
import NetInfo from '@react-native-community/netinfo';
```

- Then after the below code is added in your file.

```
NetInfo.fetch().then((state) => {
        console.log('Connection type', state.type);
        if (state.isConnected == true) {
            console.log('Is connected?', state.isConnected);
            Alert.alert(
                'Success',
                'You are now connect this application',
                [
                    {
                        text: 'Ok',
                    },
                ],
                { cancelable: false }
            );
        }
        else {
            Alert.alert(
                'OOPS ❌',
                'You are offline',
                [   {
                        text: 'Ok',
                    },  ],
                { cancelable: false }
            );
        }
    });
```

# Example

```
import React, { useEffect, useState } from "react";
import { View, Text, Alert, FlatList, ActivityIndicator, } from "react-native";
import NetInfo from '@react-native-community/netinfo';
import { openDatabase } from 'react-native-sqlite-storage';

var db = openDatabase({ name: 'UserDatabase.db' });

const DataBaseListing = () => {

    const [info, setinfo] = useState([])
    const [dataInfo, setdataInfo] = useState([])
    const [isLoading, setLoading] = useState(true);


    useEffect(() => {
        checkNetwork();  // check internet on or off
        setLoading(false);
    }, []);

    useEffect(() => {
        db.transaction(function (txn) {
            txn.executeSql(
                'CREATE TABLE IF NOT EXISTS ownerData(owner_id VARCHAR(20) PRIMARY KEY,
owner_name VARCHAR(20),owner_lastname VARCHAR(20))',
                []
            );
        });
    }, []);

     useEffect(() =>{
         db.transaction((tx) => {
             tx.executeSql(
                 'SELECT * FROM ownerData',
                 [],
                 (tx,results) => {
                     var temp = [];
                     for (let i = 0; i < results.rows.length; ++i)
                         temp.push(results.rows.item(i));
                     setdataInfo(temp);
                 }
             );
         });
     })
    const callAPI = () => {

        setLoading(true);
```

```
var myHeaders = new Headers();
myHeaders.append("Content-Type", "application/json");

var raw = JSON.stringify({
    "user_id": "1",
    "page_no": "1",
    "device_id": "48956264",
    "device_type": "android",
    "search": ""
});
var requestOptions = {
    method: 'POST',
    headers: myHeaders,
    body: raw,
};
fetch("http://rental.alakmalak.ca/api/owner_list.php", requestOptions)
    .then(response => response.json())
    .then(response => {
        info.length = 0;
        for (let i = 0; i < response.data.length; i++) {
            info.push({
                "owner_id": response.data[i].owner_id,
                "owner_name": response.data[i].owner_name,
                "owner_lastname": response.data[i].owner_lastname,
            })
            setLoading(false);
            setinfo(info);
                 db.transaction(function (tx) {
                    tx.executeSql(
                        "SELECT * from ownerData WHERE owner_id = ?", [response.data[i].owner_id],
                        (tx, results) => {
                            // console.log("Result123", + results);
                            console.log("Result", + results.rows.length);
                            if (results.rows.length > 0) {
                                db.transaction(function (tx) {
                                    console.log("update");
                                    tx.executeSql(
    'UPDATE ownerData set owner_name=?, owner_lastname=? where owner_id=?',
    [response.data[i].owner_name, response.data[i].owner_lastname, response.data[i].owner_id,],
                                        (results) => {
                                            console.log("Data Updated");
                                        }
                                    );
                                });
                            }
                            else {
```

```
                                db.transaction(function (tx) {
                                    console.log("INSERT");
                                    tx.executeSql(
                                'INSERT INTO ownerData (owner_id, owner_name, owner_lastname) VALUES (?,?,?)',
                            [response.data[i].owner_id, response.data[i].owner_name, response.data[i].owner_lastname],
                                        (results) => {
                                            console.log("Data Inserted..");
                                        }
                                    );
                                });
                            }
                        }
                    );
                });
            }
            console.log('datainfo length check_CallONLINE ==>', info.length);
            // delete database data
             console.log("Check 1");
              for (let a = 0; a < dataInfo.length; a++) {
                  var check_delete = true;
                  console.log("Check 2=>"+dataInfo[a].owner_id);
                 //  console.log('dataInfo length check ==>', dataInfo.length);
                  for (let j = 0; j < info.length; j++) {
                      if (dataInfo[a].owner_id == info[j].owner_id) {
                          check_delete = false;
                          console.log("Check 3=>"+info[j].owner_id);
                          break;
                      }
                  }
                  console.log("check 4");
                  if (check_delete == true) {
                      db.transaction(function (tx) {
                          console.log("DELETE");
                          console.log(dataInfo[i].owner_id'+ dataInfo[a].owner_id);
                            tx.executeSql(
                                'DELETE FROM ownerData where owner_id=?',
                                [dataInfo[a].owner_id],
                            )
                          console.log("Check 5");
                      })
                  }
              }
          })
          .catch(error => console.log('error', error));
    }
    const checkNetwork = () => {
```

```
    NetInfo.fetch().then((state) => {
        console.log('Connection type', state.type);
        if (state.isConnected == true) {
            console.log('Is connected?', state.isConnected);
            Alert.alert(
                'Success',
                'You are now connect this application',
                [
                    {
                        text: 'Ok',
                        onPress: () => callAPI(),
                    },
                ],
                { cancelable: false }
            );
        }
        else {
            Alert.alert(
                'OOPS ❌',
                'You are offline',
                [
                    {
                        text: 'Ok',
                        onPress: () => CallOFFLINE(),
                    },
                ],
                { cancelable: false }
            );
        }
    });
}
const CallOFFLINE = () => {
    db.transaction((tx) => {
        tx.executeSql(
            'SELECT * FROM ownerData',[],
            (tx, results) => {
                var temp = [];
                for (let i = 0; i < results.rows.length; ++i)
                    temp.push(results.rows.item(i));
                setinfo(temp);
                setdataInfo(temp);
                console.log('datainfo length check_CallOFFLINE ==>', dataInfo.length);
            }
        );
    });
}
```

```
   const renderItem = ({ item }) =>
   (
      <View>
         <View style={{ alignItems: "center", justifyContent: "space-between", marginBottom: 20, padding:
10, borderRadius: 10, backgroundColor: "lightgray", }}>
            <Text style={{ fontWeight: 'bold' }}>Owner_id   :{item.owner_id}</Text>
            <Text style={{ fontWeight: 'bold' }}>Owner_Name :  {item.owner_name}</Text>
            <Text style={{ fontWeight: 'bold' }}>Owner_Last_Name :  {item.owner_lastname}</Text>
         </View>
      </View>
   );
   return (
      <View>
         <Text style={{ textAlign: 'center', fontWeight: 'bold', fontSize: 25, }}>Data Listing</Text>
         {
            isLoading ?
               <View>
                  <ActivityIndicator size="large" />
                  <Text style={{ fontSize: 25, textAlign: "center" }}>Loading..</Text>
               </View>
               : null
         }
         <FlatList
            data={info}
            renderItem={renderItem}
            style={{ margin: 15 }}
         />
      </View>
   )
}
export default DataBaseListing;
```

# Splashscreen

- A splash screen is the first screen the users see after tapping the app icon.
- It's typically a simple screen with your app logo in the center and goes away once the app is ready to launch.
- Splash screens are typically used by particularly large applications to notify the user that the program is in the process of loading.
- The Splash Screen will automatically hide after a few seconds (3-5) from the screen and display when the application starts next time.

## Example

```
import React, { useEffect, useState } from "react";
import { StyleSheet, View, Text, Image,  } from 'react-native';

const App = () => {
  const [visible, setVisible] = useState(true);

  const Hide_Splash_Screen = () => {
    setVisible(false);
  }
  useEffect(() => {
    setTimeout(function () {
      Hide_Splash_Screen();
    }, 5000);
  }, []);
  {
    let Splash_Screen = (
        <View style={styles.RootView}>
            <View style={styles.ChildView}>
                <Image
source={{uri:'https://static.javatpoint.com/tutorial/react-native/images/react-native-tutor
ial.png'}}
                style={{width:'100%', height: '100%', resizeMode: 'contain'}} />
            </View>
        </View> )

  return (
       <View style = { styles.MainContainer }>
            <Text style={{textAlign: 'center'}}> This is Demo </Text>
              {
              (visible == true) ? Splash_Screen : null
            }
          </View>
            );
}
}
const styles = StyleSheet.create(
  {
     MainContainer:
     {
        flex: 1,
        justifyContent: 'center',
        alignItems: 'center',
     },
     RootView:
     {
        justifyContent: 'center',
```

```
        flex:1,
        margin: 10,
        position: 'absolute',
        width: '100%',
        height: '100%',
      },
    ChildView:
    {
        justifyContent: 'center',
        alignItems: 'center',
        backgroundColor: '#00BCD4',
        flex:1,
    },
  });
export default App;
```

# Moment JS

- MomentJS is a JavaScript library which helps in parsing, validating, manipulating and displaying date/time in JavaScript in a very easy way.
- This chapter will provide an overview of MomentJS and discusses its features in detail.
- Moment JS allows displaying of date as per localization and in human readable format.

## Library Install :

```
npm  install moment - - save
```

- In react-native we import the class of Moment js.

```
Import moment from “moment”;
```

## Date Picker:

- React Native Date Picker is a datetime picker for Android and iOS.
- It includes date, time and datetime picker modes.
- The datepicker is customizable and supports different languages.
- It's written with native code to achieve the best possible look, feel and performance.
- In react-native We download the library.
- Following library install for Date picker.

**Library Install :**

```
npm i @react-native-community/datetimepicker
```

- In react-native we import the class of date picker.

```
import DateTimePicker from '@react-native-community/datetimepicker';
```

## Example

```
import React, { useState } from 'react';
import { Button, SafeAreaView, ScrollView, StyleSheet, Text, TouchableOpacity, View } from
'react-native';
import DateTimePicker from '@react-native-community/datetimepicker';
import moment from "moment";

const MomentsJsDemo = ({ navigation }) => {
   const [datePicker, setDatePicker] = useState(false);
   const [date, setDate] = useState(new Date());

   const showDatePicker = () => {
      setDatePicker(true);
   };

   const onDateSelected = (event, value) => {
      setDate(value);
      setDatePicker(false);
   };
   const All = moment().format('MMMM Do YYYY, h:mm:ss a');
```

```
   const DateWithTime = moment(new Date()).format("DD-MM-YYYY");
   const Day = moment(new Date()).format("dddd");
   const DateWithMonthName = moment(new Date()).format("MMMM Do, YYYY");
   const a = moment().format('YYYY YYYY');
   const b = moment().format('MMMM');
   const c = moment().format('Do');
   const d = moment().format();
   const e = moment("20020826", "YYYYMMDD").fromNow();
   const f = moment().startOf('day').fromNow();
   const g = moment().endOf('day').fromNow();
   const h = moment().startOf('hour').fromNow();
   return (
      <SafeAreaView style={{ flex: 1 }}>
         <View style={styleSheet.MainContainer}>
            <ScrollView>
               <View>

                  {datePicker && (
                     <DateTimePicker
                        value={date}
                        onChange={onDateSelected}
                        style={styleSheet.datePicker}
                     />
                  )}

                  <View style={{ margin: 10 }}>
                     <Button title="Show Date Picker" color="green" onPress={showDatePicker} />
                  </View>
                  <Text style={styleSheet.text}>Selected Date = {date.toDateString()}</Text>

                  <Text style={{ textAlign: 'center', fontSize: 25, color: 'black', fontWeight: 'bold' }}>
----------</Text>
                  <Text style={{ textAlign: 'center', fontSize: 25, color: 'black', fontWeight: 'bold' }}>
Format Dates</Text>
                  <Text style={{ textAlign: 'center', fontSize: 25, color: 'black', fontWeight: 'bold' }}>
----------</Text>

                  <Text style={styleSheet.text}>All Item = {All}</Text>
                  <Text style={styleSheet.text}>Date with Time = {DateWithTime}</Text>
                  <Text style={styleSheet.text}>Day = {Day}</Text>
                  <Text style={styleSheet.text}>Date and Month Name = {DateWithMonthName}</Text>


                  <Text style={{ textAlign: 'center', fontSize: 25, color: 'black', fontWeight: 'bold' }}>
----------</Text>
                  <Text style={{ textAlign: 'center', fontSize: 25, color: 'black', fontWeight: 'bold' }}>
```

```
Relative Time</Text>
                    <Text style={{ textAlign: 'center', fontSize: 25, color: 'black', fontWeight: 'bold' }}>
----------------------</Text>
                    <Text style={styleSheet.text}>Year = {a}</Text>
                    <Text style={styleSheet.text}>Month Name = {b}</Text>
                    <Text style={styleSheet.text}>Date = {c}</Text>
                    <Text style={styleSheet.text}>Time  = {d}</Text>
                    <Text style={styleSheet.text}>Before  = {e}</Text>
                    <Text style={styleSheet.text}>Start of the day Hour = {f}</Text>
                    <Text style={styleSheet.text}>End of the day Hour = {g}</Text>
                    <Text style={styleSheet.text}>Currenent minutes = {h}</Text>
                    <TouchableOpacity onPress={() => navigation.navigate('MomentJS2')}>
                       <Text style={{
                          fontSize: 20,
                          color: 'blue',
                          padding: 3,
                          left: 150,
                          marginBottom: 10,
                          textAlign: 'center'
                       }}>Next </Text>
                    </TouchableOpacity>
                 </View>
              </ScrollView>
           </View>
        </SafeAreaView>
    );
}
const styleSheet = StyleSheet.create({
    MainContainer: {
        flex: 1,
        padding: 6,
        alignItems: 'center',
        backgroundColor: 'white'
    },
    text: {
        fontSize: 20,
        color: 'gray',
        padding: 3,
        marginBottom: 10,
        textAlign: 'center'
    },
})
export default MomentsJsDemo;
```

## Example 2 :

# Calculate day From place to Destination Example

```
import React, { useState } from "react";
import { Button, StyleSheet, Text, TextInput, View, Alert, Pressable } from 'react-native';
import DateTimePicker from '@react-native-community/datetimepicker';
import moment from "moment";

const MomentJS2 = ({ navigation }) => {

    const [txt, settxt] = useState("");
    const [days, setdays] = useState("");
    const [datePicker, setDatePicker] = useState(false);
    const [date, setDate] = useState(new Date());
    const [to, setTo] = useState(false);
    const [todate, setToDate] = useState(new Date());
    const [NewdatePicker, setnewDatePicker] = useState(false);
    const [Passdate, setpassDate] = useState(new Date());

    const showDatePicker = () => {
        setDatePicker(true);
    };
    const callDate = () => {
        setTo(true);
    };

    const onDateSelected = (event, value) => {
        setDate(value);
        setDatePicker(false);
    };

    const onToDateSelected = (event, value) => {
        setToDate(value);
        setTo(false);
    };

    const passDate = () => {
        setnewDatePicker(true);
    };

    const onDatePass = (event, value) => {
        setpassDate(value);
        setnewDatePicker(false);
        console.log("date " + value);
        console.log("date " + Passdate);
    };
```

```
  const conver = () => {

    var a = moment(todate);
    var b = moment(date);


    var years = a.diff(b, 'year');
    b.add(years, 'years');

    var months = a.diff(b, 'months');
    b.add(months, 'months');

    var days = a.diff(b, 'days');

    var message = todate.toDateString();
    message += " is "
    message += days + " days "
    message += months + " months "
    message += years + " years. \n"
    setdays(message)
    console.log(message);
  }

  const checkDateFormate = () => {
    // console.log(Passdate);
    // console.log(Passdate.toDateString());
    // var value = moment(Passdate).format("ddd MMM D YYYY");
    // console.log(value);

    // var date = moment('15-06-2010', 'DD-MM-YYYY')
    // console.log(date.format('MM-DD-YYYY'))
    var date = moment(Passdate).format('ddd MMM D YYYY');
    // console.log(date);
    // console.log(moment(date).format('MM/DD/YYYY'));
    // console.log(new Date());
    Alert.alert("Your Value ", moment(Passdate).format(txt));
  }
  return (
    <View style={styleSheet.MainContainer}>

      <View style={{ backgroundColor: 'white', height: "50%", width: '100%', top: 45 }}>
        {datePicker && (
          <DateTimePicker value={date} onChange={onDateSelected}
style={styleSheet.datePicker} />
        )}
        <View style={{ margin: 10 }}>
```

```
                <Button title="From" color="green" onPress={showDatePicker} />
            </View>
            <Text style={styleSheet.text}>Selected Date = {date.toDateString()}</Text>
            {to && (
                <DateTimePicker value={todate} onChange={onToDateSelected}
style={styleSheet.datePicker} />
            )}
            <View style={{ margin: 10 }}>
                <Button title="To" color="green" onPress={callDate} />
            </View>
            <Text style={styleSheet.text}>To date = {todate.toDateString()}</Text>
            <Button onPress={() => { conver() }} title="convert days" />
            <Text style={{ fontSize: 20, color: 'gray', padding: 10, marginBottom: 10, textAlign:
'center', top: 20 }}> {days}</Text>
        </View>
        <View style={{ backgroundColor: 'lightgray', height: "50%", width: '100%' }}>

            {NewdatePicker && (
                <DateTimePicker
                    value={Passdate}
                    onChange={onDatePass}
                    style={styleSheet.datePicker}
                />
            )}

            <View style={{ margin: 20, top: 25 }}>
                <Button title="Check Date" color="black" onPress={passDate} />
            </View>
            <TextInput style={{ backgroundColor: 'white', fontSize: 15, width: 350, left: 20,
borderRadius: 10, top: 40 }} placeholder="Type Date Formate" value={txt} onChangeText={txt
=> settxt(txt)} />
            <Pressable style={{ width: "90%", borderRadius: 10, top: 50, height: 50, alignItems:
"center", justifyContent: "center", marginTop: 20, marginLeft: 20, backgroundColor:
"#e33b4c", }} onPress={() => { checkDateFormate() }}>
                <Text style={{ color: "white", fontSize: 20, }} >Check</Text>
            </Pressable>
        </View>
    </View>
  )
}
const styleSheet = StyleSheet.create({
  MainContainer: {
    flex: 1,
    alignItems: 'center',
    backgroundColor: 'white'
  },
```

```
    text: {
        fontSize: 20,
        color: 'gray',
        padding: 10,
        marginBottom: 10,
        textAlign: 'center',
    },
})
export default MomentJS2;
```

# Time Picker:

- Time Picker allows you to select the time of day in either 24 hour or AM/PM mode.
- The time consists of hours, minutes and clock format.
- In react-native we use the same library for time picker which we use Date picker.
- And same process to import library.

## Example :

```
import React, { useState } from 'react';
import { View, Button, StyleSheet, Text, TouchableOpacity } from 'react-native';
import DateTimePicker from '@react-native-community/datetimepicker';
import moment from "moment";

const TimepickerDemo = () => {
    const [time, setTime] = useState("");
    const [timepicker, setTimePicker] = useState(false);
    const [displaymode, setMode] = useState('time');
    const [time1, setTime1] = useState("");
    const [timepicker1, setTimePicker1] = useState(false);
    const [displaymode1, setMode1] = useState('time');
    const [sendMSG, setSendMsg] = useState("");


 const showTimePicker = () => {
        setTimePicker(true);
    };
```

```
  const onTimeSelected = (event, value) => {
    setTimePicker(false);
    setTime(value.toLocaleTimeString());
    console.log("Your start time :" + value.toLocaleTimeString());
  };
  const showTimePicker1 = () => {
    setTimePicker1(true);
  };
  const onTimeSelected1 = (event, endvalue) => {
    setTimePicker1(false);
    setTime1(endvalue.toLocaleTimeString());
    console.log("Your end time :" + endvalue.toLocaleTimeString());
  };
  const calculate = () => {
    let end = moment(time1, "hh:mm");
    let start = moment(time, "hh:mm");

    let minutesDiff = end.diff(start, "minutes");
    console.log("Minutes:" + minutesDiff);
    var num = minutesDiff;
    var hours = (num / 60);
    var rhours = Math.floor(hours);
    var minutes = (hours - rhours) * 60;
    var rminutes = Math.round(minutes);
      console.log(num + " minutes = " + rhours + " hour(s) and " + rminutes + "
minute(s).");
    var message = "Your Time";
    message += " is "
    message += rhours + " hours "
    message += rminutes + " minutes "
    setSendMsg(message);
  }
  return (
    <View style={styles.container}>
      <View style={{ margin: 30 }}>
        {timepicker && (
          <DateTimePicker
            value={new Date()}
            mode={displaymode}
            is24Hour={true}
            display="default"
            onChange={onTimeSelected}
            style={styles.datePicker}
          />
        )}
        <Button title="Start Time" color="green" onPress={showTimePicker} />
```

```
            <Text style={styles.text}>Start Time = {time}</Text>
          </View>
          <View style={{ margin: 30 }}>
            {timepicker1 && (
                <DateTimePicker value={new Date()}  mode={displaymode1}
is24Hour={true} display="default" onChange={onTimeSelected1}
                    style={styles.datePicker}
                />
            )}
            <Button title="End Time" color="green" onPress={showTimePicker1} />
            <Text style={styles.text}>End Time = {time1}</Text>
          </View>
          <TouchableOpacity style={styles.button} onPress={() => calculate()}>
            <Text style={{ color: 'white', fontSize: 20 }}> Calculate Time </Text>
          </TouchableOpacity>
          <Text style={{ fontSize: 20, color: 'red',padding: 3,marginBottom: 10,textAlign:
'center', fontWeight: 'bold', top: 20 }}>{sendMSG}</Text>
        </View>
    );
};
const styles = StyleSheet.create({
    container: {
        flex: 1,
       justifyContent:'center',
        backgroundColor: "white",
    },
    text: {
        fontSize: 20,
        color: 'gray',
        padding: 3,
        marginBottom: 10,
        textAlign: 'center'
    },
    button: {
        alignItems: 'center',
        backgroundColor: 'blue',
        padding: 10,
        marginTop: 16,
        marginLeft: 35,
        marginRight: 35,
    },
});
export default TimepickerDemo;
```

# Tab View

- In TabView components we download the library which command is given below.

```
npm i react-native-tab-view
```

- After the download library we import the class for the use of tab view.

```
import { TabView, SceneMap } from 'react-native-tab-view';
```

- After importing the class following code.

```
import React, { useState } from 'react';
import { Image, StyleSheet, } from 'react-native';
import { TabView, SceneMap, TabBar, } from 'react-native-tab-view';
import Female from './Female';
import Male from './Male';
import AllItems from './AllItems';

const TabViewScreen = () => {
    const [index, setIndex] = useState(0);
    const [routes] = useState([
        { key: 'first', title: 'Male' },
        { key: 'second', title: 'Female' },
        { key: 'third', title: 'AllItems' },
    ]);
    const renderTabBar = props => (
        <TabBar
            renderIcon={({ route, focused, color }) => (
                route.title == "Male" ?
                    <Image source={require('./assets/male-user.png')} style={{ height: 20,
width: 20, color: 'white', }} tintColor={focused ? "white" : "black"} />
                    :
                    route.title == "Female" ?
                        <Image source={require('./assets/ask.png')} style={{ height: 20, width:
20, color: 'white', }} tintColor={focused ? "white" : "black"} />
                        :
                        <Image source={require('./assets/back.png')} style={{ height: 20, width:
20, color: 'white', }} tintColor={focused ? "white" : "black"} />
            )}
            {...props}
```

```
            indicatorStyle={{ backgroundColor: 'white', }}
            style={{ backgroundColor: '#909090' }} />
    );
    return (
        <TabView
            navigationState={{ index, routes }}
            renderScene={SceneMap({
                first: Male,
                second: Female,
                third: AllItems,
            })}
            onIndexChange={setIndex}
            renderTabBar={renderTabBar}
        />
    );
}
export default TabViewScreen;
```

- In this task i am import 3 other files which name is
    - Male.js
    - Female.js
    - Allitems.js
- After importing the file tab view, render the file and display the output from selected options.
- And which option you selected then this option focused and white color part.

# React-native Hooks :

- **What is Hook ?**

- In react-native we must import Hooks.
- There are 3 rules for hooks:
  - (i) Hooks can only be called inside React function components.
  - (ii) Hooks can only be called at the top level of a component.
  - (iii) Hooks cannot be conditional
- Note: Hooks will not work in React class components.
- There are many types of hooks available in React native.
  - ➢ useState
  - ➢ useEffect
  - ➢ useContext
  - ➢ useRef
  - ➢ useReducer
  - ➢ useCallback
  - ➢ useMemo
  - ➢ Custom hooks

# 1) useState

- The React useState Hook allows us to track state in a function component.
- At the top of your component, import the useState Hook.

```
import { useState } from "react"
```

- After initializing our state by calling useState in our task.

```
const[test , setTest ] = useState()
```

**Example :**

```
import { useState } from "react";
   const FavoriteColor = () => {
   const [color, setColor] = useState("red");
         return (
            <View>
              <Text>My favorite color is {color}!</Text>
              <button
                type="button"
                onClick={() => setColor("blue")}
              >Blue</button>
            </View>
          )
        }
   export default FavoriteColor;
```

# 2) useEffect

- The useEffect Hook allows you to perform side effects in your components.
- useEffect accepts two arguments. The second argument is optional.

useEffect(<function>, <dependency>)

- it's one type of when Screen opens then first it will execute this code.

**Example :**

```
import { useState, useEffect } from "react";
import ReactDOM from "react-dom/client";
const Timer = () => {
  const [count, setCount] = useState(0);
  useEffect(() => {
    setTimeout(() => {
      setCount((count) => count + 1);
    }, 1000);
  });
  return(
          <Text>I've rendered {count} times!</Text>
       );
}
```

# React-native Searchbar :

- Searchbar is a simple input box where users can type search queries.
- SearchBars are used to search or filter items.
- Use a SearchBar when the number of items directly impacts a user's ability to find one of them.

## Example :

```
import React, { useState, useEffect } from 'react';

import { SafeAreaView, Text, StyleSheet, View, FlatList, TextInput, } from 'react-native';

const App = () => {
    const [search, setSearch] = useState('');
    const [filteredDataSource, setFilteredDataSource] = useState([]);
    const [masterDataSource, setMasterDataSource] = useState([]);
    useEffect(() => {
        fetch('https://jsonplaceholder.typicode.com/posts')
            .then((response) => response.json())
            .then((responseJson) => {
                setFilteredDataSource(responseJson);
                setMasterDataSource(responseJson);
            })
            .catch((error) => {
                console.error(error);
            });
    }, []);

    const searchFilterFunction = (text) => {
        if (text) {
            const newData = masterDataSource.filter(function (item) {
                const itemData = item.title ? item.title.toUpperCase() : ''.toUpperCase();
                const textData = text.toUpperCase();
                return itemData.indexOf(textData) > -1;
            });
            setFilteredDataSource(newData);
            setSearch(text);
        } else {
            setFilteredDataSource(masterDataSource);
            setSearch(text);
        }
    };
```

```
    const ItemView = ({ item }) => {
      return (
        <Text style={styles.itemStyle} onPress={() => getItem(item)}>
          {item.id} {'.'} {item.title.toUpperCase()}
        </Text>
      );
    };

    const ItemSeparatorView = () => {
      return (
        <View style={{ height: 1, width: '100%', backgroundColor: '#C8C8C8', }} />
      );
    };

    return (
      <SafeAreaView style={{ flex: 1 }}>
        <View style={styles.container}>
          <TextInput style={styles.textInputStyle} onChangeText={(text) =>
searchFilterFunction(text)} value={search}  placeholder="Search Here" />
          {filteredDataSource.length == 0 ?
            <Text style={{ fontSize: 20, fontWeight: 'bold', textAlign: 'center' }}> No
data</Text>
            :
            <FlatList
              data={filteredDataSource}
              keyExtractor={(item, index) => index.toString()}
              ItemSeparatorComponent={ItemSeparatorView}
              renderItem={ItemView}
            />}
        </View>
      </SafeAreaView>
    );
};

const styles = StyleSheet.create({
    container: {
      backgroundColor: 'white',
    },
    itemStyle: {
      padding: 10,
    },
    textInputStyle: {
      height: 40,
      borderWidth: 1,
      paddingLeft: 20,
      margin: 5,
```

```
        borderColor: '#009688',
        backgroundColor: '#FFFFFF',
    },
});

export default App;
```

# React-native Upgrade :

- In react -native Project info below command used.

```
react-native info
```

- After the command executes you get all information about the project.
- So you check details and show the version of the project.
- And if your version is old then below command used then you upgrade the latest version of react-native project.

```
npx react-native upgrade
```

- You may specify a React Native version by passing an argument, e.g. to upgrade to 0.61.0-rc.0 run:

```
npx react-native upgrade 0.61.0-rc.0.70.1
```

- If you check the project which file updates the old version to the latest version then below link is helpful to show the updated file and give location of the code which code changes.

https://react-native-community.github.io/upgrade-helper/

# Dark/Light Mode :

- The Appearance module exposes information about the user's appearance preferences, such as their preferred color scheme (light or dark).
- We import the class from react-naitve library.

```
import { Appearance } from 'react-native';
```

- You can use the Appearance module to determine if the user prefers a dark color scheme:

```
const colorScheme = Appearance.getColorScheme();
if (colorScheme === 'dark') {
  // Use dark color scheme
}
```

**Example :**

```
import React, { useEffect } from "react";
import { Alert, Button, StyleSheet, Text, View } from "react-native";
import { Appearance } from 'react-native';

const DarkMode = () => {
    useEffect(() => {
        //  we find which mode used in your phobne  ==>
Appearance.getColorScheme()
        console.log("Mode is => " + Appearance.getColorScheme())
        Alert.alert("Hello", "This is " + Appearance.getColorScheme() + " mode")
    }, []);
    return (
        <View style={Appearance.getColorScheme() == 'dark' ? styles.dark1 :
styles.light1}>
            <View>
                <Text style={Appearance.getColorScheme() == 'dark' ? styles.dark1 :
styles.light1}>{Appearance.getColorScheme()} mode..</Text>
            </View>
```

```
        </View>
    )
}
const styles = StyleSheet.create({
    dark1: {
        backgroundColor: 'black',
        color: "white",
        fontSize: 25,
        textAlign: "center",
    },
    light1: {
        backgroundColor: "white",
        color: 'black',
        fontSize: 25,
        textAlign: "center",
    }
})
export default DarkMode;
```

- When your Mobile theme changes then this code works.

# Web View :

- A WebView is an embedded browser that can be used to display web pages inside your React Native applications.
- It can display anything (from custom HTML elements to entire web applications) inside React Native.
- In React Native, we can use WebView by using a third-party package called react-native-webview.
- Open a Terminal in your project's folder and run:

```
npm i react-native-webview
```

- After sometime the process completes then your library is installed successfully and imports the library top of the code.

```
import React from 'react';
import { SafeAreaView, } from 'react-native';
import { WebView } from 'react-native-webview';

const App = () => {
    return (
        <SafeAreaView style={{ flex: 1 }}>
            <WebView source={{ uri: 'http://darshanm.000webhostapp.com/' }} />
        </SafeAreaView>
    )
};

export default App;
```

# React-native Slider :

- First, install the library using the below npm code.

```
npm i react-native-app-intro-slider
```

- After sometime the process completes then your library is installed successfully and imports the library top of the code.

```
import ImageSlider from 'react-native-image-slider';
```

### Example

```
import React, { Component } from 'react';
import { StyleSheet, Text, View, Image, TouchableHighlight, } from 'react-native';
import ImageSlider from 'react-native-image-slider';
const ImageSliderDemo = () => {
    const images = [
        'https://placeimg.com/640/640/nature',
        'https://placeimg.com/640/640/animals',
        'https://placeimg.com/640/640/beer',
        'https://placeimg.com/640/640/cocke',
```

```
    ];
        return (
            <View style={styles.container}>
                <View style={styles.content1}>
                    <Text style={styles.contentText}>Image slider</Text>
                </View>
                <ImageSlider
                    autoPlayWithInterval={3000}
                    images={images}
                    customButtons={(position, move) => (
                        <View style={styles.buttons}>
                            {images.map((image, index) => {
                                return (
                                    <TouchableHighlight
                                        key={index}
                                        underlayColor="#ccc"
                                        onPress={() => move(index)}
                                        style={styles.button}>
                                        <Text style={position === index && styles.buttonSelected}>
                                            {index + 1}
                                        </Text>
                                    </TouchableHighlight>
                                );
                            })}
                        </View>
                    )} />
            </View>
        );     }
const styles = StyleSheet.create({
    container: {  flex: 1,},
    content1: {
        width: '100%',
        height: '5%',marginBottom:10,
        backgroundColor: '#000',
        justifyContent: 'center',
        alignItems: 'center',
    },
    content2: {
        width: '100%',
        height: 50,
        bottom: 300,
        backgroundColor: '#000',
        justifyContent: 'center',
        alignItems: 'center',
    },
    contentText: { color: '#fff', fontSize: 25 },
```

```
    buttons: {
        height: 350,
        justifyContent: 'center',
        alignItems: 'center',
        flexDirection: 'row',
    },
    button: {
        bottom: 160,
        margin: 3,
        width: 15,
        height: 15,
        alignItems: 'center',
        justifyContent: 'center',
    },
    buttonSelected: {
        color: 'red',
    },
});
export default ImageSliderDemo;
```

# React-native Intro - Screen :

- React Native Intro Slider is used to introduce your App.
- It is used to showcase the attractive features of your App,
- Example, if you are making an E-commerce App then you can showcase the features like best deals and offers, Fast Delivery, vast variety.
- To make an Intro Slider we are going to use react-native-app-intro-slider library.

```
npm install react-native-app-intro-slider --save
```

- After sometime the process completes then your library is installed successfully and imports the library top of the code.

```
import AppIntroSlider from 'react-native-app-intro-slider';
```

## Example

```
import React, { useState } from 'react';
import { SafeAreaView, StyleSheet, View, Text, Image, Button, } from 'react-native';
import AppIntroSlider from 'react-native-app-intro-slider';

const App = () => {
  const [showRealApp, setShowRealApp] = useState(false);

  const onDone = () => {
    setShowRealApp(true);
  };
  const onSkip = () => {
    setShowRealApp(true);
  };
  const RenderItem = ({ item }) => {
    return (
      <View style={{ flex: 1, backgroundColor: item.backgroundColor, alignItems: 'center', justifyContent:
'space-around', paddingBottom: 100, }}>
        <Text style={styles.introTitleStyle}>
          {item.title}
        </Text>
        <Image source={{uri:item.image}}
          style={styles.introImageStyle}/>

        <Text style={styles.introTextStyle}>
          {item.text}
        </Text>
      </View>
    );
  };
  return (
    <>
      {showRealApp ? (
        <SafeAreaView style={styles.container}>
          <View style={styles.container}>
            <Text style={styles.titleStyle}>
              React Native App Intro Slider using AppIntroSlider
            </Text>
            <Text style={styles.paragraphStyle}>
              This will be your screen when you click Skip
              from any slide or Done button at last
            </Text>
            <Button
              title="Show Intro Slider again"
```

```
            onPress={() => setShowRealApp(false)}
          />
        </View>
      </SafeAreaView>
    ) : (
      <AppIntroSlider
        data={slides}
        renderItem={RenderItem}
        onDone={onDone}
        showSkipButton={true}
        onSkip={onSkip}
      />
    )}
  </>
 );
};

export default App;

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: '#fff',
  alignItems: 'center',
  padding: 10,
  justifyContent: 'center',
 },
 titleStyle: {
  padding: 10,
  textAlign: 'center',
  fontSize: 18,
  fontWeight: 'bold',
 },
 paragraphStyle: {
  padding: 20,
  textAlign: 'center',
  fontSize: 16,
 },
 introImageStyle: {
  width: 200,
  height: 200,
 },
 introTextStyle: {
  fontSize: 18,
  color: 'white',
  textAlign: 'center',
```

```
    paddingVertical: 30,
  },
  introTitleStyle: {
    fontSize: 25,
    color: 'white',
    textAlign: 'center',
    marginBottom: 16,
    fontWeight: 'bold',
  },
});

const slides = [
  {
    key: 's1',
    text:  'Best Recharge offers',
    title: 'Mobile Recharge',
    image:
'https://raw.githubusercontent.com/AboutReact/sampleresource/master/intro_mobile_recharge.png',
    backgroundColor: '#20d2bb',
  },
  {
    key: 's2',
    title: 'Flight Booking',
    text: 'Upto 25% off on Domestic Flights',
    image:
'https://raw.githubusercontent.com/AboutReact/sampleresource/master/intro_flight_ticket_booking.png',
    backgroundColor: '#febe29',
  },
  {
    key: 's3',
    title: 'Great Offers',
    text: 'Enjoy Great offers on our all services',
 image:'https://raw.githubusercontent.com/AboutReact/sampleresource/master/intro_discount.png',
    backgroundColor: '#22bcb5',
  },
  {
    key: 's4',
    title: 'Best Deals',
    text: 'Best Deals on all our services',
image:'https://raw.githubusercontent.com/AboutReact/sampleresource/master/intro_best_deals.png',
    backgroundColor: '#3395ff',
  },
  {
    key: 's5',
    title: 'Bus Booking',
    text: 'Enjoy Traveling on Bus with flat 100% off',
```

```
image:'https://raw.githubusercontent.com/AboutReact/sampleresource/master/intro_bus_ticket_booking.
png',
    backgroundColor: '#f6437b',
  },
  {
    key: 's6',
    title: 'Train Booking',
    text: ' 10% off on first Train booking',
image:'https://raw.githubusercontent.com/AboutReact/sampleresource/master/intro_train_ticket_booking.
png',
  backgroundColor: '#febe29',
  },
];
```

# Firebase :

- Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps.
- Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiments.
- Installing React Native Firebase requires a few steps; installing the NPM module, adding the Firebase config files & rebuilding your application.

```
npm install --save @react-native-firebase/app
```

- The @react-native-firebase/app module must be installed before using any other Firebase service.
- Download the google-services.json file and place it inside of your project at the following location: /android/app/google-services.json.
- After add the google-services plugin as a dependency inside of your /android/build.gradle file:

```
classpath 'com.google.gms:google-services:4.3.14'
```

- Lastly, execute the plugin by adding the following to your /android/app/build.gradle .

```
apply plugin: 'com.google.gms.google-services'
```

# 1) Cloud Firestore

- This module requires that the @react-native-firebase/app module is already set up and installed.

```
# Install & setup the app module
npm add @react-native-firebase/app
```

```
# Install the firestore module
npm add @react-native-firebase/firestore
```

- Cloud Firestore stores data within "documents", which are contained within "collections", and documents can also contain collections.
- For example, we could store a list of our users documents within a "Users" collection.
- The collection method allows us to reference a collection within our code:

```
import React, { useEffect } from "react";
import { Text, View } from "react-native";
import firestore from '@react-native-firebase/firestore'


const AddDataFirebase = () => {
   useEffect(() => {
      getdata();
   })
   const getdata = async () => {
      const users = await firestore().collection('Users').doc('ABC').get();
      console.log('User Details', users);
   }
   return (
      <View>
         <Text style={{ textAlign: "center", color: 'black', fontWeight: "bold", fontSize: 25
}}>Firebase </Text>
```

```
      </View>
    )
}
export default AddDataFirebase;
```

# 2) Cloud Messaging

- This module requires that the @react-native-firebase/app module is already set up and installed.

```
# Install & setup the app module
npm add @react-native-firebase/app
```

```
# Install the firestore module
npm add @react-native-firebase/messaging
```

## Foreground state messages

```
import React, { useEffect } from 'react';
import { Alert } from 'react-native';
import messaging from '@react-native-firebase/messaging';
const PushNotificationDemo = () => {
    useEffect(() => {
        getToken();
        const unsubscribe = messaging().onMessage(async remoteMessage => {
            Alert.alert('A new FCM message arrived!', JSON.stringify(remoteMessage));
          });
        console.log("End");
        return unsubscribe;
    }, []);
    async function getToken (){
        await messaging().registerDeviceForRemoteMessages();
        const token = await messaging().getToken();
        console.log("Start "+JSON.stringify(token,2,null));
    }
}
export default PushNotificationDemo;
```

# 3) Phone Authentication :

- Phone authentication allows users to sign in to Firebase using their phone as the authenticator.
- An SMS message is sent to the user via their phone number containing a unique code.
- Once the code has been authorized, the user is able to sign in to Firebase.

| |
|---|
| *# Install & setup the app module*<br>npm add @react-native-firebase/app |
| *# Install the firestore Authentication module*<br>npm add @react-native-firebase/auth |

**Example :**

```
import React, { useState } from 'react';
import { Button, SafeAreaView, Text, View, } from 'react-native';
import auth from '@react-native-firebase/auth';
import { TextInput } from 'react-native-paper';

const App = () => {
   var confirm = "";
   const [code, setCode] = useState('');
   const [phone, setphone] = useState("");

   async function SendOTP() {
      console.log("Number " + phone);
      try {
         const confirmation = await auth().signInWithPhoneNumber("+91 ENTER
NUMBER");
         console.log(confirmation)
      } catch (error) {
         console.log("error " + error);
      }
   }
    return (
      <SafeAreaView style={{ alignItems: 'center', flex: 1, marginTop: 100 }}>
         <View style={{ margin: 10 }}>
            <Text>Mobile Sign</Text>
```

```
        </View>
        <Button title="Phone Number To send OTP" onPress={() => SendOTP()} />
       </SafeAreaView>
   );
};
export default App;
```

# 4) Email/Password Authentication :

- We will use Email + Password for authentication.
- Users can register using Email and password and can login with the same Email and password.

```
# Install & setup the app module
npm add @react-native-firebase/app
```

```
# Install the firestore module
npm add @react-native-firebase/messaging
```

- You can see
    1. **SplashScreen.js** for the Splash Screen.
    2. **LoginScreen.js** for the Login
    3. **RegisterScreen.js** for the Register
    4. **HomeScreen.js**, as a landing page after login/register and have a logout option.

## SplashScreen(SplashScreen.js)

- Check if currentUser is set or not If not then send for Authentication else send to Home Screen.

```
import React, { useState, useEffect } from "react";
import { SafeAreaView, ActivityIndicator, Text, View, StyleSheet, } from "react-native";
import auth from "@react-native-firebase/auth";

const Splash = ({ navigation }) => {
    const [animating, setAnimating] = useState(true);
    useEffect(() => {
        setTimeout(() => {
            setAnimating(false);
            navigation.replace(
                auth().currentUser ? "FHomeScreen" : "LoginScreen"
            );
        }, 5000);
    }, []);
    return (
        <SafeAreaView style={{ flex: 1, backgroundColor: "#307ecc" }} >
            <View style={styles.container}>
                <Text style={{ fontSize: 18, textAlign: "center", color: "white", }}  >
                    React Native Firebase Authentication
                </Text>
                <ActivityIndicator animating={animating}
                    color="#FFFFFF"
                    size="large"
                    style={styles.activityIndicator} />
            </View>
        </SafeAreaView>
    );
};
export default Splash;
const styles = StyleSheet.create({
    container: {
        flex: 1,
        alignItems: "center",
        justifyContent: "center",
    },
    activityIndicator: {
        alignItems: "center",
        height: 80,
    },
});
```

## Login Screen(LoginScreen.js)

```
import React, { useState, createRef } from "react";
import { SafeAreaView, StyleSheet, TextInput, View, Text, ScrollView, Keyboard,
TouchableOpacity, KeyboardAvoidingView, } from "react-native";
import auth from "@react-native-firebase/auth";

const LoginScreen = ({ navigation }) => {
  const [userEmail, setUserEmail] = useState("");
  const [userPassword, setUserPassword] = useState("");
  const [errortext, setErrortext] = useState("");

  const passwordInputRef = createRef();

  const handleSubmitPress = () => {
    setErrortext("");
    if (!userEmail) {
      alert("Please fill Email");
      return;
    }
    if (!userPassword) {
      alert("Please fill Password");
      return;
    }
    auth()
      .signInWithEmailAndPassword(userEmail, userPassword)
      .then((user) => {
        console.log('user', user);
        // If server response message same as Data Matched
        if (user) navigation.replace("FHomeScreen");
      })
      .catch((error) => {
        console.log(error);
        if (error.code === "auth/invalid-email")
          setErrortext(error.message);
        else if (error.code === "auth/user-not-found")
          setErrortext("No User Found");
        else {
          setErrortext("Please check your email id or password");
        }
      });
  };
  return (
    <SafeAreaView style={styles.mainBody}>
      <ScrollView
        keyboardShouldPersistTaps="handled"
```

```
        contentContainerStyle={{
          flex: 1,
          justifyContent: "center",
          alignContent: "center",
        }}>
        <View>
          <KeyboardAvoidingView enabled>
            <View style={{ alignItems: "center" }}>
            </View>
            <View style={styles.sectionStyle}>
              <TextInput
                style={styles.inputStyle}
                onChangeText={(UserEmail) => setUserEmail(UserEmail) }
                placeholder="Enter Email"
                placeholderTextColor="#8b9cb5"
                autoCapitalize="none"
                keyboardType="email-address"
                returnKeyType="next"
                onSubmitEditing={() =>
                  passwordInputRef.current &&
                  passwordInputRef.current.focus()
                }
                underlineColorAndroid="#f000" />
            </View>
            <View style={styles.sectionStyle}>
              <TextInput
                style={styles.inputStyle}
                onChangeText={(UserPassword) => setUserPassword(UserPassword)}
                placeholder="Enter Password"
                placeholderTextColor="#8b9cb5"
                keyboardType="default"
                ref={passwordInputRef}
                onSubmitEditing={Keyboard.dismiss}
                secureTextEntry={true}
                underlineColorAndroid="#f000"
                returnKeyType="next" />
            </View>
            {errortext != "" ? (
              <Text style={styles.errorTextStyle}>
                {" "}
                {errortext}{" "}
              </Text>
            ) : null}
            <TouchableOpacity
              style={styles.buttonStyle}
              activeOpacity={0.5}
```

```
                    onPress={handleSubmitPress}>
                    <Text style={styles.buttonTextStyle}>
                      LOGIN
                    </Text>
                  </TouchableOpacity>
                  <Text
                    style={styles.registerTextStyle}
                    onPress={() =>navigation.navigate("FRegisterScreen") }>
                    New Here ? Register
                  </Text>
                </KeyboardAvoidingView>
              </View>
            </ScrollView>
            <Text style={{
                  fontSize: 18,
                  textAlign: "center",
                  color: "white",
              }}>
              React Native Firebase Authentication
            </Text>
        </SafeAreaView>
    );
};
export default LoginScreen;

const styles = StyleSheet.create({
    mainBody: {
        flex: 1,
        justifyContent: "center",
        backgroundColor: "#307ecc",
        alignContent: "center",
    },
    sectionStyle: {
        flexDirection: "row",
        height: 40,
        marginTop: 20,
        marginLeft: 35,
        marginRight: 35,
        margin: 10,
    },
    buttonStyle: {
        backgroundColor: "#7DE24E",
        borderWidth: 0,
        color: "#FFFFFF",
        borderColor: "#7DE24E",
        height: 40,
```

```
        alignItems: "center",
        borderRadius: 30,
        marginLeft: 35,
        marginRight: 35,
        marginTop: 20,
        marginBottom: 25,
    },
    buttonTextStyle: {
        color: "#FFFFFF",
        paddingVertical: 10,
        fontSize: 16,
    },
    inputStyle: {
        flex: 1,
        color: "white",
        paddingLeft: 15,
        paddingRight: 15,
        borderWidth: 1,
        borderRadius: 30,
        borderColor: "#dadae8",
    },
    registerTextStyle: {
        color: "#FFFFFF",
        textAlign: "center",
        fontWeight: "bold",
        fontSize: 14,
        alignSelf: "center",
        padding: 10,
    },
    errorTextStyle: {
        color: "red",
        textAlign: "center",
        fontSize: 14,
    },
});
```

## RegisterScreen (RegisterScreen.js)

```
import React, { useState, createRef } from "react";
import { SafeAreaView, StyleSheet, TextInput, View, Text, KeyboardAvoidingView, Keyboard,
TouchableOpacity, ScrollView, } from "react-native";
import auth from "@react-native-firebase/auth";

const FRegisterScreen = ({ navigation }) => {
    const [userName, setUserName] = useState("");
```

```
  const [userEmail, setUserEmail] = useState("");
  const [userPassword, setUserPassword] = useState("");
  const [errortext, setErrortext] = useState("");

  const emailInputRef = createRef();
  const passwordInputRef = createRef();

  const handleSubmitButton = () => {
    setErrortext("");
    if (!userName) return alert("Please fill Name");
    if (!userEmail) return alert("Please fill Email");
    if (!userPassword) return alert("Please fill Address");

    auth()
      .createUserWithEmailAndPassword(userEmail, userPassword)
      .then((user) => {
        console.log("Registration Successful. Please Login to proceed");
        console.log(user);
        if (user) {
          auth().currentUser.updateProfile({ displayName: userName, })
            .then(() => navigation.replace("FHomeScreen"))
            .catch((error) => {
              alert(error);
              console.error(error);
            });
        }
      })
      .catch((error) => {
        console.log(error);
        if (error.code == "auth/email-already-in-use") {
          setErrortext("That email address is already in use!");
        } else {
          setErrortext(error.message);
        }
      });
  };
  return (
    <SafeAreaView style={{ flex: 1, backgroundColor: "#307ecc" }}>
      <ScrollView
        keyboardShouldPersistTaps="handled"
        contentContainerStyle={{ justifyContent: "center", alignContent: "center", }}>
        <KeyboardAvoidingView enabled>
          <View style={styles.sectionStyle}>
            <TextInput
              style={styles.inputStyle}
              onChangeText={(UserName) => setUserName(UserName)}
```

```
            underlineColorAndroid="#f000"
            placeholder="Enter Name"
            placeholderTextColor="#8b9cb5"
            autoCapitalize="sentences"
            returnKeyType="next"
            onSubmitEditing={() =>
              emailInputRef.current &&
              emailInputRef.current.focus()
            }
            blurOnSubmit={false} />
        </View>
        <View style={styles.sectionStyle}>
          <TextInput
            style={styles.inputStyle}
            onChangeText={(UserEmail) => setUserEmail(UserEmail)}
            underlineColorAndroid="#f000"
            placeholder="Enter Email"
            placeholderTextColor="#8b9cb5"
            keyboardType="email-address"
            ref={emailInputRef}
            returnKeyType="next"
            onSubmitEditing={() =>
              passwordInputRef.current &&
              passwordInputRef.current.focus()
            }
            blurOnSubmit={false} />
        </View>
        <View style={styles.sectionStyle}>
          <TextInput
            style={styles.inputStyle}
            onChangeText={(UserPassword) => setUserPassword(UserPassword)}
            underlineColorAndroid="#f000"
            placeholder="Enter Password"
            placeholderTextColor="#8b9cb5"
            ref={passwordInputRef}
            returnKeyType="next"
            secureTextEntry={true}
            onSubmitEditing={Keyboard.dismiss}
            blurOnSubmit={false} />
        </View>
        {errortext != "" ? (
          <Text style={styles.errorTextStyle}>
            {" "}
            {errortext}{" "}
          </Text>
        ) : null}
```

```
                <TouchableOpacity
                    style={styles.buttonStyle}
                    activeOpacity={0.5}
                    onPress={handleSubmitButton}>
                    <Text style={styles.buttonTextStyle}>
                        REGISTER
                    </Text>
                </TouchableOpacity>
                <Text style={{
                    color: "#FFFFFF",
                    textAlign: "center",
                    fontWeight: "bold",
                    fontSize: 14,
                    alignSelf: "center",
                    padding: 10,
                }}
                    onPress={() => navigation.navigate("LoginScreen")}>
                    Login Screen
                </Text>
            </KeyboardAvoidingView>
        </ScrollView>
        <Text style={{
            fontSize: 18,
            textAlign: "center",
            color: "white",
        }}>
            React Native Firebase Authentication
        </Text>
    </SafeAreaView>
  );
};
export default FRegisterScreen;

const styles = StyleSheet.create({
  sectionStyle: {
    flexDirection: "row",
    height: 40,
    marginTop: 20,
    marginLeft: 35,
    marginRight: 35,
    margin: 10,
  },
  buttonStyle: {
    backgroundColor: "#7DE24E",
    borderWidth: 0,
    color: "#FFFFFF",
```

```
        borderColor: "#7DE24E",
        height: 40,
        alignItems: "center",
        borderRadius: 30,
        marginLeft: 35,
        marginRight: 35,
        marginTop: 20,
        marginBottom: 20,
    },
    buttonTextStyle: {
        color: "#FFFFFF",
        paddingVertical: 10,
        fontSize: 16,
    },
    inputStyle: {
        flex: 1,
        color: "white",
        paddingLeft: 15,
        paddingRight: 15,
        borderWidth: 1,
        borderRadius: 30,
        borderColor: "#dadae8",
    },
    errorTextStyle: {
        color: "red",
        textAlign: "center",
        fontSize: 14,
    },
});
```

## HomeScreen(HomeScreen.js)

```
import React, { useEffect, useState } from "react";
import { View, Text, SafeAreaView, StyleSheet, TouchableOpacity, Alert, } from
"react-native";
import auth from "@react-native-firebase/auth";

const FHomeScreen = ({ navigation }) => {
    const [user, setUser] = useState();

    useEffect(() => {
        const subscriber = auth().onAuthStateChanged((user) => {
            console.log("user", JSON.stringify(user));
            setUser(user);
        });
```

```
    return subscriber;
}, []);

const logout = () => {
    Alert.alert(
        "Logout",
        "Are you sure? You want to logout?",
        [
            {
                text: "Cancel",
                onPress: () => {
                    return null;
                },
            },
            {
                text: "Confirm",
                onPress: () => {
                    auth()
                        .signOut()
                        .then(() => navigation.replace("LoginScreen"))
                        .catch((error) => {
                            console.log(error);
                            if (error.code === "auth/no-current-user")
                                navigation.replace("LoginScreen");
                            else alert(error);
                        });
                },
            },
        ],
        { cancelable: false }
    );
};
return (
    <SafeAreaView style={{ flex: 1 }}>
        <View style={{ flex: 1, padding: 16 }}>
            <View style={{ flex: 1, alignItems: "center", justifyContent: "center", }}>
                <Text style={{ fontSize: 20, textAlign: "center", marginBottom: 16, }}>
                    Firebase Authentication
                </Text>
                {user ? (
                    <Text>
                        Welcome{" "}
                        {user.displayName
                            ? user.displayName
                            : user.email}
                    </Text>
```

```
                ) : null}
                {user ? (
                  <Text>
                    Email {" "}
                    {user.displayName
                      ? user.email
                      : user.email}
                  </Text>
                ) : null}
                <TouchableOpacity
                  style={styles.buttonStyle}
                  activeOpacity={0.5}
                  onPress={logout}>
                  <Text style={styles.buttonTextStyle}>
                    Logout
                  </Text>
                </TouchableOpacity>
              </View>
              <Text style={{ fontSize: 18, textAlign: "center", color: "grey", }}>
                React Native Firebase Authentication
              </Text>
            </View>
        </SafeAreaView>
    );
};
export default FHomeScreen;
const styles = StyleSheet.create({
    buttonStyle: {
        minWidth: 300,
        backgroundColor: "#7DE24E",
        borderWidth: 0,
        color: "#FFFFFF",
        borderColor: "#7DE24E",
        height: 40,
        alignItems: "center",
        borderRadius: 30,
        marginLeft: 35,
        marginRight: 35,
        marginTop: 20,
        marginBottom: 25,
    },
    buttonTextStyle: {
        color: "#FFFFFF",
        paddingVertical: 10,
        fontSize: 16,
    },  });
```

# 5) Facebook Authentication :

- Facebook is a very popular platform so we will see the example to integrate the Facebook login in our App.
- In firebase console facebook authentication enables and generates App Id and App Secret Id.

```
import React, { useEffect } from 'react';
import { Button, View } from 'react-native';
import auth from '@react-native-firebase/auth';
import { LoginManager, AccessToken } from 'react-native-fbsdk-next';
import { LoginButton } from 'react-native-fbsdk-next';

async function onFacebookButtonPress() {
    // Attempt login with permissions
    const result = await LoginManager.logInWithPermissions(['public_profile', 'email']);

    if (result.isCancelled) {
      throw 'User cancelled the login process';
    }
    // Once signed in, get the user's Access Token
    const data = await AccessToken.getCurrentAccessToken();
    if (!data) {
      throw 'Something went wrong obtaining access token';
    }
    // Create a Firebase credential with the AccessToken
    const facebookCredential = auth.FacebookAuthProvider.credential(data.accessToken);
    // Sign-in the user with the credential
    return auth().signInWithCredential(facebookCredential);
  }
const FBAuthentication = () => {
    return (
        <View>
         <LoginButton
          onLoginFinished={
            (error, result) => {
                console.log("login.");
              if (error) {
                console.log("login has error: " + JSON.stringify(result,null,2));
              } else if (result.isCancelled) {
                console.log("login is cancelled.");
              } else {
                AccessToken.getCurrentAccessToken().then(
```

```
            (data) => {
              console.log(data.accessToken.toString())
            }
          )
        }
      }
    }
    onLogoutFinished={() => console.log("logout.")}/>
  </View>
 );
};
export default FBAuthentication;
```

# Table View :

- Install the module using the below code.

```
npm install react-native-table-component
```

## Example :

```
import React, { useState } from "react";
import { View, Text } from "react-native";
import { Table, Row, Rows } from 'react-native-table-component';
const App = () => {
        const header = ['heading 1', 'heading 2', 'heading 3']
        const data = [
                ['A', 'B', 'C'],
                ['D', 'E', 'F'],
                ['G', 'H', 'I']
        ]
        return (
                <View style={{ marginTop: 200 }}>
                        <Text style={{ fontSize: 18 }}> React Native Table</Text>
                        <Table borderStyle={{ borderWidth: 2,
                                borderColor: '#c8e1ff' }}>
                                <Row data={header} />
                                <Rows data={data} />
                        </Table>
                </View>
        );
};
export default App;
```

# Google Map :

- Generate Google map API key.
- In google map to add your project then download & install library which is below.

```
npm i react-native-maps
```

- After downloading the library then executing the below code in your project.

```
import React from 'react';
import {  StyleSheet, View } from 'react-native';
import MapView, { Marker } from 'react-native-maps';
const GoogleMap_Demo = () => {
   return (
      <View style={styles.container}>
         <MapView style={styles.mapStyle}>
         </MapView>
      </View>
   );
};
export default GoogleMap_Demo;
const styles = StyleSheet.create({
   container: {
      position: 'absolute',
      top: 0,
      left: 0,
      right: 0,
      bottom: 0,
      alignItems: 'center',
      justifyContent: 'flex-end',
   },
   mapStyle: {
      position: 'absolute',
      top: 0,
      left: 0,
      right: 0,
      bottom: 0,
   },   });
```

# Google Find Location :

```
import React, { useEffect, useState } from 'react';
import { SafeAreaView, StyleSheet, View } from 'react-native';
import GetLocation from 'react-native-get-location'
import MapView, { Marker } from 'react-native-maps';

const GooglePlacesInput = () => {

    const [currentLongitude, setCurrentLongitude] = useState(0);
    const [currentLatitude, setCurrentLatitude] = useState(0);

    useEffect(() => {
        GetLocation.getCurrentPosition({
            enableHighAccuracy: true,
            timeout: 15000,
        })
            .then(location => {
                console.log(location);
                const currentLongitude = location.longitude;
                const currentLatitude = location.latitude;
                setCurrentLongitude(currentLongitude);
                setCurrentLatitude(currentLatitude);
                console.log("currentLongitude " + currentLongitude)
                console.log("currentLatitude " + currentLatitude)
            })
            .catch(error => {
                console.warn(error);
            })

    }, []);
    return (
        <SafeAreaView style={{ flex: 1 }}>
            <View style={styles.container}>
                <MapView
                    style={styles.mapStyle}>
                    <Marker
                        coordinate={{
                            // longitude: 72.5111511,
                            // latitude: 23.0360275
                            latitude: currentLatitude,
                            longitude: currentLongitude
                        }}
```

```
                    title={'Your Location'}
                    description={'Location '} />
                </MapView>
            </View>
        </SafeAreaView>
    );
};

export default GooglePlacesInput;

const styles = StyleSheet.create({
    btn: {
        alignItems: 'center',
        backgroundColor: 'blue',
        padding: 10,
        marginTop: 16,
        marginLeft: 35,
        marginRight: 35,
    },
    mapStyle: {
        position: 'absolute',
        top: 0,
        left: 0,
        right: 0,
        bottom: 0,
    },
    container: {
        position: 'absolute',
        top: 0,
        left: 0,
        right: 0,
        bottom: 0,
        alignItems: 'center',
        justifyContent: 'flex-end',
    },
})
```