----------React-Native for windows----------

1: set up the Your jdk andriod and node and react enviroment
2: Refer from Udemy course go and see the instalation
3: go to the command line and see the version of your node

node -v

4: And install your react by doing

install -g react-native-cli

5: if you want to change the folder

E:\>cd E:\React-Native-project

6: install react-native project called albums

E:\React-Native-project>react-native init albums

7: install the path of andriod go to the C drive go to karan select view select hidden
go to Appdata go to local go to andriod go to sdk go inside and copy the entire path and paste into the ANDRIOD_HOME values

C:\Users\karan\AppData\Local\Android\Sdk

8: go to command

E:\React-Native-project>cd albums

E:\React-Native-project\albums>npm install ----> install npm

9: E:\React-Native-project\albums>react-native -v
react-native-cli: 2.0.1
react-native: 0.63.2

install remaining efficient gradle scripts

10: restart your computer

11: go to the command and start the app

react-native start

12: for VS code

npm install -g eslint

13: Add the extensions of eslint in VS code

14: npm install --save-dev eslint-config-rallycoding

15: go to VS code add the file

16: for run the project write

Edit with WPS Office

react-native run-android

17:

.eslintrc

inside this file extends the rallycoding

```
{
    "extends": "rallycoding"
}
```

16: React-native Component

there are 4 part which we need to remember

part 1: Import libraries we need to create a component
part 2: create a component-a function that return some 'jsx'
part 3: Create a stylesheet to style our component: for styling the layout for component
part 4: Export the component so we can use it elsewhere in our project

17: lets start with creating the component

create a folder screens inside the src folder
create a file name ComponentsScreen.js inside the screens folder

import React from 'react'; ---> react library tells how diffrent component works toghter
import {Text, Stylesheet} from 'react-native'; --> react-native library know how to take the information from those component and show the content to actual device

// now create the componet

----------difference between React & React-native libraries----------

React: Knows how a compoent should behave
React-native: Know how to take the output from a component
React:Knows how to take a bunch of components and make them work togehter
React-native: provide default core component(image,text,style)

note: JSX is the Extension of javaScript langauge that is use to write the react component

16: Starting with index.js

Edit with WPS Office

```
// Import a library to help create a component

import React from 'react';
import ReactNative from 'react-native';


// Create a component


const App = () => {

return (

   <Text>Some Text </Text>

);

};


// Render it to the device
ReactNative.AppRegistry.registerComponent('albums', () => App);
```

AFTER DESTRUCTING THE IMPORT

```
// Import a library to help create a component

import React from 'react';
import {Text,AppRegistry} from 'react-native';


// Create a component


const App = () => (                    // Before one is also good with return but its a choice to
choose

   <Text>Some Text </Text>

);


// Render it to the device
AppRegistry.registerComponent('albums', () => App);


// Refresh your emulator

by pressing R R
```

Edit with WPS Office

17: Create the Header section of the app

Create src folder inside the project
Create components folder inside the src
Creare header.js file inside the components

```
// Import the libraries for making the components

import React from 'react';
import { Text } from 'react-native';

// Making the component

const Header = () => {
return <Text>Albums!</Text>

};

// Available the component for the other parts of the app

export default Header;
```

18: Component Nasty

CN: Means we take one component and place it inside another component

index.js

```
// Import a library to help create a component


import React from 'react';
import { AppRegistry } from 'react-native';    /*If we working with the some component that we want to render */
import {Header} from './android/app/src/components/header' /* Import the Header the component from */
import App from './App';


const App = () => (
  <Header />
);



// Render it to the device
//ReactNative.AppRegistry.registerComponent('albums', () => App);
// We change according to ourself we import the AppRegistry upwards


AppRegistry.registerComponent('albums', () => App);
```

header.js

Edit with WPS Office

```
import React from 'react';
import { Text, View } from 'react-native';      // Note: view tag is used for helping the positning
of components just like Div Section clearfix



// Making the component

const Header = () => {

      const {textStyle} = Styles;


return (

   <view>
     <Text style={textStyle}>Albums!</Text>
   </view>

);


};

const Styles = {

      textStyle: {

            fontSize: 20
      }

};

// Available the component for the other parts of the app

export default Header;
```

----------React-native expo for App ----------

1:Download the rn-starter
2: And Git bit
3: put the rn-starter into fresh folder in C drive
4: Extract the files
5: Go to cmd
6: npm install
7: allow yes for expo-cli
8: npm start
9: go to your pyshical mobile download Expo App from play store
10: scan your code

11: go to Vs code project
12: make a ComponentsScreen.js file inside the src inside screens

Edit with WPS Office

ComponentsScreen.js

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

const ComponentsScreen = () => {
return <Text style = {styles.textStyle}> This is Components Screen</Text>;

};

const styles    = StyleSheet.create({

      textStyle: {

            fontSize: 30
      }

});

export default ComponentsScreen;
```

HomeScrren.js

```
mport React from "react";
import { Text, StyleSheet } from "react-native";      ---> Text,View,image etc are the primitive react element

const HomeScreen = () => {
   return <Text style={styles.text}>Hey Hrithik</Text>;
};

const styles = StyleSheet.create({
   text: {
      fontSize: 30
   }
});

export default HomeScreen;
```

App.js

```
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
import HomeScreen from "./src/screens/HomeScreen"; ----------> import the HomeScreen
import ComponentsScreen from './src/screens/ComponentsScreen';----------> import the ComponentScreen

const navigator = createStackNavigator( ----------> StackNavigator is allow to run your scrren in your mobile
   {
      Home: HomeScreen,------> for HomeScreen
      Components: ComponentsScreen      ----> for ComponentsScreen
   },
```

Edit with WPS Office

```
    {
      initialRouteName: "Components",    ----> whatever you want to run write your route name
      defaultNavigationOptions: {
        title: "App"
      }
    }
);

export default createAppContainer(navigator);
```

Note: if it is not realoded then exit from your server start again

Q: What's Html in React?
Ans: JSX it is 'Dialect' of JS, it is not actual JS but the JSX code we write that pass inside the React-native bundler that
running inside the terminal. That bundler using the tool 'Babel' that turn or convert the JSX code into plain javascript code

Q: Which tool is used to convert JSX code into JS code?
Ans: Babel

Q: What is 'appNavigator' in the 'App.js' file?
Ans: it is tool from library called 'App Navigations' that is used to Screen different Screens to user

Q: What is props in JSX?
Ans: We configure diffrenet JSX elements using 'props'
props is also called as properties
props system is all about pass our data from a parent to a child

Q: using js variable in JSX
Ans: For ex

```
    const greeting = 'Hey' ---> variable of Js
                                    ---> You want to show in JSX

    return(

      <Text> {greeting} </Text>      ----> use allways {} to run the variable of JS in JSX
);
```

Q: Can we use inline Css and how it should work ?

Instead of Creating Styles component

```
    const HomeScreen = () => {
    return <Text style={styles.text}>Hey Hrithik</Text>;
};

const styles = StyleSheet.create({
```

Edit with WPS Office

```
  text: {
    fontSize: 30
  }
});
```

Use your Style as inline

```
 const HomeScreen = () => {
  return <Text style={{fontSize: 30 }}>Hey Hrithik</Text>; ---> it works in {{}} pattern
};
```

1: create ListScreen.js inside Screens

Note: FlatList is used as a prop to store the data of array
Also required to pass 'renderitem' prop - fun that will turn each indvidual item into elements

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


const ListScreen = () =>{
    const friends = [

        { name: 'Friend #1' },
        { name: 'Friend #2' },
        { name: 'Friend #3' },
        { name: 'Friend #4' },
        { name: 'Friend #5' },
        { name: 'Friend #6' },
        { name: 'Friend #7' },
        { name: 'Friend #8' }
    ];

return(

 <FlatList
  horizontal                                    /* show the list horizontal*/
  showsHorizontalScrollIndicator = {true}    /*perform the Scrool indicator*/
  keyExtractor = {friend => friend.name }    /* KeyExtractor use as passing a unique to each item */
  data={friends}
  renderItem={({item}) => {

  return <Text style = {Styles.textStyle}> {item.name} </Text>;

  }}
   />

  //Element ===={item: {name: 'friend#1'},index: 0}
  // item ==={name: 'friend#1'}

);
};
```

Edit with WPS Office

```
const Styles = StyleSheet.create({

      textStyle: {

            marginVertical: 30
      }

});

export default ListScreen;
```

-

--> (Nevegate between the screens)

1: Add button to homeScreen.js for navigating between screens

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

const HomeScreen = props => {


   return (

   <View>
        <Text style={styles.text}>Hey Hrithik</Text>
        <Button

          onPress={() => props.navigation.navigate('Components') }
          title ="Go to Components Screen"

           />
          <Button

           onPress={() => props.navigation.navigate('List')}
           title ="Go to List Screen"

          />


   </View>

   );
};

const styles = StyleSheet.create({
   text: {
      fontSize: 30
   }
});

export default HomeScreen;
```

Edit with WPS Office

---> (Destructing Your Props)

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

const HomeScreen = ({ navigation }) => {                                    --------> Apply navigation


   return (

   <View>
       <Text style={styles.text}>Hey Hrithik</Text>
       <Button

         onPress={()  =>  navigation.navigate('Components')  }                    ----------> delete
navigation
         title ="Go to Components Screen"

          />
         <Button

          onPress={() => navigation.navigate('List')}
          title ="Go to List Screen"

         />


   </View>

   );
};

const styles = StyleSheet.create({
   text: {
      fontSize: 30
   }
});

export default HomeScreen;
```

---> (Adding Image Screen with Image Details Screen)

1: Add your image into assest folder
2: Add ImageScreen.js file inside Screens folder
3: Add Components folder inside src
4: Add ImageDetail.js file inside Components folder
5: Add ImageScreen Path to App.js

ImageScreen.js

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

Edit with WPS Office

```
import ImageDetails from '../components/ImageDetails';


const ImageScreen = () => {

      return(

        <View>

        <ImageDetails
          title ="Beach"
          imageSource = {require('../../assets/beach.png')}
          score={10} />

        <ImageDetails
          title ="Forest"
          imageSource = {require('../../assets/forest.png')}
          score={9} />

        <ImageDetails
          title ="Mountain"
          imageSource = {require('../../assets/mountain.png')}
          score={8} />

        </View>

      );
};

const Styles = StyleSheet.create({


});

export default ImageScreen;
```

ImageDetails.js

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


const ImageDetails = props =>{

      return(

        <View>

        <Image source = {props.imageSource} />
      <Text style={Styles.ImageText}>{props.title}</Text>
      <Text>Image Score-{props.score}</Text>
        </View>

      );
};
```

Edit with WPS Office

```
const Styles = StyleSheet.create({

    ImageText:{

       fontSize: 40

    }

});

export default ImageDetails;
```

HomeScreen.js

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

const HomeScreen = props => {

   return (
      <View>
      <Text style = {Styles.textStyle}> Hey Hrithik!</Text>
      <Button
        onPress = {() => props.navigation.navigate('Components')}
        title = "Go to Component Screen"    />

      <Button
        onPress = {() => props.navigation.navigate('List')}
        title = "Go to List Screen"    />

      <Button
        onPress = {() => props.navigation.navigate('Image')}
        title = "Go to Image Screen"    />



      </View>




   );
};


const Styles = StyleSheet.create({

   textStyle:{

      fontSize: 30


   }
```

Edit with WPS Office

```
});

export default HomeScreen;
```

App.js

```
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
import HomeScreen from "./src/screens/HomeScreen";
import ComponentsScreen from './src/screens/ComponentsScreen';
import ListScreen from './src/screens/ListScreen';
import ImageScreen from './src/screens/ImageScreen';

const navigator = createStackNavigator(
  {
    Home: HomeScreen,
    Components: ComponentsScreen,
    List: ListScreen,
    Image: ImageScreen,

  },
  {
    initialRouteName: "Home",
    defaultNavigationOptions: {
      title: "App"
    }
  }
);

export default createAppContainer(navigator);
```

Q: What is State in React-native?

Ans: System to track a piece of data that will change over a time.
if the data changes our app will be 'rerender'

Q: What is useState in React-native?

Ans: useState is Hook that add a functionality to the components

Q: What is StateCounter in React-native?

Ans: Is used to update a state value in function

Q: How many type of Components?

Ans: There are of Many type but we use Function-based Component
Class-based Component

Edit with WPS Office

Note: When a Component is rerendered all the of its children get rerendered

---

-----> (Working with the State)
-----> (Counter App)

1: Add the CounterScreen.js inside the Screens folder
2: import with useState

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


const CounterScreen = () =>{

      // Dont do this
    // let counter = 0;


  // use Usestate function
 // set the useState intially at 0 default value
 // genrally we declare array as color = ['red','blue']

  const [counter, setCounter] = useState(0);      // This one is also declaring variable as array
set
                                                          // This is Function-based-component
       return (
         <View>

            <Button
              title = "Increase"
              onPress = {() => {

                // Dont do this
                // Counter ++

                setCounter (counter + 1);    // do this

              }} />


            <Button
              title = "Decrease"
              onPress = {() => {

                 // Dont do this
                // Counter ++

                 setCounter (counter - 1);    // do this
```

Edit with WPS Office

```
            }} />

            <Text>Current Count: {counter}</Text>
        </View>

  );


};

const Style = StyleSheet.create({



});

export default CounterScreen;
```

---->(Color App)

1: Add ColorScreen.js inside Screen folder
2: Do the Bioler plate

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

const ColorScreen = () => {
      const [colors, setColor] = useState([]);

            return (

            <View>

                  <Button
                    title = "Add Color"
                    onPress ={() =>{

                          setColor ([...colors, randomRgb()]);

                    } }/>
                    <View style ={{ height: 100, width: 100, backgroundColor: randomRgb() }} />
            </View>
      );


};

const randomRgb = () =>{

    const red = Math.floor (Math.random() * 256);
    const green = Math.floor (Math.random() * 256);
```

Edit with WPS Office

```
    const blue = Math.floor (Math.random() * 256);

    return `rgb(${red}, ${green}, ${blue})`;

};


const Styles = StyleSheet.create({



});

export default ColorScreen;
```

1: Adding the flatlist into colors App

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

const ColorScreen = () => {
      const [colors, setColor] = useState([]);

            return (

            <View>

                  <Button
                    title = "Add Color"
                    onPress ={() =>{

                          setColor ([...colors, randomRgb()]);

                    } }/>

                    <FlatList
                       keyExtractor = {item => item}
                       data={colors}
                       renderItem={({item}) => {
                         return(
                         <View style ={{ height: 100, width: 100, backgroundColor: item }} />
                         );
                      }}/>
            </View>
      );

};
```

Edit with WPS Office

```
const randomRgb = () =>{

    const red = Math.floor (Math.random() * 256);
    const green = Math.floor (Math.random() * 256);
    const blue = Math.floor (Math.random() * 256);

    return `rgb(${red}, ${green}, ${blue})`;


};



const Styles = StyleSheet.create({




});


export default ColorScreen;
```

1: (color App 2)

--> Add SquareScreen.js inside screens folder
--> Add ColorCounter.js inside components folder
--> In this app we pass callback function from parent to children

SquareScreen.js

```
import React, { useState } from 'react';
import {View,Text,StyleSheet} from 'react-native';
import ColorCounter from '../components/ColorCounter';

const COLOR_INCREMENT = 15;
const SquareScreen = () =>{
      const [red, SetRed] = useState (0);
      const [green, SetGreen] = useState (0);
      const [blue, SetBlue] = useState (0);
 return (
      <View>

      <ColorCounter
         color= "red"
         onIncrease={()=>SetRed (red + COLOR_INCREMENT)}
         onDecrease={()=>SetRed (red - COLOR_INCREMENT)}
           />
      <ColorCounter
        color="green"
        onIncrease={()=>SetGreen (green + COLOR_INCREMENT)}
        onDecrease={()=>SetGreen (green - COLOR_INCREMENT)}
          />
      <ColorCounter
```

Edit with WPS Office

```
        color="blue"
        onIncrease={()=>SetBlue (blue + COLOR_INCREMENT)}
        onDecrease={()=>SetBlue (blue - COLOR_INCREMENT)}/>

        <View style={{
              height: 100,
              width:100,
              backgroundColor: `rgb(${red},${green},${blue})`}}
        />

      </View>
  );
};


const Styles = StyleSheet.create({


});

export default SquareScreen;
```

ColorCounter.js

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


const ColorCounter = ({color, onIncrease,onDecrease}) =>{
  return (
      <View>
      <Text>{color}</Text>

      <Button
         onPress = {() => onIncrease()}
         title = {`Increase ${color}`}
      />
      <Button
         onPress = {() => onDecrease()}
         title = {`Decrease ${color}`}
      />
      </View>
  );

};


const Styles = StyleSheet.create({


});

export default ColorCounter;
```

Edit with WPS Office

1: (Add some color limit in color App)

SquareScreen.js

```
import React, { useState } from 'react';
import {View,Text,StyleSheet} from 'react-native';
import ColorCounter from '../components/ColorCounter';
import { Switch } from 'react-native-gesture-handler';

const COLOR_INCREMENT = 15;
const SquareScreen = () =>{
      const [red, SetRed] = useState (0);
      const [green, SetGreen] = useState (0);
      const [blue, SetBlue] = useState (0);


const SetColor =(color,change) =>{
    // color === 'red','green','blue'
  // change === '+15','-15'

  switch (color) {
        case 'red':
              red + change > 255 || red + change < 0 ? null : SetRed(red + change);
              return;
        case 'green':
               green + change > 255 || green + change < 0
               ? null
               :SetGreen(green + change);
               return;
        case 'blue':
               blue + change > 255 || blue + change < 0
               ? null
               :SetBlue(red + change);
               return;

               default:
               return;
    }
  };
 return (
     <View>

     <ColorCounter
        color= "red"
        onIncrease={()=>SetColor('red',COLOR_INCREMENT)}
        onDecrease={()=>SetColor('red', - 1 * COLOR_INCREMENT)}
          />
     <ColorCounter
       color="green"
       onIncrease={()=>SetGreen (green + COLOR_INCREMENT)}
       onDecrease={()=>SetGreen (green - COLOR_INCREMENT)}
         />
```

Edit with WPS Office

```
      <ColorCounter
        color="blue"
        onIncrease={()=>SetBlue (blue + COLOR_INCREMENT)}
        onDecrease={()=>SetBlue (blue - COLOR_INCREMENT)}/>

        <View style={{
              height: 100,
              width:100,
              backgroundColor: `rgb(${red},${green},${blue})`}}
        />

      </View>
  );
};


const Styles = StyleSheet.create({


});

export default SquareScreen;
```

Q What is Reducer in React-native?
Ans: function that manage changes to an object
function that gets called with two object

--(Working with reducer)

```
import React, { useState, useReducer } from 'react';
import {View,Text,StyleSheet} from 'react-native';
import ColorCounter from '../components/ColorCounter';
import { Switch } from 'react-native-gesture-handler';

const COLOR_INCREMENT = 15;
const reduce = (state, action) =>{

      //state === {red: number,green:number,blue:number}
    //action === {colorTochange: 'red' || 'green' || 'blue', amount: 15||-15}

    switch(action.colorToChange){
          case 'red':
          return state.red + action.amount > 255 || state.red + action.amount < 0
                ? state
                :{...state, red: state.red + action.amount};
          case 'green':
          return state.green + action.amount > 255 || state.green + action.amount < 0
                ? state
                :{...state, green: state.green + action.amount};

          case 'blue':
          return state.blue + action.amount > 255 || state.blue + action.amount < 0
                ? state
                :{...state, blue: state.blue + action.amount};
```

Edit with WPS Office

```
            default:
                return state;

    }


};

const SquareScreen = () =>{
      const [state, dispatch] = useReducer (reduce, {red:0,green:0,blue:0} );
      const {red, green, blue} = state;

 return (
      <View>

      <ColorCounter
         color= "red"
         onIncrease={()=> dispatch({colorToChange: 'red', amount: COLOR_INCREMENT})}
         onDecrease={()=> dispatch({colorToChange: 'red', amount: -1 * COLOR_INCREMENT})}
          />
      <ColorCounter
        color="green"
        onIncrease={()=> dispatch({colorToChange: 'green', amount: COLOR_INCREMENT})}
        onDecrease={()=> dispatch({colorToChange: 'green', amount: -1 * COLOR_INCREMENT})}
          />
      <ColorCounter
        color="blue"
        onIncrease={()=> dispatch({colorToChange: 'blue', amount: COLOR_INCREMENT})}
        onDecrease={()=> dispatch({colorToChange: 'blue', amount: -1 * COLOR_INCREMENT})}
        />

        <View style={{
            height: 100,
            width:100,
            backgroundColor: `rgb(${red},${green},${blue})`}}
        />

      </View>
 );
};


const Styles = StyleSheet.create({


});

export default SquareScreen;
```

1: (working with convention)

it consist of Type -> String that discribe the exact change operation we want to make
payload -> some data that is critical to the change operation

{type: 'change_red', payload: 15}

SquareScreen.js

Edit with WPS Office

```
import React, { useState, useReducer } from 'react';
import {View,Text,StyleSheet} from 'react-native';
import ColorCounter from '../components/ColorCounter';
import { Switch } from 'react-native-gesture-handler';

const COLOR_INCREMENT = 15;
const reduce = (state, action) =>{

      //state === {red: number,green:number,blue:number}
    //action === {type: 'change_red' || 'change_green' || 'change_blue', payload: 15||-15}

    switch(action.type){
          case 'change_red':
          return state.red + action.payload > 255 || state.red + action.payload < 0
                ? state
                :{...state, red: state.red + action.payload};
          case 'change_green':
          return state.green + action.payload > 255 || state.green + action.payload < 0
                ? state
                :{...state, green: state.green + action.payload};

          case 'change_blue':
          return state.blue + action.payload > 255 || state.blue + action.payload < 0
                ? state
                :{...state, blue: state.blue + action.payload};

          default:
                return state;

    }


};

const SquareScreen = () =>{
      const [state, dispatch] = useReducer (reduce, {red:0,green:0,blue:0} );
      const {red, green, blue} = state;

 return (
      <View>

      <ColorCounter
         color= "red"
         onIncrease={()=> dispatch({type: 'change_red', payload: COLOR_INCREMENT})}
         onDecrease={()=> dispatch({type: 'change_red', payload: -1 * COLOR_INCREMENT})}
          />
      <ColorCounter
        color="green"
        onIncrease={()=> dispatch({type: 'change_green', payload: COLOR_INCREMENT})}
        onDecrease={()=> dispatch({type: 'change_green', payload: -1 * COLOR_INCREMENT})}
          />
      <ColorCounter
        color="blue"
        onIncrease={()=> dispatch({type: 'change_blue', payload: COLOR_INCREMENT})}
        onDecrease={()=> dispatch({type: 'change_blue', payload: -1 * COLOR_INCREMENT})}
        />

        <View style={{
              height: 100,
```

Edit with WPS Office

```
            width:100,
            backgroundColor: `rgb(${red},${green},${blue})`}}
        />

      </View>
  );
};


const Styles = StyleSheet.create({


});

export default SquareScreen;
```

1: (TextScreem.js)

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

const TextScreen = () => {
      const [name, setName] = useState ('');
      return(
            <View>
                  <TextInput
                  style={Styles.input}
                  autoCapitalize="none"
                  autoCorrect={false}
                  value = {name}
                  onChangeText = {newValue => setName(newValue)}

            />
            <Text>My Name is: {name}</Text>
        </View>
      );
};

const Styles = StyleSheet.create ({

      input: {

            margin: 15,
            borderColor: 'black',
            borderWidth: 1
      }


});

export default TextScreen;
```

Edit with WPS Office

1:(Add password field to Screen)

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

const TextScreen = () => {
      const [password, setPassword] = useState ('');
      return(
            <View>
                  <Text>Enter Your Password:</Text>
                  <TextInput
                  style={Styles.input}
                  autoCapitalize="none"
                  autoCorrect={false}
                  value = {password}
                  onChangeText = {newValue => setPassword(newValue)}

            />
            {password.length < 4 ? <Text>Password Must be 4 characters</Text> : null }
        </View>
      );
};

const Styles = StyleSheet.create ({

      input: {

            margin: 15,
            borderColor: 'black',
            borderWidth: 1
      }


});

export default TextScreen;
```

_---

1:(layout) consist of
--> Box object Model
--> Flex Box
-->Position

Below Three properties assigned for parents to set their children

(working with flexbox)

1: alignItem: 'flex-start' --> Takes inside content of view to left
2: alignItem: 'center' ----> Takes to center
3: alignItem: 'flex-end' ---> Takes to right

Edit with WPS Office

4: alignItem: 'stretch' ---> default

(flex direction)

1:flex-direction: 'column' --> sets to horizontal
2:flex-direction: 'row' --> sets to vertical

(justify content) --> workkflow depends on flex direction

1: justifyContent: 'flex-start'
2: justifyContent: 'center'
3: justifyContent: 'flex-end'
4: justifyContent: 'space-between'
5: justifyContent: 'space-around'

Below Two properties of Flex Box are assigned to children to work

(flex) --> Also Based on parent flex-direction

suppose we have 3 children and one parent

<view> ----> Add flex-direction: 'row'

<Text> Child:1 </Text>
<Text> Child:2 </Text> --> suppose we add flex: 1 ---> Then it take max space to set between others
<Text> Child:3 </Text>

</view>

(alignSelf) -> Based on parents alignItem

suppose we have 3 children and one parent

<view> ----> Add alignItem: 'center'

<Text> Child:1 </Text>
<Text> Child:2 </Text> --> suppose we add alignSelf: 'flex-end' ---> Then it take max space to set between others
<Text> Child:3 </Text>

</view>

alignSelf: 'flex-start'
alignSelf: 'center'
alignSelf: 'flex-end'

Edit with WPS Office

(Position) ---> Two types based on Parents

position: 'relative' ----> default
position: 'absolute' ---> ignored by other child properties, it just override to other one

If

position: 'absolute'
top: 0 ----------> all are for distance of edges between the each child component
bottom: 0
left: 0
right: 0

instead of write this 5 properties Write

...StyleSheet.absoulteFillObject

(Working with Restaurant Search projetct)

-->How to Create project
--> There are two different way

1:expo-cli
2:react-native-cli

Q: Reason behind expo-cli?
Ans: Add in ton default config to use feature common in aps like icons,videos,better camera use etc

Q: Reason behind react-native-cli?
Ans: Default cli to genrate a project,
Requires lots of extra work to add in common features.

Q: React-Navigation provides what?

Ans: Is Third Party Navigation dependence
Provide diffrent object for navigating user through out the app
3 important objects are

1: StackNavigator --> Screen Take User to Another Scrren by press Button and Also Takes Back Again in Same Screen by click back arrow
2: BottomTabNavigator --> click at the end of screen their are button to navigate between the screens
3: Drawer Navigator --> click at uperleft most of screen their are button or link to navigate between the screens

They have their own screen

Note:(intall npm install react-navigation)

Edit with WPS Office

React Navigation Fix
updated 6-26-2020

Important Message about React Navigation v5
This project will be using the v4 version of React Navigation and not the v5 version which was released a few months ago. To date, this code still works as expected. The v4 version is still important to know and understand as it is used widely in existing codebases in many organizations. Stephen is aware of the interest in an update and will hopefully post a supplement at some time in the future. In the meantime, I highly encourage you to continue on with the course as it is so that you fully understand how React Navigation works. Once you have finished the course, you can use this is a self-study opportunity to attempt a migration of your finished working projects to v5. The docs provide some great resources on this: https://reactnavigation.org/docs/upgrading-from-4.x/.

React Navigation v4 Installation
note: You cannot mix Yarn and npm in the same project as it will break your dependencies. You must consistently use the same package manager. If you have yarn installed on your computer it will be used by default to generate the project. In this case, you must use Yarn to install your dependencies.

1. Install React Navigation

npm install react-navigation

or

yarn add react-navigation

2. Install Dependencies

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

3. Install React Navigation Stack

npm install react-navigation-stack @react-native-community/masked-view

or

yarn add react-navigation-stack @react-native-community/masked-view

4. Start the app and clear cache with expo r -c

Update Imports
Our imports in the upcoming lecture will now look like this:

```
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
```

Errors?
If you are still seeing errors and complaints about versions, you can try an upgrade:

1. expo upgrade

2. expo r -c

Edit with WPS Office

React Navigation v4 Docs:
https://reactnavigation.org/docs/4.x/getting-started

1: Delete All inside the App.js
2: Buid with Scratch
3: Build new folder src inside src Build new folder Screen
4: Inside Screen Make file named as SearchScreen.js
5: Make Folder inside src components for resuable screens in our project
6: Make file inside components named as SearchBar.js
7: Go to github.com/expo/vector-icon for choosing icon for your project
8: select the icon and import their library into your project
9:

```
    const SearchBar = () =>{

      return (
            <View style ={Styles.background}>
                  <Feather name="search" size={24} color="black" />    ---> inside Your view Name
your icon and set the properties
                  <Text>Search Screen</Text>

            </View>
      );

};
```

10: Add some style to SearchBar

```
import React from 'react';
import {View,TextInput,StyleSheet} from 'react-native';
import { Feather } from '@expo/vector-icons';


const SearchBar = () =>{

      return (
            <View style ={Styles.background}>
                  <Feather name="search" size={24} color="black" style={Styles.iconStyle}/>
                  <TextInput style={Styles.inputText} placeholder="Search" />
            </View>
      );

};

const Styles = StyleSheet.create({

background:{
      marginTop: 15,
      backgroundColor: '#F0EEEE',
      height: 50,
      borderRadius: 5,
      marginHorizontal: 15,
      flexDirection: 'row'
},
inputText:{

      // borderColor:'black',
```

Edit with WPS Office

```
        // borderWidth: 1,
        flex: 1,
        fontSize: 18
},

iconStyle:{

        fontSize: 35,
        alignSelf: 'center',
        marginHorizontal: 15

}



});

export default SearchBar;
```

11: Set the SearchScreen by rendering States from parent to child by useState

```
import React, { useState } from 'react';
import {View,Text,StyleSheet} from 'react-native';
import SearchBar from '../components/SearchBar';

const SearchScreen = () =>{
        const [term, setTerm] = useState('');    ---> Two State one is current search Term and Next
is SetTerm to new search

        return (
                <View>
                        <SearchBar
                          term ={term} ----------------------------------------------> Apply term here
                          onTermChange = {newTerm => setTerm(newTerm)} ----------------> Apply
onTermChange and set new fun called newTerm and set the term
                            />
                        <Text>{term}</Text> ---> pass the term value inside the text for output

                </View>
        );

};

const Styles = StyleSheet.create({

SearchStyle:{


}


});

export default SearchScreen;
```

Edit with WPS Office

SearchBar.js

```
import React from 'react';
import {View,TextInput,StyleSheet} from 'react-native';
import { Feather } from '@expo/vector-icons';


const SearchBar = ({term,onTermChange}) =>{      -----> Two props

      return (
            <View style ={Styles.background}>
                  <Feather name="search" size={24} color="black" style={Styles.iconStyle}/>
                  <TextInput
                    style={Styles.inputText}
                    placeholder="Search"
                    autoCapitalize="none"
                    autoCorrect = {false}
                    value={term} ----------------------------------------------------------> set the value to term so we
can write inside searchBar
                    onChangeText={newTerm    =>    onTermChange(newTerm)}    ---------------------->
Apply onChangeText and set the function
                     />
            </View>
      );

};

const Styles = StyleSheet.create({

background:{
      marginTop: 15,
      backgroundColor: '#F0EEEE',
      height: 50,
      borderRadius: 5,
      marginHorizontal: 15,
      flexDirection: 'row'
},
inputText:{

      // borderColor:'black',
      // borderWidth: 1,
      flex: 1,
      fontSize: 18
},

iconStyle:{

      fontSize: 35,
      alignSelf: 'center',
      marginHorizontal: 15

}



});

export default SearchBar;
```

Edit with WPS Office

1:(working with final submit button in phoneKeypad)

Add OnTermSubmit

SearchScreen.js

```
const SearchScreen = () =>{
      const [term, setTerm] = useState('');

      return (
            <View>
                  <SearchBar
                    term ={term}
                    onTermChange = {newTerm => setTerm(newTerm)}
                    onTermSubmit = {() => console.log('Term is Submited')}
                       />
                  <Text>{term}</Text>

            </View>
      );

};
```

---

onEndEditing

SearchBar.js

```
const SearchBar = ({term,onTermChange, onTermSubmit}) =>{

      return (
            <View style ={Styles.background}>
                  <Feather name="search" size={24} color="black" style={Styles.iconStyle}/>
                  <TextInput
                    style={Styles.inputText}
                    placeholder="Search"
                    autoCapitalize="none"
                    autoCorrect = {false}
                    value={term}
                    onChangeText={onTermChange}
                    onEndEditing={onTermSubmit}       ------> Here
                      />
            </View>
      );

};
```

(Working with outside Api)

Q: What is fetch in API?
Ans: fetch is Built in function for making network request
Error Handling is littel bit Strange

Edit with WPS Office

Requires lots of wrapper code to make it work 'sensibly'

Q: What is axios in API?
Ans: Seprate librabry for making request
Easy to use,Sensible defaults
Increase our App size(very,very,Slightly)

Two Ways for Making network request in API

1 fetch
2 axois

Note:(first you install the package)

Any of them

npm install fetch
npm install axios

1: Make New folder api inside src
2: Make New file yelp.js inside api

---

yelp.js

```
import axios from 'axios';

export default axios.create({
  baseURL: 'https://api.yelp.com/v3/businesses',       -----> api url
  headers: {
    Authorization:       -----> api Key
      'Bearer
l2cmhf2ezRl6ZkcHWNYKoDpiaro1zlqUjwkA7nVxnWzryiTwFDk35PJiucoLhjjFY9ECD8GTBGaH
Bg5yv5YDLiszKQx8EMvm30ply0UWoHQOnYFjLozpYnZOx-UsXXYx'
});

// (note: Allways space between Bearer and Key otherwise it shows an error)
```

SearchScreen.js

```
import React, { useState } from 'react';
import { View, Text, StyleSheet } from 'react-native';
import SearchBar from '../components/SearchBar';
import yelp from '../api/yelp';

const SearchScreen = () => {
  const [term, setTerm] = useState('');
  const [results, setResults] = useState([]);

  const searchApi = async () => { ----------> Helper Function, Make
request Api Async() is used for permission
    const response = await yelp.get('/search', {       ----------> yelp.get get is http request
```

Edit with WPS Office

'/search' basically Api baseurl
params: { ----------> Assign the respons to response variable
limit: 50, ----------> second argument have pass the object called params with having keyvalue that we passed inside params
term, ----------> that automaticaly apped to baseUrl of Api
location: 'san jose'
}
});
setResults(response.data.businesses); ------> response.data is a response properties that have json data back from API, Businesses is array of object that we want to store
};

return (
<View>
<SearchBar term={term} onTermChange={setTerm} onTermSubmit={searchApi} />
<Text>Search Screen</Text>
<Text>We have found {results.length} results</Text>
</View>
);
};

const styles = StyleSheet.create({});

export default SearchScreen;

(Working with Error finding Message with Try and catch)

```
const SearchScreen = () => {
  const [term, setTerm] = useState('');
  const [results, setResults] = useState([]);
  const [errorMessage, SetErrorMessage] = useState('');     // component


  const searchApi = async () => {

    try {

        const response = await yelp.get('/search', {
            params: {
            limit: 50,
            term,
            location: 'san jose'
          }
        });

        setResults(response.data.businesses);
      } catch(err){
        SetErrorMessage ('Something Went Wrong');       // Set the error message
      }
  };
```

Edit with WPS Office

```
  return (
    <View>
      <SearchBar
        term={term}
        onTermChange={setTerm}
        onTermSubmit={searchApi}

        />
     {errorMessage ? <Text>{errorMessage}</Text>: null }       // call the errorMessage here
      <Text>We have found {results.length} results</Text>
    </View>
  );
};
```

(FLOW IN SEARCH SCREEN COMPONENT)

1: SearchScreen Function called
2: Search Api called Immediately
3: Make request to yelp Api
4: get search result ,call setter
5: update state cause component to rerenderd

(UseEffect()---> Hook func which adds the extra properties to components)

```
 useEffect(() => {     // useEffect() Hook function is used to when we want to run one time

    searchApi('pasta');

  }, []);    // Or want to run code many time depanding on 2 argument array
```

1: Make hooks folder inside src
2: Make useResults.js file inside hooks ------> for extract the hooks bussinness search logic

cut paste the code of searchScreen in useResults.js, Note: only the hooks logic code

useResults.js

```
import {useEffect,useState} from 'react';
import yelp from '../api/yelp';

export default () => {

    const [results, setResults] = useState([]);
    const [errorMessage, SetErrorMessage] = useState('');


    const searchApi = async searchTerm => { //Make request Api Async() is used for permission, yelp.get get is http request '/search' basically Api baseurl

      try {
```

Edit with WPS Office

```
            const response = await yelp.get('/search', {        // helper function, Assign the respons to response variable
                params: {    // keyvalue that append directly to '/search' baseUrl
                limit: 50,
                term: searchTerm,    // term:term
                location: 'san jose'
              }
            });

            setResults(response.data.businesses);            // response.data is a response properties that have json data back from API,
        } catch(err){
            SetErrorMessage ('Something Went Wrong');
        }
    };                                                  // Businesses is array of object that we want to store

    useEffect(() => {     // useEffect() Hook function is used to when we want to run one time

      searchApi('pasta');

    }, []);    // Or want to run code many time depanding on 2 argument array

    return [searchApi,results,errorMessage];
};
```

SearchScreen.js

```
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet } from 'react-native';
import SearchBar from '../components/SearchBar';
import useResults from '../hooks/useResults';

const SearchScreen = () => {
  const [term, setTerm] = useState('');
  const [searchApi,results,errorMessage] = useResults();    // useResults() that we extrat to hooks

  return (
    <View>
      <SearchBar
        term={term}
        onTermChange={setTerm} // {newTerm => setTerm(newTerm)}
        onTermSubmit={() => searchApi(term)}      // { () => searchApi()}

        />
      {errorMessage ? <Text>{errorMessage}</Text>: null }
        <Text>We have found {results.length} results</Text>
    </View>
  );
};

const styles = StyleSheet.create({});

export default SearchScreen;
```

Edit with WPS Office

1: (Make ResultsList.js inside the components)

```
import React from 'react';
import {View, Text, StyleSheet, FlatList } from 'react-native';
import ResultsDetail from '../components/ResultsDetail';

const ResultsList = ({title, results }) =>{

return (
<View style={Styles.container}>
  <Text style={Styles.title}>{title}</Text>
  <FlatList
     horizontal
     data={results}
     keyExtractor={(result) => result.id}
     renderItem={({ item }) => {

       return <ResultsDetail result={item} />


     }}
  />
</View>

)

};

const Styles = StyleSheet.create({

     title: {
          fontSize: 18,
          fontWeight: 'bold',
          marginLeft: 15,
          marginBottom: 5
     },
     container:{

          marginBottom: 15

     }

});

export default ResultsList;
```

2: (Make ResultsDetail.js inside the components)

```
import React from 'react';
import {View, Text, StyleSheet, Image } from 'react-native';
import { Entypo } from '@expo/vector-icons';
```

Edit with WPS Office

```
const ResultsDetail = ({ result }) =>{

return (
<View style={Styles.container}>

    <Image style={Styles.image} source={{uri: result.image_url}} />
    <Text> {result.name} </Text>
    <Text>{result.rating}  <Entypo  name="star-outlined"  size={21}  color="black"  />,
{result.review_count} Reviews </Text>
</View>
)


};

const Styles = StyleSheet.create({

      image:{

            width: 250,
            borderRadius: 4,
            height: 120
      },

      container:{

            marginLeft: 15

      }

});

export default ResultsDetail;
```

--

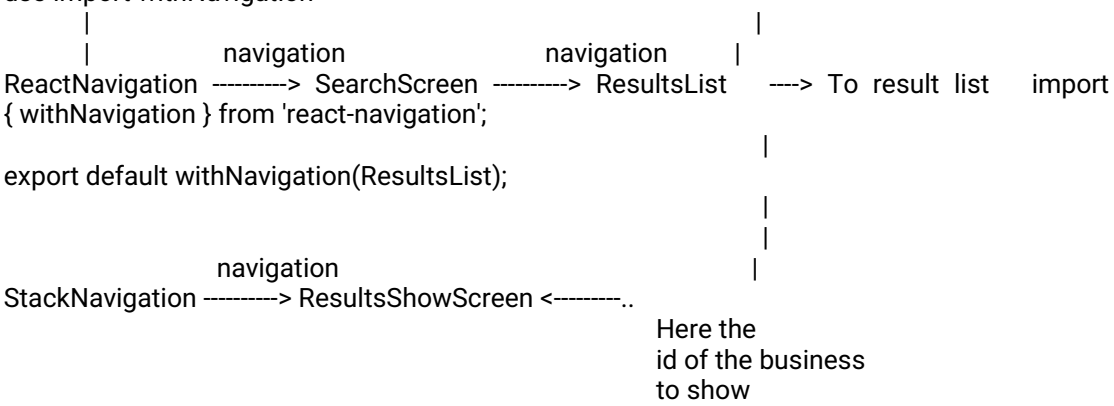
4: (Make ResultsShowScreen.js inside screens)

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

const ResultsShowScreen = () => {
   return (
      <View>
         <Text>Results Show</Text>
      </View>
   );
};

const styles = StyleSheet.create({});

export default ResultsShowScreen;
```

_____navigation_____ ----> for direct navigation

Edit with WPS Office

use import withNavigation



Note:(Delete navigation props and their properties inside SearchScreen.js if you use direct navigation with resultList)

```
const SearchScreen = ({ navigation }) => {    ----> delete this and change to = ()

 <ScrollView>
          <ResultsList
            results={filterResultsByPrice('$')}
            title="Cost Effective"
            navigation={navigation}          ----------> delete this
          />
```

ResultList.js

```
import React from 'react';
import {View, Text, StyleSheet, FlatList, TouchableOpacity } from 'react-native';
import ResultsDetail from '../components/ResultsDetail';
import { withNavigation } from 'react-navigation';

const ResultsList = ({title, results, navigation }) =>{

return (
<View style={Styles.container}>
  <Text style={Styles.title}>{title}</Text>
  <FlatList
     horizontal
     data={results}
     keyExtractor={(result) => result.id}
     renderItem={({ item }) => {

        return(
                  <TouchableOpacity onPress={() => navigation.navigate('ResultsShow' , {id: item.id } )}>   ---> Here give the id
                       <ResultsDetail result={item} />
                 </TouchableOpacity>


        )
```

Edit with WPS Office

```
      }}
  />
</View>

)

};

const Styles = StyleSheet.create({

      title: {
            fontSize: 18,
            fontWeight: 'bold',
            marginLeft: 15,
            marginBottom: 5
      },
      container:{

            marginBottom: 15

      }

});

export default withNavigation(ResultsList);
```

(End of Resturant App)

(Screens)

(App.js)

```
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
import SearchScreen from './src/screens/SearchScreen';
import ResultsShowScreen from './src/screens/ResultsShowScreen';

const navigator = createStackNavigator(
   {
      Search: SearchScreen,
      ResultsShow: ResultsShowScreen
   },
   {
      initialRouteName: "Search",
      defaultNavigationOptions: {
         title: "Business Search"
      }
   }
);

export default createAppContainer(navigator);
```

(SearchScreen.js)

Edit with WPS Office

```
import React, { useState } from 'react';
import { View, Text, StyleSheet, ScrollView } from 'react-native';
import SearchBar from '../components/SearchBar';
import useResults from '../hooks/useResults';
import ResultsList from '../components/ResultsList';

const SearchScreen = ({ navigation }) => {
  const [term, setTerm] = useState('');
  const [searchApi, results, errorMessage] = useResults();

  const filterResultsByPrice = price => {
    // price === '$' || '$$' || '$$$'
    return results.filter(result => {
      return result.price === price;
    });
  };

  return (
    <>
      <SearchBar
        term={term}
        onTermChange={setTerm}
        onTermSubmit={() => searchApi(term)}
      />
      {errorMessage ? <Text>{errorMessage}</Text> : null}
      <ScrollView>
        <ResultsList
          results={filterResultsByPrice('$')}
          title="Cost Effective"
          navigation={navigation}
        />
        <ResultsList
          navigation={navigation}
          results={filterResultsByPrice('$$')}
          title="Bit Pricier"
        />
        <ResultsList
          navigation={navigation}
          results={filterResultsByPrice('$$')}
          title="Big Spender"
        />
      </ScrollView>
    </>
  );
};

const styles = StyleSheet.create({});

export default SearchScreen;
```

(SearchBar.js)

Edit with WPS Office

```
import React from 'react';
import {View,TextInput,StyleSheet} from 'react-native';
import { Feather } from '@expo/vector-icons';


const SearchBar = ({term,onTermChange, onTermSubmit}) =>{

	return (
		<View style ={Styles.background}>
			<Feather name="search" size={24} color="black" style={Styles.iconStyle}/>
			<TextInput
			  style={Styles.inputText}
			  placeholder="Search"
			  autoCapitalize="none"
			  autoCorrect = {false}
			  value={term}
			  onChangeText={onTermChange}
			  onEndEditing={onTermSubmit}
			   />
		</View>
	);

};

const Styles = StyleSheet.create({

background:{
	marginTop: 15,
	backgroundColor: '#F0EEEE',
	height: 50,
	borderRadius: 5,
	marginHorizontal: 15,
	flexDirection: 'row',
	marginBottom: 10
},
inputText:{

	// borderColor:'black',
	// borderWidth: 1,
	flex: 1,
	fontSize: 18
},

iconStyle:{

	fontSize: 35,
	alignSelf: 'center',
	marginHorizontal: 15

}



});

export default SearchBar;
```

Edit with WPS Office

(ResultList.js)

```
import React from 'react';
import {View, Text, StyleSheet, FlatList, TouchableOpacity } from 'react-native';
import ResultsDetail from '../components/ResultsDetail';
import { withNavigation } from 'react-navigation';

const ResultsList = ({title, results, navigation }) =>{

      if (!results.length){     // Use for Showing the only actual possible result
            return null;
      }

return (

<View style={Styles.container}>

 <Text style={Styles.title}>{title}</Text>

 <FlatList
    horizontal
    data={results}
    keyExtractor={(result) => result.id}
    renderItem={({ item }) => {

       return(
                  <TouchableOpacity onPress={() => navigation.navigate('ResultsShow' , {id:
item.id } )}>
                        <ResultsDetail result={item} />
                  </TouchableOpacity>


       )

     }}
 />
</View>

)

};

const Styles = StyleSheet.create({

      title: {
            fontSize: 18,
            fontWeight: 'bold',
            marginLeft: 15,
            marginBottom: 5
      },
      container:{

            marginBottom: 15

      }

});

export default withNavigation(ResultsList);
```

Edit with WPS Office

(ResultDetail.js)

```
import React from 'react';
import {View, Text, StyleSheet, Image } from 'react-native';
import { Entypo } from '@expo/vector-icons';

const ResultsDetail = ({ result }) =>{

return (
<View style={Styles.container}>

    <Image style={Styles.image} source={{uri: result.image_url}} />
    <Text> {result.name} </Text>
    <Text>{result.rating}  <Entypo  name="star-outlined"  size={21}  color="black"  />,
{result.review_count} Reviews </Text>
</View>
)


};

const Styles = StyleSheet.create({

      image:{

            width: 250,
            borderRadius: 4,
            height: 120
      },

      container:{

            marginLeft: 15

      }

});

export default ResultsDetail;
```

(ResultsShowScreen.js)

```
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet, Image } from 'react-native';
import yelp from '../api/yelp';
import { FlatList } from 'react-native-gesture-handler';

const ResultsShowScreen = ({ navigation }) => {
   const [result , SetResult] = useState(null);
   const id = navigation.getParam('id');

   const getResult = async (id) =>{
```

Edit with WPS Office

```
  const response = await yelp.get(`/${id}`);
    SetResult(response.data);

  };

  useEffect ( () => {

      getResult(id);


  }, []);

  if (!result){

     return null;
  }
  return (
     <View>
       <Text>{result.name}</Text>
       <FlatList
        keyExtractor={(photo)=> photo}
        data={result.photos}
        renderItem={ ({item}) => {

          return <Image style={styles.Image}
                        source={{uri: item }}/>       // {{}} outer brackets is for javaScript
expreesion, And inner one is actual object.

        }}
          />
     </View>
  );
};

const styles = StyleSheet.create({

  Image:{


     height: 200,
     width: 300
  }

});

export default ResultsShowScreen;
```

(useResults.js)

```
import {useEffect,useState} from 'react';
import yelp from '../api/yelp';

export default () => {

     const [results, setResults] = useState([]);
     const [errorMessage, SetErrorMessage] = useState('');
```

Edit with WPS Office

```
    const searchApi = async searchTerm => { //Make request Api Async() is used for permission, yelp.get get is http request '/search' basically Api baseurl

        try {

            const response = await yelp.get('/search', {       // helper function, Assign the respons to response variable
                params: {    // keyvalue that append directly to '/search' baseUrl
                limit: 50,
                term: searchTerm,    // term:term
                location: 'san jose'
              }
            });

            setResults(response.data.businesses);            // response.data is a response properties that have json data back from API,
          } catch(err){
            SetErrorMessage ('Something Went Wrong');
          }
    };                                                  // Businesses is array of object that we want to store

    useEffect(() => {     // useEffect() Hook function is used to when we want to run one time

      searchApi('pasta');

    }, []);    // Or want to run code many time depanding on 2 argument array

    return [searchApi,results,errorMessage];
```

(yelp.js)

```
import axios from 'axios';

export default axios.create({
  baseURL: 'https://api.yelp.com/v3/businesses',
  headers: {
    Authorization:
      'Bearer
l2cmhf2ezRl6ZkcHWNYKoDpiaro1zlqUjwkA7nVxnWzryiTwFDk35PJiucoLhjjFY9ECD8GTBGaH
Bg5yv5YDLiszKQx8EMvm30ply0UWoHQOnYFjLozpYnZOx-UsXXYx'
  }
});

// (note: Allways space between Bearer and Key otherwise it shows an error)
```

[BLOG-APP]

Edit with WPS Office

1: When your project is ready

To run your project, navigate to the directory and run one of the following npm commands.

- cd blog
- npm start # you can open iOS, Android, or web from here, or run them directly with the commands below.
- npm run android
- npm run ios # requires an iOS device or macOS for access to an iOS simulator
- npm run web

Go for Anyone of them later

Now npm install react

1: Blog App is Based on CRUD -----IMP

2: There is one change One BLog Post Provider component is include as a data provider to other

3: Blog Post Provider Component Includes

React-Stack Navigation
IndexScreen
ShowScreen
CreateScreen
EditScreen
All Child Screen also

4: Basically Blog Post Provider is Redux Component , And in Blog App it is our Parent Componenet -----IMP

5: Now Wrape the All React-Stack Navigation with own custom component

Before App.js

```
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
import IndexScreen from './src/screens/IndexScreen';


const navigator = createStackNavigator({

  Index: IndexScreen

}, {

        initialRouteName: 'Index',
        defaultNavigationOptions: {
           title: 'Blogs'
    }
});

export default createAppContainer(navigator);
```

Edit with WPS Office

After App.js

```
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
import IndexScreen from './src/screens/IndexScreen';
import React from 'react';


const navigator = createStackNavigator({

  Index: IndexScreen

}, {

        initialRouteName: 'Index',
        defaultNavigationOptions: {
           title: 'Blogs'
    }
});

const App = createAppContainer(navigator); ----> Chnage is Here // Assign the Result to App Variable

export default () => {

    return <App /> -----> Change is Here    /// Own Custom Compoenet that include all stuff

};
```

Note:(if you see the error Then You might forget to import React upward)

---> INTRO ABOUT CONTEXT

Difference Between Props And Context

(Props)

1: Communication info from parent directly down to a child
2: Easy to Set up
3: To Communicate data down to multiple layes. We have to Write lots of code

(Context)

1: Moves Some information from a Parent to some nested child
2: Complicated to setup lots of special term
3: Easy to communicate data from a parent to some super nested child

Note:(We use Context in Blog App)

Edit with WPS Office

1(Adding Context)

2: Add a context folder inside src folder
3: Add all data realted component screen inside context folder

--> Now Add BlogContext.js inside context

Before
BlogContext.js

```
import React, { Children } from 'react';

// Create BlogContext object

const BlogContext = React.createContext();


// Export the BlogProvider Context for Children Components ex: App from App.js

const BlogProvider = ({children}) =>{

return <BlogContext.Provider>{Children}</BlogContext.Provider> // Here Children is CustomComponent form App.js

};
```

Note:(BLogContext.Provider) --> provider is source of info that make available to all screens

_----------

After Changes occur App.js Will be

```
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
import IndexScreen from './src/screens/IndexScreen';
import React from 'react';
import { BlogProvider } from './src/context/BlogContext';     // Import the BlogProvider Here


const navigator = createStackNavigator({

  Index: IndexScreen

}, {

        initialRouteName: 'Index',
        defaultNavigationOptions: {
           title: 'Blogs'
    }
});

const App = createAppContainer(navigator);    // Assign the Result to App Variable

export default () => {

   return <BlogProvider>    ----------> Add Parent Here
              <App />
```

Edit with WPS Office

```
    </BlogProvider>     /// Own Custom Compoenet that include all stuff
};                      // If you see the error Then might not import React upward
```

After Changes occured in BlogContext.js

```
import React, { Children } from 'react';

// Create BlogContext object

const BlogContext = React.createContext();


// Export the BlogProvider Context for Children Components ex: App from App.js

const BlogProvider = ({children}) =>{

return <BlogContext.Provider value={5} >{children}</BlogContext.Provider> // Here Children is CustomComponent form App.js

};

export default BlogContext;


//Note:(value={} is piece of info that allows which Screen should access Parents Data )
// Apply the value component to those Screen which want to access their Parents Data
```

IndexScreen.js

```
import React, { useState } from 'react';
import {View,Text,StyleSheet} from 'react-native';
import BlogContext from '../context/BlogContext';    // Import Here to access the prop of Parent

const IndexScreen = () => {

    // Hook fun is used to access the props or context of parent Component BlogProvider

    const value = useState(BlogContext); // Simply pass the BLogContext object here

    return (

        <View>
            <Text>Index Screen</Text>
              <Text>{value}</Text>      -----> value Appear here when we change from parent Component ContextProvider
        </View>

    )
};
```

Edit with WPS Office

```
const Styles = StyleSheet.create({



});

export default IndexScreen;
```

Note:(When you pass the object as a value from parent to child its throws a Error)

(Rendering a List of Posts) ---> Pass the posts

App.js

```
import { createStackNavigator } from 'react-navigation-stack';
import { createAppContainer } from    'react-navigation';
import IndexScreen from './src/screens/IndexScreen';
import BlogContext from './src/context/BlogContext';
import React, {useState} from 'react';
const navigator = createStackNavigator(
   {
      Index: IndexScreen,
   },
   {
      initialRouteName: 'Index',
      defaultNavigationOptions: {
         title: 'Blogs',
      },
   }
);

const App = createAppContainer(navigator);

export default () => {
   const [blogPosts, setBlogPosts] = useState([]);

   const addBlogPost = () => { // Creating a func that used setter method and add new
blogpost to our Main blogPosts variable
      setBlogPosts([     // Create a new array // Not changing the orginal blogPosts variable
         ...blogPosts, // ...blogPosts means take all the current blogPosts we have add them it
in new array [...blogPosts]
         { title: `Blog Post #${blogPosts.length + 1}` }, // blogPosts.length is current intial value
  ,and + 1 and a new value
      ]);
   };

   return (
      <BlogContext.Provider value={{ data: blogPosts, addBlogPost }}>
         <App />
      </BlogContext.Provider>
   );
};
```

Edit with WPS Office

IndexScreen.js

```
import React, { useContext } from 'react';
import { View, Text, StyleSheet, Button, FlatList } from 'react-native';
import BlogContext from '../context/BlogContext';

const IndexScreen = () => {
   const { data, addBlogPost } = useContext(BlogContext);    // {data , addBlogPost} are the
props that we used for getting data

   return (
      <View>
         <Text>Index Screen</Text>
         <Button title="Add Post" onPress={addBlogPost} />
         <FlatList
            data={data}
            keyExtractor={blog => blog.title}
            renderItem={({ item }) => {
               return <Text>{item.title}</Text>;
            }}
         />
      </View>
   );
};

const styles = StyleSheet.create({});

export default IndexScreen;
```

Q: What is useReducer hook?
Ans: Is all about Using Switch Statements with actions objects
Manage our data with useReducer hook

(Updating With usereducer)

useReducer is working with (state,action)

BlogContext.js

```
import React, { useReducer } from 'react';

const BlogContext = React.createContext();

const blogReducer = (state, action) => {    // Step 4
   switch (action.type) {
      case 'add_blogpost':
         return [...state, { title: `Blog Post #${state.length + 1}` }];
      default:
         return state;
   }
};
```

Edit with WPS Office

```
export const BlogProvider = ({ children }) => {
  const [blogPosts, dispatch] = useReducer(blogReducer, []); // dispact variable is used for addblogpost    // Step 1

  const addBlogPost = () => {
    dispatch({ type: 'add_blogpost' }); // here type is action object // Step: 3
  };

  return (                                                    // step 2:
    <BlogContext.Provider value={{ data: blogPosts, addBlogPost }}>
      {children}
    </BlogContext.Provider>
  );
};

export default BlogContext;



//Steps

//step 1: When our application is first render there is    empty array -> const [blogPosts, dispatch] = useReducer(blogReducer, []);
//step 2: Set up our provider to render the entire application    <BlogContext.Provider value={{ data: blogPosts, addBlogPost }}>    ----> intial data & Callback function
 //        {children}
  //       </BlogContext.Provider>

//step 3: addBlogPost call back function is now dispatch with -->        dispatch({ type: 'add_blogpost' }); // here type is action object

// step 4: react now call
// const blogReducer = (state, action) => {    --- state is first argument and action is what we have to change
 // switch (action.type) {    ------------------> There is action.type
  //    case 'add_blogpost':       -------> this is the case is to dispatch inside provider function
    //      return [...state, { title: `Blog Post #${state.length + 1}` }];
    // default:
      //    return state;
  //}
//};



IndexScreen.js



import React, { useContext } from 'react';
import { View, Text, StyleSheet, FlatList, Button } from 'react-native';
import BlogContext from '../context/BlogContext';

const IndexScreen = () => {
  const { data, addBlogPost } = useContext(BlogContext);

  return (
    <View>
      <Text>Index Screen</Text>
      <Button title="Add Post" onPress={addBlogPost} />
```

Edit with WPS Office

```
        <FlatList
          data={data}
          keyExtractor={blogPost => blogPost.title}
          renderItem={({ item }) => {
            return <Text>{item.title}</Text>;
          }}
        />
      </View>
    );
};

const styles = StyleSheet.create({});

export default IndexScreen;
```

Note:(When we add different resources or updating the resources Three thing should be change)

1:reducer function
2: Action ex: dispatcher ---> This is object which have all callback functions
3: initialState ex: [] empty array used in Context.js

For creating a another resources

1: Add createDataContext.js in context folder

Note: [Alltough it is reusable function]

```
import React, {useReducer} from 'react';

export default (reducer,actions,initialState) => {    // Pass the common 3 argument that we required to change or create our new resources

    const Context = React.createContext();     // Create a object

    const Provider = ({children}) => {

        const [state,dispatch] = useReducer(reducer,initialState);

        // action === {addBlogPost: (dispatch) => {return() => {} }}

          const boundActions = {};

          for (let key in actions){

            // key === 'addBlogPost'
            boundActions[key] = actions[key](dispatch); // call that function with dispatch
          };

        return <Context.Provider value={{ state, ...boundActions}}>
                    {children}
        </Context.Provider>



    };
```

Edit with WPS Office

```
    return {Context,Provider};

};
```

2: Delete some Code that we write in Context.js
Because we write in createDataContext

Firstly import the createDataContext into Context.js

Delete the code

After deleting the code we get

BlogContext.js

```
import createDataContext from './createDataContext';

const blogReducer = (state, action) => {
  switch (action.type) {
    case 'add_blogpost':
      return [...state, { title: `Blog Post #${state.length + 1}` }];
    default:
      return state;
  }
};


const addBlogPost = () => {

    return () => {

      dispatch({ type: 'add_blogpost' }); // here type is action object //

    };
  };


export const {Context, Provider } = createDataContext (blogReducer, {addBlogPost}, []);




// All we have createDataContext that we assigned in it

// we just passed the Three agrument state,action,initialState

// export const {Context, Provider } = createDataContext (blogReducer, {addBlogPost}, []);
```
Here we passed the 3 arguments

Edit with WPS Office

// We passed {addBlogPost} as object in argument

_____

(Now Destructing the IndexScreen.js)

Before

```
import React, { useContext } from 'react';
import { View, Text, StyleSheet, FlatList, Button } from 'react-native';
import BlogContext from '../context/BlogContext';

const IndexScreen = () => {
   const { data, addBlogPost } = useContext(BlogContext);

   return (
      <View>
         <Text>Index Screen</Text>
         <Button title="Add Post" onPress={addBlogPost} />
         <FlatList
            data={data}
            keyExtractor={blogPost => blogPost.title}
            renderItem={({ item }) => {
               return <Text>{item.title}</Text>;
            }}
         />
      </View>
   );
};

const styles = StyleSheet.create({});

export default IndexScreen;
```

After

```
import { View, Text, StyleSheet, FlatList, Button } from 'react-native';
import {Context} from '../context/BlogContext';      ----> change is here


const IndexScreen = () => {
   const { state, addBlogPost } = useContext(Context);      ------> change is here state instead of
data

   return (
      <View>
         <Text>Index Screen</Text>
         <Button title="Add Post" onPress={addBlogPost} />
         <FlatList
            data={state}    ---------------------------------------> change is here state instead of data
            keyExtractor={blogPost => blogPost.title}
            renderItem={({ item }) => {
```

Edit with WPS Office

```
                return <Text>{item.title}</Text>;
            }}
          />
        </View>
    );
};

const styles = StyleSheet.create({});

export default IndexScreen;
```

(Destructing App.js)

Before

```
import React from 'react';
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
import IndexScreen from './src/screens/IndexScreen';
import { BlogProvider } from './src/context/BlogContext';

const navigator = createStackNavigator(
    {
        Index: IndexScreen,
    },
    {
        initialRouteName: 'Index',
        defaultNavigationOptions: {
            title: 'Blogs',
        },
    }
);

const App = createAppContainer(navigator);

export default () => {
    return (
        <BlogProvider>
            <App />
        </BlogProvider>
    );
};
```

After

```
import React from 'react';
import { createAppContainer } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
import IndexScreen from './src/screens/IndexScreen';
import { Provider } from './src/context/BlogContext';      ---> Change is here

const navigator = createStackNavigator(
    {
```

Edit with WPS Office

```
      Index: IndexScreen,
   },
   {
      initialRouteName: 'Index',
      defaultNavigationOptions: {
         title: 'Blogs',
      },
   }
);

const App = createAppContainer(navigator);

export default () => {
   return (
      <Provider>    ----------------------------------------> Change is here
         <App />
      </Provider>
   );
};
```

(Now Style the IndexScreen.js for delete the post)

```
import React, { useContext } from 'react';
import { View, Text, StyleSheet, FlatList, Button } from 'react-native';
import { Context } from '../context/BlogContext';
import { Feather } from '@expo/vector-icons';      ----------------------------> Add icon Import here

const IndexScreen = () => {
   const { state, addBlogPost } = useContext(Context);

   return (
      <View>
         <Button title="Add Post" onPress={addBlogPost} />
         <FlatList
            data={state}
            keyExtractor={blogPost => blogPost.title}
            renderItem={({ item }) => {
               return (
                  <View style={styles.row}> ----------------------------------> Add view then style your
titile and icon
                     <Text style={styles.title}>{item.title}</Text>
                     <Feather style={styles.icon} name="trash" />
                  </View>
               );
            }}
         />
      </View>
   );
};

const styles = StyleSheet.create({
   row: {----------------------------------------------------------> Add Style here
      flexDirection: 'row',
      justifyContent: 'space-between',
      paddingVertical: 20,
      paddingHorizontal: 10,
      borderTopWidth: 1,
```

Edit with WPS Office

```
        borderColor: 'gray'
    },
    title: {
        fontSize: 18
    },
    icon: {
        fontSize: 24
    }
});

export default IndexScreen;
```

(Now Add Random Id to each title and icon)

for random genrating id use Math function

Math flor (Math.random() * 99999)

Add into BlogContext.js

```
import createDataContext from './createDataContext';

const blogReducer = (state, action) => {
    switch (action.type) {
        case 'add_blogpost':
            return [...state,
                {
                  id: Math.floor(Math.random()* 99999),
                  title: `Blog Post #${state.length + 1}`
                }
            ];
        default:
            return state;
    }
};

const addBlogPost = dispatch => {
    return () => {
        dispatch({ type: 'add_blogpost'});
    };
};

export const { Context, Provider } = createDataContext(
    blogReducer,
    { addBlogPost },
    []
);
```

(Now return the id) & (Add Touchable Opacity)

So Add {item.id} into IndexScreen.js

Edit with WPS Office

So Add Touchable opacity to icon into IndexScreen.js So that we can touch our delete icon and it response well

```
import React, { useContext } from 'react';
import { View, Text, StyleSheet, FlatList, Button, TouchableOpacity } from 'react-native';
import { Context } from '../context/BlogContext';
import { Feather } from '@expo/vector-icons';

const IndexScreen = () => {
   const { state, addBlogPost } = useContext(Context);

   return (
      <View>
         <Button title="Add Post" onPress={addBlogPost} />
         <FlatList
            data={state}
            keyExtractor={blogPost => blogPost.title}
            renderItem={({ item }) => {
               return (
                  <View style={styles.row}>
                     <Text style={styles.title}> {item.title} - {item.id} </Text>
                     <TouchableOpacity onPress = {() => console.log(item.id)}>
                           <Feather style={styles.icon} name="trash" />
                      </TouchableOpacity>
                  </View>
               );
            }}
         />
      </View>
   );
};

const styles = StyleSheet.create({
   row: {
      flexDirection: 'row',
      justifyContent: 'space-between',
      paddingVertical: 20,
      paddingHorizontal: 10,
      borderTopWidth: 1,
      borderColor: 'gray'
   },
   title: {
      fontSize: 18
   },
   icon: {
      fontSize: 24
   }
});

export default IndexScreen;
```

(Now We add more functionality to BlogContext.js)

note: every time we add new functionality to context we have to two thing to remember
1: Add the new dispatcher function object
2: And apply to the case into reducer function

Edit with WPS Office

In this Case we add delete_blogpost dispacther function and give them id to delete a particular context

BlogContext.js

```
import createDataContext from './createDataContext';

const blogReducer = (state, action) => {
   switch (action.type) {----------------------------------------> add the new case and return the state with filter and apply the condition and check it
      case 'delete_blogpost':
         return state.filter((blogPost) => blogPost.id !== action.payload ); //here payload is id that we given inside deleteBlogPost
      case 'add_blogpost':
         return [...state,
            {
              id: Math.floor(Math.random()* 99999),
              title: `Blog Post #${state.length + 1}`
            }
         ];
      default:
         return state;
   }
};

const addBlogPost = dispatch => {
   return () => {
      dispatch({ type: 'add_blogpost'});
   };
};

const deleteBlogPost = dispatch => {    ---------------------------------> Add dispatcher function here
   return (id) => {
      dispatch({ type: 'delete_blogpost', payload: id }); ----------------> give them id as payload
   };
};

export const { Context, Provider } = createDataContext(
   blogReducer,
   { addBlogPost,deleteBlogPost },    ----------------------------------------------> Add here also
   []
);
```

(Now Add that dispatcher deleteBlogPost and id into IndexScreen.js for run them)

IndexScreen.js

```
import React, { useContext } from 'react';
import { View, Text, StyleSheet, FlatList, Button, TouchableOpacity } from 'react-native';
import { Context } from '../context/BlogContext';
```

Edit with WPS Office

```
import { Feather } from '@expo/vector-icons';

const IndexScreen = () => {
   const { state, addBlogPost,deleteBlogPost } = useContext(Context);    ----------------------------->
Add here dispatcher fun

   return (
      <View>
         <Button title="Add Post" onPress={addBlogPost} />
         <FlatList
            data={state}
            keyExtractor={blogPost => blogPost.title}
            renderItem={({ item }) => {
               return (
                  <View style={styles.row}>
                     <Text style={styles.title}> {item.title} - {item.id} </Text>
                     <TouchableOpacity onPress = {() => deleteBlogPost(item.id)}>    ----------> Add
here with id
                           <Feather style={styles.icon} name="trash" />
                      </TouchableOpacity>
                  </View>
               );
            }}
         />
      </View>
   );
};

const styles = StyleSheet.create({
   row: {
      flexDirection: 'row',
      justifyContent: 'space-between',
      paddingVertical: 20,
      paddingHorizontal: 10,
      borderTopWidth: 1,
      borderColor: 'gray'
   },
   title: {
      fontSize: 18
   },
   icon: {
      fontSize: 24
   }
});

export default IndexScreen;
```

:> Now run the app

Now (Add ShowScreen to naviagte from IndexScreen)

ShowScreen.js

Edit with WPS Office

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

const ShowScreen = () => {

      return (

            <View>
                  <Text>ShowScreen</Text>
            </View>

      )

};

const Styles = StyleSheet.create({});

export default ShowScreen;
```

Now(Add navigation prop and add Touchbleopactity to outside for onpress and add the component reference to navigate the screen)
(And Add id as object inside naviagtion.navigate('show', {id: item.id })))

IndexScreen.js

```
const IndexScreen = ({navigation}) => { ----------------------------------------> Apply navigation as prop
   const { state, addBlogPost,deleteBlogPost } = useContext(Context);

   return (
      <View>
         <Button title="Add Post" onPress={addBlogPost} />
         <FlatList
            data={state}
            keyExtractor={blogPost => blogPost.title}
            renderItem={({ item }) => {
               return (
                        <TouchableOpacity  onPress={()  =>  navigation.navigate('Show',
{id:item.id})}>   ------> apply here and add the component here with passing the id so that we
get acctual post here that we want to show
                              <View style={styles.row}>
                                 <Text style={styles.title}> {item.title} - {item.id} </Text>
                                    <TouchableOpacity onPress = {() => deleteBlogPost(item.id)}>
                                       <Feather style={styles.icon} name="trash" />
                                    </TouchableOpacity>
                              </View>
                        </TouchableOpacity>
               );
            }}
         />
      </View>
   );
};
```

Edit with WPS Office

Now (Update the ShowScreen.kjs)

```
import React, { useContext } from 'react';
import {View,Text,StyleSheet} from 'react-native';
import { Context } from '../context/BlogContext';    ----------------------------------> Here We import the Context
const ShowScreen = ({navigation}) => {

      const {state} = useContext(Context);    // pass the state as data here actual data is export from Contex means BlogContext so we pass Context

       const blogPost = state.find((blogPost) => blogPost.id === navigation.getParam('id'));
       // Here we find the state and compared our id to navigation id and assign to variable blogPost

      return (

           <View>
                <Text>{blogPost.title}</Text> ---------------------------> Here we rendered the title of post
           </View>

      )

};

const Styles = StyleSheet.create({});

export default ShowScreen;
```

Now (Create the CreateScreen & Add + Button to IndexScreen.js Header at right for going to Index to Create screen)

IndexScreen.js

```
import React, { useContext } from 'react';
import { View, Text, StyleSheet, FlatList, Button, TouchableOpacity } from 'react-native';
import { Context } from '../context/BlogContext';
import { Feather } from '@expo/vector-icons';


const IndexScreen = ({navigation}) => {
   const { state, addBlogPost,deleteBlogPost } = useContext(Context);

   return (
      <View>
         <Button title="Add Post" onPress={addBlogPost} />
         <FlatList
            data={state}
            keyExtractor={blogPost => blogPost.title}
```

Edit with WPS Office

```
            renderItem={({ item }) => {
               return (
                        <TouchableOpacity      onPress={()      =>      navigation.navigate('Show',
{id:item.id})}>
                              <View style={styles.row}>
                                 <Text style={styles.title}> {item.title} - {item.id} </Text>
                                    <TouchableOpacity onPress = {() => deleteBlogPost(item.id)}>
                                       <Feather style={styles.icon} name="trash" />
                                    </TouchableOpacity>
                              </View>
                        </TouchableOpacity>
               );
            }}
         />
      </View>
   );
};

IndexScreen.navigationOptions = ({navigation}) => { ----------------------------------------------------------->
for CreateScreen Navigation we use this function
 return {
   headerRight: () => ( ----------------------------------------------------------------------> add + to headeRight
side
      <TouchableOpacity onPress={() => navigation.navigate('Create')}>
         <Feather name="plus" size={30} />
      </TouchableOpacity>
   ),
}
};

const styles = StyleSheet.create({
   row: {
      flexDirection: 'row',
      justifyContent: 'space-between',
      paddingVertical: 20,
      paddingHorizontal: 10,
      borderTopWidth: 1,
      borderColor: 'gray'
   },
   title: {
      fontSize: 18
   },
   icon: {
      fontSize: 24
   }
});



Now:(CreateScreen.js with TextInput & Having Button to call And Add some Style)
      (Add State also )



import React, { useContext, useState } from 'react';
import {View,Text,StyleSheet,TextInput,Button} from 'react-native';
import { Context } from '../context/BlogContext';
const CreateScreen = ({navigation}) => {

      const [title, setTitle] = useState('');
```

Edit with WPS Office

```
    const [content, setContent] = useState('');

    return (

        <View>
            <Text style={Styles.label}>Enter Title:</Text>
            <TextInput style={Styles.input} value={title} onChangeText={(text) =>
setTitle(text) } />

            <Text style={Styles.label}>Enter Content:</Text>
            <TextInput style={Styles.input}value={content} onChangeText={(text) =>
setTitle(text) }/>

            <Button title="Add Blog Post"/>---------------------------------------------------------------->
Add Button
        </View>

    )

};

const Styles = StyleSheet.create({
    input: {

        fontSize: 10,
        borderWidth: 1,
        borderColor: 'black',
        marginBottom: 15,
        margin: 5,
        padding: 5

    },

    label: {

        fontSize: 20,
        marginBottom: 5,
        marginLeft: 5
    }
});

export default CreateScreen;
```

Now(Now Click the AddBlogPost and Apply the actions to dispatch function of blogContex.js)
(Aplly the titile and content as payload and pass as argument to dispatch function and apply them to the Swith and case)

BlogContext.js

```
import createDataContext from './createDataContext';

const blogReducer = (state, action) => {
  switch (action.type) {
    case 'delete_blogpost':
```

Edit with WPS Office

```
        return state.filter((blogPost) => blogPost.id !== action.payload ); //here payload is id
that we given inside deleteBlogPost
      case 'add_blogpost':
        return [...state,
            {
              id: Math.floor(Math.random()* 99999),
              title: action.payload.title,----------------------------------> Add title here
              content: action.payload.content----------------------------------> Add content here
            }
        ];
      default:
        return state;
  }
};

const addBlogPost = dispatch => {
  return (title,content) => {-------------------------------------------------------------> Add the title and
content here
    dispatch({ type: 'add_blogpost', payload: {title, content}});     ----------------------------> Add
payload here
  };
};

const deleteBlogPost = dispatch => {
  return (id) => {
    dispatch({ type: 'delete_blogpost', payload: id });
  };
};

export const { Context, Provider } = createDataContext(
  blogReducer,
  { addBlogPost, deleteBlogPost },
  []
);
```

Now(add Context to Createscreen.js to onpress the button to navigate)
(We also pass the call back function to button and also pass to as 3 argument in dispatch fun in BlogContext.js)
(We delete the AddPoSt button in the Indexscreen and Delete the AddBlogPost as state in IndexScreen.js)

CreateScreen.js

```
import React, { useContext, useState } from 'react';
import {View,Text,StyleSheet,TextInput,Button} from 'react-native';
import { Context } from '../context/BlogContext';
const CreateScreen = ({navigation}) => {

    const [title, setTitle] = useState('');
    const [content, setContent] = useState('');
    const { addBlogPost } = useContext(Context);

    return (
```

Edit with WPS Office

```
            <View>
                <Text style={Styles.label}>Enter Title:</Text>
                <TextInput      style={Styles.input}      value={title}      onChangeText={(text)      =>
setTitle(text) } />

                <Text style={Styles.label}>Enter Content:</Text>
                <TextInput         style={Styles.input}value={content}    onChangeText={(text)    =>
setContent(text) }/>

                <Button title="Add Blog Post" onPress={() => { addBlogPost(title,content, () => { -
------------------------>Here apply the call back fun

                    navigation.navigate('Index');----------------------> navigate to Index

                });

                }}/>
            </View>

        )

};

const Styles = StyleSheet.create({
    input: {

        fontSize: 10,
        borderWidth: 1,
        borderColor: 'black',
        marginBottom: 15,
        margin: 5,
        padding: 5

    },

    label: {

        fontSize: 20,
        marginBottom: 5,
        marginLeft: 5
    }
});

export default CreateScreen;
```

BlogContext.js

```
const addBlogPost = dispatch => {
  return (title,content,callback) => {    ----------------------------------------> add Callback here
    dispatch({ type: 'add_blogpost', payload: {title, content}});
    callback();    ----------------------------> apply here
  };
};
```

Edit with WPS Office

IndexScreen.js

```
const IndexScreen = ({navigation}) => {
   const { state ,deleteBlogPost } = useContext(Context);      ----------> deleted the addBlogPost which is no longer in use
```

Now(go to src make a folder called components inside components make a folder called BlogPostForm)
(Cut and paste some code of EditScreen and show screen)

Before

EditScreen.js

```
import React, { useContext, useState } from 'react';
import {View,Text,StyleSheet,TextInput,Button} from 'react-native';
import { Context } from '../context/BlogContext';
const EditScreen = ({navigation}) => {

      const { state } = useContext(Context);

      const blogPost = state.find(
            blogPost=> blogPost.id === navigation.getParam('id')
            );
      // Here we find the state and compared our id to navigation id and assign to variable blogPost

        const [title,setTitle] = useState(blogPost.title);
        const [content,setContent] = useState(blogPost.content);

      return (

            <View>
                  <Text style={Styles.label}>Edit Title:</Text>
                  <TextInput    style={Styles.input}    value={title}    onChangeText={(newTitle)    => setTitle(newTitle) } />

                  <Text style={Styles.label}>Edit Content:</Text>
                  <TextInput     style={Styles.input}value={content}  onChangeText={(newContent) => setContent(newContent) }/>

                  <Button title="Add Blog Post" onPress={() => { addBlogPost(title,content, () => {

                        navigation.navigate('Index');

                  });

                  }}/>
            </View>

      )
```

Edit with WPS Office

```
};

const Styles = StyleSheet.create({
    input: {

        fontSize: 10,
        borderWidth: 1,
        borderColor: 'black',
        marginBottom: 15,
        margin: 5,
        padding: 5

    },

    label: {

        fontSize: 20,
        marginBottom: 5,
        marginLeft: 5
    }
});

export default EditScreen;
```

After------>

```
import React, { useContext } from 'react';
import {StyleSheet} from 'react-native';
import { Context } from '../context/BlogContext';
import BlogPostForm from '../components/BlogPostForm';
const EditScreen = ({navigation}) => {
    const id = navigation.getParam('id');

    const { state,editBlogPost } = useContext(Context);

    const blogPost = state.find(
            blogPost=> blogPost.id === id
        );
    // Here we find the state and compared our id to navigation id and assign to variable
blogPost

   return <BlogPostForm
            initialValues={{title: blogPost.title, content: blogPost.content}}
            onSubmit={(title,content) => {

              editBlogPost(id,title,content, () => navigation.pop()); // callback func
navigation.pop()

            }} />

};

const Styles = StyleSheet.create({

});
```

Edit with WPS Office

```
export default EditScreen;
```

Before CreateScreen.js

```
import React, { useContext, useState } from 'react';
import {View,Text,StyleSheet,TextInput,Button} from 'react-native';
import { Context } from '../context/BlogContext';
const CreateScreen = ({navigation}) => {

    const [title, setTitle] = useState('');
    const [content, setContent] = useState('');
    const { addBlogPost } = useContext(Context);

    return (

        <View>
            <Text style={Styles.label}>Enter Title:</Text>
            <TextInput style={Styles.input} value={title} onChangeText={(text) => setTitle(text) } />

            <Text style={Styles.label}>Enter Content:</Text>
            <TextInput style={Styles.input}value={content} onChangeText={(text) => setContent(text) }/>

            <Button title="Add Blog Post" onPress={() => { addBlogPost(title,content, () => {

                navigation.navigate('Index');

            });

            }}/>
        </View>

    )

};

const Styles = StyleSheet.create({
    input: {

        fontSize: 10,
        borderWidth: 1,
        borderColor: 'black',
        marginBottom: 15,
        margin: 5,
        padding: 5

    },

    label: {

        fontSize: 20,
        marginBottom: 5,
```

Edit with WPS Office

```
        marginLeft: 5
    }
});

export default CreateScreen;
```

After----->

```
import React, { useContext} from 'react';
import {StyleSheet} from 'react-native';
import { Context } from '../context/BlogContext';
import BlogPostForm from '../components/BlogPostForm';
const CreateScreen = ({navigation}) => {

    const { addBlogPost } = useContext(Context);

    return <BlogPostForm    onSubmit={(title,content) => {

        addBlogPost (title,content, () => navigation.navigate('Index'));

    }}/>

};

const Styles = StyleSheet.create({

});

export default CreateScreen;
```

BlogPostForm.js

```
import React, { useContext, useState } from 'react';
import {View,Text,StyleSheet,TextInput,Button} from 'react-native';
import { Context } from '../context/BlogContext';
const BlogPostForm = ({onSubmit, initialValues}) => {

    const [title, setTitle] = useState(initialValues.title);
    const [content, setContent] = useState(initialValues.content);

        return (

        <View>
            <Text style={Styles.label}>Enter Title:</Text>
            <TextInput    style={Styles.input}    value={title}    onChangeText={(text)    =>
setTitle(text) } />

            <Text style={Styles.label}>Enter Content:</Text>
            <TextInput      style={Styles.input}value={content}    onChangeText={(text)    =>
setContent(text) }/>
```

Edit with WPS Office

```
                <Button
                  title="Save Blog Post"
                  onPress={() => onSubmit(title,content) } />
            </View>

      )

};

BlogPostForm.defaultProps = {

      initialValues:{
            title: '',
            content: ''
      }
};

const Styles = StyleSheet.create({
      input: {

            fontSize: 10,
            borderWidth: 1,
            borderColor: 'black',
            marginBottom: 15,
            margin: 5,
            padding: 5

      },

      label: {

            fontSize: 20,
            marginBottom: 5,
            marginLeft: 5
      }
});

export default BlogPostForm;
```

ShowScreen.js

```
import React, { useContext } from 'react';
import {View,Text,StyleSheet} from 'react-native';
import { Context } from '../context/BlogContext';
import {EvilIcons} from '@expo/vector-icons';
const ShowScreen = ({navigation}) => {

      const {state} = useContext(Context);    // pass the state as data here actual data is export
from Contex means BlogContext so we pass Context

        const blogPost = state.find((blogPost) => blogPost.id === navigation.getParam('id'));
        // Here we find the state and compared our id to navigation id and assign to variable
blogPost

      return (
```

Edit with WPS Office

```
            <View>
                <Text>{blogPost.title}</Text>
                <Text>{blogPost.content}</Text>
            </View>

        )

};

ShowScreen.navigationOptions = ({navigation}) => {

        return {
            headerRight: () => (
              <TouchableOpacity          onPress={()          =>          navigation.navigate('Edit',{id:
navigation.getParam('id')})}>
                <EvilIcons name="pencil" size={35} />
              </TouchableOpacity>
            ),
          };



};


const Styles = StyleSheet.create({});

export default ShowScreen;
```

IndexScreen.js

```
import React, { useContext } from 'react';
import {View,Text,StyleSheet} from 'react-native';
import { Context } from '../context/BlogContext';
import {EvilIcons} from '@expo/vector-icons';
const ShowScreen = ({navigation}) => {

        const {state} = useContext(Context);    // pass the state as data here actual data is export
from Contex means BlogContext so we pass Context

          const blogPost = state.find((blogPost) => blogPost.id === navigation.getParam('id'));
         // Here we find the state and compared our id to navigation id and assign to variable
blogPost

        return (

            <View>
                <Text>{blogPost.title}</Text>
                <Text>{blogPost.content}</Text>
            </View>

        )

};

ShowScreen.navigationOptions = ({navigation}) => {
```

Edit with WPS Office

```
      return {
            headerRight: () => (
              <TouchableOpacity          onPress={()          =>          navigation.navigate('Edit',{id:
navigation.getParam('id')})}>
                  <EvilIcons name="pencil" size={35} />
              </TouchableOpacity>
            ),
          };



};


const Styles = StyleSheet.create({});

export default ShowScreen;
```

BlogContext.js

```
import createDataContext from './createDataContext';

const blogReducer = (state, action) => {
   switch (action.type) {
      case 'edit_blogpost':
         return state.map(() => {

            return blogPost.id === action.payload.id ? action.payload : blogPost;

         });
      case 'delete_blogpost':
         return state.filter((blogPost) => blogPost.id !== action.payload ); //here payload is id
that we given inside deleteBlogPost
      case 'add_blogpost':
         return [...state,
              {
                id: Math.floor(Math.random()* 99999),
                title: action.payload.title,
                content: action.payload.content
              }
         ];
      default:
         return state;
   }
};

const addBlogPost = dispatch => {
   return (title,content,callback) => {
      dispatch({ type: 'add_blogpost', payload: {title, content}});
      if (callback){
        callback();

      }
```

Edit with WPS Office

```
    };
};

const deleteBlogPost = dispatch => {
    return (id) => {
        dispatch({ type: 'delete_blogpost', payload: id });
    };
};

const editBlogPost = dispatch =>{
  return (id,title,content,callback) => {
      dispatch({type: 'edit_blogpost',
      payload: {id,title,content}});
      if (callback){
        callback();

      }
  };
};

export const { Context, Provider } = createDataContext(
    blogReducer,
    { addBlogPost, deleteBlogPost, editBlogPost },
    [] // default screen title
);


// All we have createDataContext that we assigned in it

// we just passed the Three agrument state,action,initialState

// export const {Context, Provider } = createDataContext (blogReducer, {addBlogPost}, []);
Here we passed the 3 arguments


// We passed {addBlogPost} as object in argument
```

---

createDataContext.js

```
import React, { useReducer } from 'react';

export default (reducer, actions, initialState) => {
    const Context = React.createContext();

    const Provider = ({ children }) => {
        const [state, dispatch] = useReducer(reducer, initialState);

        // actions === { addBlogPost: (dispatch) => { return () => {} } }
        const boundActions = {};
        for (let key in actions) {
            boundActions[key] = actions[key](dispatch);
        }

        return (
            <Context.Provider value={{ state, ...boundActions }}>
```

Edit with WPS Office

```
            {children}
        </Context.Provider>
    );
  };

  return { Context, Provider };
};
```

Now(OutSide Data Api)----------> new topic related to existing one

JASON-SERVER

npmjs.com/package/json-server

1: Make new folder outside your project called jsonserver

2: open into cmd

3: npm init

4: press Enter till we get our cmd line

5: There is json.pakage is inside the folderr

6: npm install json-server ngrok

7: open the vscode for this

8: create db.json file

db.json

```
{
    "blogposts": []
}
```

9: go to package.json and Add 2 script one for jsonserver and another for ngrok tool

```
 "name": "jasonserver",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "db": "json-server -w db.json",      ----> Here are for run the json-sever    Note: (if the server is not working try this: json-server -w db.json -p 3001)
    "tunnel": "ngrok http 3002"      ------> here are the json port that will help to expose ngork to local machine (if server is not working then: ngrok http 30002)
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "json-server": "^0.16.1",
    "ngrok": "^3.2.7"
```

Edit with WPS Office

```
  }
}
```

9: open another cmd jasonserver directory
run also the exisiting one

10: npm run db ----> on second server

11: npm run tunnel ---------->on existing fist cmd

Forwarding http://4616ee3e331d.ngrok.io -> http://localhost:3000

copy the http and paste into new browser and your json server is running

http://4616ee3e331d.ngrok.io

12: to restart the server

npm run tunnel

13: Note: (When u run the project there are there cmd neccessary)

open 3 in 3 diffrent cmd

1: react-native bundler

2: npm run db -----> db server

3: npm run tunnel ---> ngrok

14: However we intreact from react-native app over to json-server,
We going to making plain network request
for making plain network request we going to use axios library

15: open another terminal cmd window inside blog app

and write

npm install axios

16: Diagram to Show to interact with json server

	Method	Route	Result
1	GET	/blogposts	Retrieve all stored blog posts
2	GET	/blogposts/{id}	Get blog posts with particular id
3	POST	/blogposts	Create a new blogposts
4	PUT	/blogposts/{id}	Upadte blog posts with given id

Edit with WPS Office

5 DELETE /blogposts/{id} Delete blog posts with given id

15:http://4616ee3e331d.ngrok.io/blogposts -----> Actual request is

Domain is change everytime so you should update your request according to ngork domain

16: Making a Network Request

currentflow:

1:IndexScreen get displayed by react nivigation
2:IndexScreen looks a context object to get the current list of blog posts
3:IndexScreen render the list of blogposts
4:Done

Newflow:

1:IndexScreen get displayed by react nivigation
2:IndexScreen needs to call a function that will make a request to the json server for the current list of objects ---> that will be our action function
3:IndexScreen recive list of blogposts currently empty
4: Time passes...
5:Request complete, list of blogposts get stored through our 'useReducer hook'
6:State changed, rendered whole app ,IndexScreen get new posts list

17: go to project src folder and make new folder api

18: inside api make a file named as jsonServer.js

jsonServer.js

```
import axios from 'axios';

export default axios.create({

baseURL: ' http://4616ee3e331d.ngrok.io '    //remember this url is tempory for 7 hours and then u want to update it

});
```

19: Go to BlogContext.js

now import axios instance

```
import jsonServer from '../api/jsonServer';
```

20: need to call a function that will make a request

-> In our case it is our action function

-> Make this function

Edit with WPS Office

```
const getBlogPost = dispatch => {
   return async () => {
      const response = await jsonServer.get('/blogposts')
      //response.data === [{},{},{}]    list of objects
      // now call the dispatch

      dispatch({type: 'get_blogposts', payload: response.data});
   };

};
```

->Add the action type to switch case

```
  case 'get_blogposts':
         return action.payload;
```

->Add them into object so it will available for all

```
export const { Context, Provider } = createDataContext(
   blogReducer,
   { addBlogPost, deleteBlogPost, editBlogPost },----------------------------------> Here we add
   [] // default screen title
);
```

21: Remote Fetch of Posts

Q: What is useEffect in react-native?
Ans: It is hook fun is to make sure we run some bit of code run one time when component is first rendered

so import the useEffect into your indexScreen

IndexScreen.js

```
import React, { useContext ,useEffect} from 'react';
import { View, Text, StyleSheet, FlatList, Button, TouchableOpacity } from 'react-native';
import { Context } from '../context/BlogContext';
import { Feather } from '@expo/vector-icons';


const IndexScreen = ({navigation}) => {
   const { state ,deleteBlogPost, getBlogPost } = useContext(Context);

   useEffect (() => {

      getBlogPost();

   }, []);
```

22: Manualy add the object into db.json into json server
open the vs code of jsonserver

```
   {
      "blogposts": [
```

Edit with WPS Office

```
    {
      "id": 1,
       "title": "API POST",
       "content": "content for my post"

    }
]
}
```

23: If you to see your db.json is working

http://4616ee3e331d.ngrok.io/blogposts

paste to crome

If it is not working then somthing wrong with installation

If its working ans still see the error then error in your react-native

24: Creating a posts with blogposts

```
const addBlogPost = dispatch => {

  return async (title,content,callback) => {----------------------------------------> return with async
    await  jsonServer.post('/blogposts',{title,content});----------------------------> we use .post
method for creating new fresh post
    // dispatch({ type: 'add_blogpost', payload: {title, content}});
    if (callback){
      callback();

    }
  };
}
```

IndexScreen.js

```
const IndexScreen = ({navigation}) => {
  const { state ,deleteBlogPost, getBlogPost } = useContext(Context);

  useEffect (() => {

    // first time we vist the screen do one fatch
    getBlogPost();

    // Anytime to return the screen do fatch again

      const listener = navigation.addListener('didFocus', () => {   ----------------------> Add a listner
for showing our update to screen

        getBlogPost();

    });

      return () =>{   ----------------------------> return the listner

      listener.remove();
```

Edit with WPS Office

```
        };

    }, []);
```

25: Delete the post

```
const deleteBlogPost = dispatch => {
   return    async id => {
       await jsonServer.delete(`/blogposts/${id}`);    // delete method use for delete and pass the id

       dispatch({ type: 'delete_blogpost', payload: id });
   };
};
```

26: Edit the Existing post

```
const editBlogPost = dispatch =>{
  return    async (id,title,content,callback) => {

   await  jsonServer.put(`/blogposts/${id}`,{title,content}); // put method use for edit the exsiting post

   dispatch({type: 'edit_blogpost',
     payload: {id,title,content}});

     if (callback){
       callback();

     }
  };
};
```

_____END Of App_____

```
import React from 'react';
import {View,Text,StyleSheet} from 'react-native';
import ImageDetails from '../components/ImageDetails';


const ImageScreen = () => {

       return(

         <View>

         <ImageDetails
           title ="Beach"
           imageSource = {require('../../assets/beach.png')}
           score={10} />

         <ImageDetails
           title ="Forest"
           imageSource = {require('../../assets/forest.png')}
           score={9} />
```

Edit with WPS Office

```
        <ImageDetails
          title ="Mountain"
          imageSource = {require('../../assets/mountain.png')}
          score={8} />

        </View>

      );
};

const Styles = StyleSheet.create({


});

export default ImageScreen;
```

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


const ImageDetails = props =>{

      return(

        <View>

        <Image source = {props.imageSource} />
      <Text style={Styles.ImageText}>{props.title}</Text>
      <Text>Image Score-{props.score}</Text>
        </View>

      );
};

const Styles = StyleSheet.create({

    ImageText:{

        fontSize: 40

    }

});

export default ImageDetails;
```

Edit with WPS Office