Basic layout

Create New Project with expo

`expo init rn-complete-guide`

choose blank template

1. **Basic layout - 1**

```
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { StyleSheet, Text, View,TextInput, Button } from 'react-native';

export default function App() {
  return (
    <View style={{padding: 50}}>
      <View style={{flexDirection:'row',justifyContent:'space-between',
          alignItems:'center'}}>
        <TextInput placeholder="Course Goal"
        style={{width:'80%',borderColor:'black',borderBottomWidth:1
          ,padding:10}}/>
        <Button title="Add"/>
      </View>
      <View>

      </View>
    </View>
  );
}

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

Flexbox

Ref:

https://medium.com/wix-engineering/the-full-react-native-layout-cheat-sheet-a4147802405c

https://reactnative.dev/docs/flexbox

```
// each box
<View style={{
        backgroundColor:'red',
        width:100,
        height:100,
        justifyContent:'center',
        alignItems:'center'
      }}>
        <Text>1</Text>
</View>
```

```
// For padding with each border
<View style={{padding:50}}>
</Vie>
```

```
// flex box direcetion
<View style={{padding:50, flexDirection:'row'}}>
</View>
```

```
// add parent view this style
// remove child view width and height
<View style={{
      padding:50,
      flexDirection:'row',
      width:'80%',
      height:300,
      justifyContent:'space-around',
      alignItems:'stretch'
      }}>
</View>
```

```
// add flex:1 at child view

 <View style={{
        backgroundColor:'red',
        flex:1,
        justifyContent:'center',
        alignItems:'center'
      }}>
        <Text>1</Text>
      </View>
```

```
// add flex:2 for second child view
<View style={{
        backgroundColor:'green',
        flex:2,
        justifyContent:'center',
        alignItems:'center'
      }}>
        <Text>2</Text>
      </View>
```

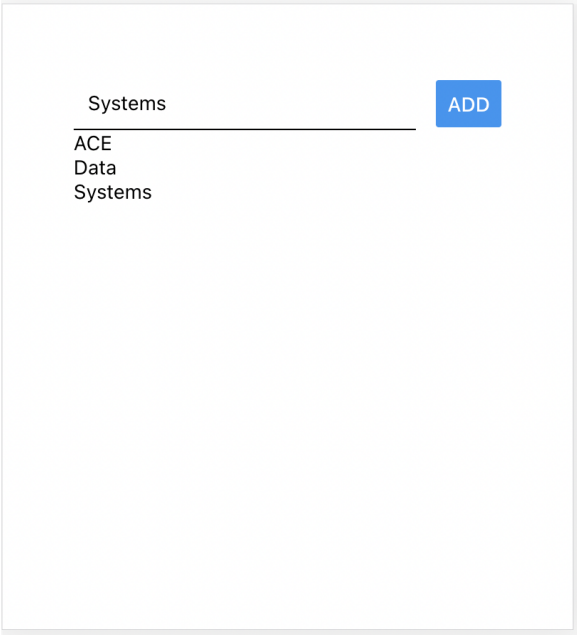
Basic layout - 2

```
// create a style with style sheet object

<View style={styles.screen}>
      <View style={styles.inputContainer}>
        <TextInput placeholder="Course Goal" style={styles.input} />
        <Button title="Add" />
      </View>
      <View></View>
    </View>

const styles = StyleSheet.create({
  screen: {
    padding: 50,
  },
  inputContainer: {
    flexDirection: "row",
    justifyContent: "space-between",
    alignItems: "center",
  },
  input: {
    width: "80%",
    borderColor: "black",
    borderBottomWidth: 1,
    padding: 10,
  },
});
```

Working with Stage & Events

```
//1. import useStage
import React,{useState} from "react";

//2. set for null value for text box
const [enteredGoal,setEnteredGoal] = useState('');

//3. create function goInputHandler
function goInputHandler(enterText) {
    setEnteredGoal(enterText);
  }

//3.1 arrow function
const goInputHandler = enterText => setEnteredGoal(enterText);

//4. add onChangeText events
<TextInput
          placeholder="Course Goal"
          style={styles.input}
          onChangeText={goInputHandler}
  />

//5. add value property
<TextInput
          placeholder="Course Goal"
          style={styles.input}
          onChangeText={goInputHandler}
          value={enteredGoal}
   />

//6. add onPress Events at button
<Button title="Add" onPress={addGoHandler} />

//7. implement addGoHandler function
const addGoHandler = () => {
    console.log(enteredGoal);
  }


//8
const [courseGoals, setCourseGoals] = useState([]);

// courseGoal is previous stage, enterGoal is update stage
const addGoHandler = () => {
   setCourseGoals(currentGoals => [...currentGoals,enteredGoal]);
  }
// 9 Bind the new view
<View>
       {courseGoals.map((goal) => <Text key={goal}>{goal}</Text>)}
</View>
```

Testing the app



2. **Styling the items**

```
// Text component to wrap with View component
// View component is more style option
<View>
        <View style={styles.listItem}>
          {courseGoals.map((goal) => (
            <Text key={goal}>{goal}</Text>
          ))}
        </View>
</View>

// list item sytle create

listItem:{
    padding:10,
    backgroundColor:'#ccc',
    borderColor:'black',
    borderWidth:1
  }


// add scroll view
<ScrollView>
      <View>
```

```
        {courseGoals.map((goal) => (
          <View key={goal} style={styles.listItem}>
            <Text >{goal}</Text>
          </View>
        ))}
      </View>
 </ScrollView>
```

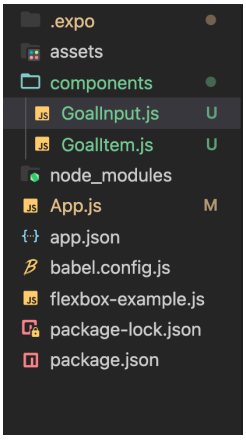
```
// Replace with scroll view

<FlatList
        data={courseGoals}
        renderItem={(itemData) => (
          <View style={styles.listItem}>
            <Text>{itemData.item}</Text>
          </View>
        )}
 ></FlatList>

// fix for key warning message
const addGoHandler = () => {
    setCourseGoals((currentGoals) => [
      ...currentGoals,
      { key: Math.random().toString(), value: enteredGoal},
    ]);
  };


<FlatList
        data={courseGoals}
        renderItem={(itemData) => (
          <View style={styles.listItem}>
            <Text>{itemData.item.value}</Text>
          </View>
        )}
      ></FlatList>

// add Key
<FlatList
        keyExtractor={(item,index) => item.id}
        data={courseGoals}
        renderItem={(itemData) => (
          <View style={styles.listItem}>
            <Text>{itemData.item.value}</Text>
          </View>
        )}
      ></FlatList>
```

3. **Styling the app into components**

3.1 Create component folder

3.1.1 Create **GoalInput.js**

3.1.2 Create **GoalItem.js**




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

// implement
const GoalItem = (props) => {
  return (
    <View style={styles.listItem}>
      <Text>{props.title}</Text>
    </View>
  );
};

// create style object
const styles = StyleSheet.create({
  listItem: {
    padding: 10,
    marginVertical: 5,
    backgroundColor: "#ccc",
    borderColor: "black",
    borderWidth: 1,
```

```
  },
});


// 2 import goal item to app.js
import GoalItem from "./components/GoalItem";

<FlatList
        keyExtractor={(item, index) => item.id}
        data={courseGoals}
        renderItem={(itemData) => <GoalItem title={itemData.item.value} />}
 ></FlatList>
```

4. **Passing data between components**

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

const GoalInput = (props) => {
  const [enteredGoal, setEnteredGoal] = useState("");

  const goInputHandler = (enteredText) => {
    setEnteredGoal(enteredText);
  };


  return (
    <View style={styles.inputContainer}>
      <TextInput
        placeholder="Course Goal"
        style={styles.input}
        onChangeText={goInputHandler}
        value={enteredGoal}
      />
      <Button title="Add" onPress={props.onAddGo.bind(this,enteredGoal)} />
    </View>
  );
};

const styles = StyleSheet.create({
  inputContainer: {
    flexDirection: "row",
    justifyContent: "space-between",
    alignItems: "center",
  },
  input: {
    width: "80%",
    borderColor: "black",
    borderBottomWidth: 1,
```

```
    padding: 10,
  },
});

export default GoalInput;
```

```
import { StatusBar } from "expo-status-bar";
import React, { useState } from "react";
import {
  StyleSheet,
  Text,
  View,
  TextInput,
  Button,
  ScrollView,
  FlatList,
} from "react-native";

import GoalItem from "./components/GoalItem";
import GoalInput from "./components/GoalInput";

export default function App() {

  const [courseGoals, setCourseGoals] = useState([]);


  const addGoHandler = goTitle => {
    setCourseGoals((currentGoals) => [
      ...currentGoals,
      { id: Math.random().toString(), value: goTitle },
    ]);
  };

  return (
    <View style={styles.screen}>
      <GoalInput onAddGo={addGoHandler} />
      <FlatList
        keyExtractor={(item, index) => item.id}
        data={courseGoals}
        renderItem={(itemData) => <GoalItem title={itemData.item.value} />}
      ></FlatList>
    </View>
  );
}

const styles = StyleSheet.create({
  screen: {
    padding: 50,
  },

});
```

5. **Working with touchable components**

```
onPress={() => console.Console("Does it work?")}

// add on press event at GoalItem
<FlatList
        keyExtractor={(item, index) => item.id}
        data={courseGoals}
        renderItem={(itemData) => (
          <GoalItem
            onPress={() => console.Console("Does it work?")}
            title={itemData.item.value}
          />
        )}
></FlatList>

// you don't see the output here

// Your parent view should be pressable
<View style={styles.listItem}>
      <Text>{props.title}</Text>
</View>

// wrap with touchable components
// Touchable , TouchableOpacit, TouchableHighlight, ...

<TouchableOpacity onPress={props.onDelete}>
      <View style={styles.listItem}>
        <Text>{props.title}</Text>
      </View>
 </TouchableOpacity>

<View style={styles.screen}>
      <GoalInput onAddGo={addGoHandler} />
      <FlatList
        keyExtractor={(item, index) => item.id}
        data={courseGoals}
        renderItem={(itemData) => (
          <GoalItem
            onDelete={() => console.Console("Does it work?")}
            title={itemData.item.value}
          />
        )}
      ></FlatList>
    </View>
```

6. **Delete Items**

```
const removeGoHanlder = goalID =>{
    setCourseGoals(currentGoals => {
      return currentGoals.filter((goal) => goal.id !== goalID)
    });
  }

// App.js
<View style={styles.screen}>
      <GoalInput onAddGo={addGoHandler} />
      <FlatList
        keyExtractor={(item, index) => item.id}
        data={courseGoals}
        renderItem={(itemData) => (
          <GoalItem
            id={itemData.item.id}
            onDelete={removeGoHanlder}
            title={itemData.item.value}
          />
        )}
      ></FlatList>
    </View>

// GoItem.js
const GoalItem = (props) => {
  return (
    <TouchableOpacity onPress={props.onDelete.bind(this,props.id)}>
      <View style={styles.listItem}>
        <Text>{props.title}</Text>
      </View>
    </TouchableOpacity>
  );
};
```

Git:Repo

https://github.com/arkarhtetmyint/rn-complete-guide.git