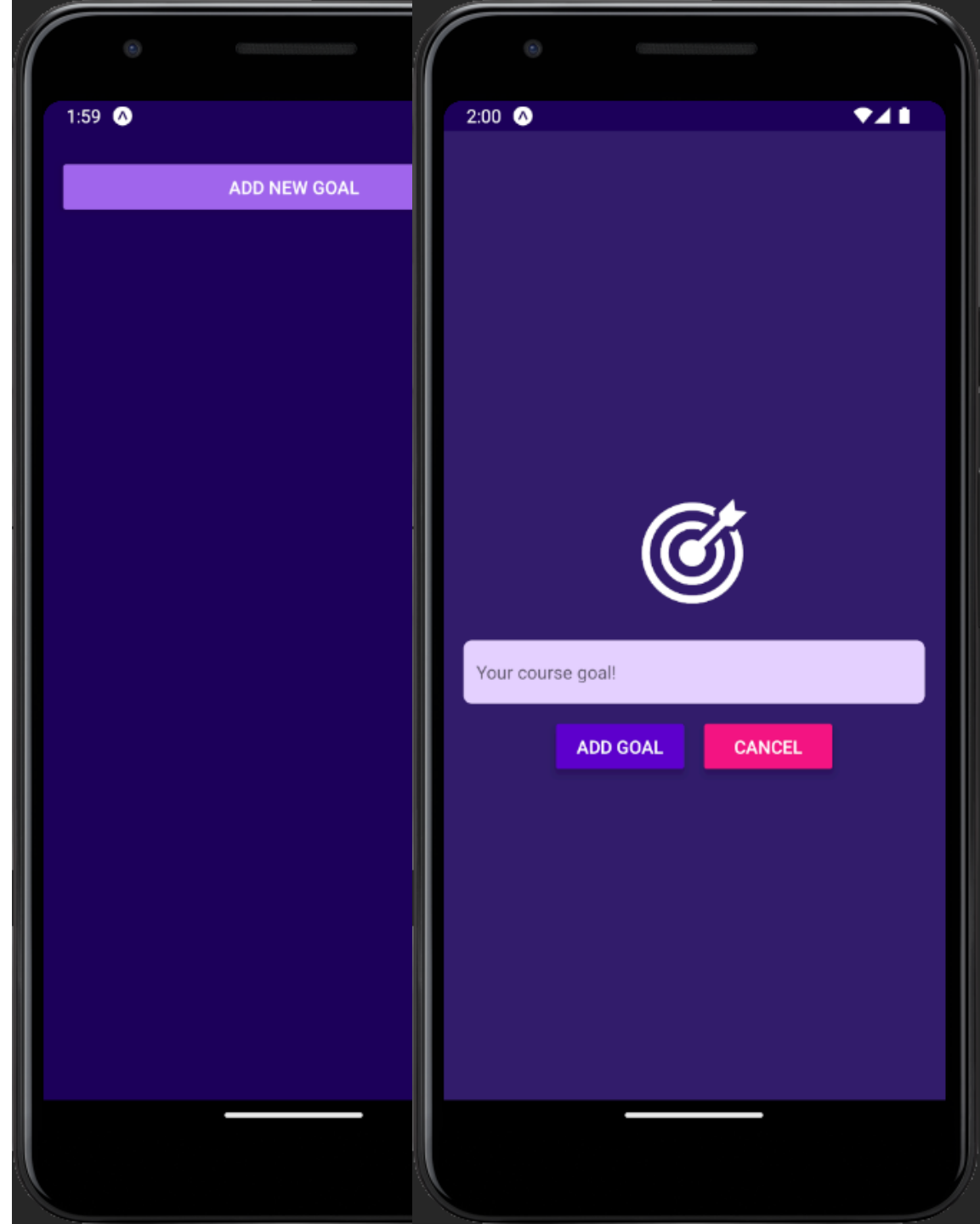


React Native

목표 기록 앱

목표 기록 앱 개요

- 사용자가 목표를 추가하고 리스트로 관리할 수 있는 앱
- 모달 창을 통해 목표 입력 및 리스트 항목 클릭으로 삭제 가능
- 주요 기능:
 - 목표 추가 기능
 - 목표 삭제 기능
 - 스크롤 가능한 목표 리스트
 - 모달 화면을 통한 목표 입력
 - 플랫폼별 스타일링



사용한 기법 및 개념

1. 상태 관리 (State Management)

- useState 혹은 사용하여 목표 리스트와 입력된 목표 텍스트 상태 관리

2. 컴포넌트 분리 (Component Separation)

- GoalItem과 GoalInput 컴포넌트 분리하여 모듈화 및 재사용 가능

3. 리스트 렌더링 (List Rendering)

- FlatList 컴포넌트를 사용하여 성능 최적화된 리스트 렌더링

4. 이벤트 처리 (Event Handling)

- 버튼 클릭 및 목표 항목 클릭 이벤트 처리하여 목표 추가 및 삭제

5. 모달 (Modal)

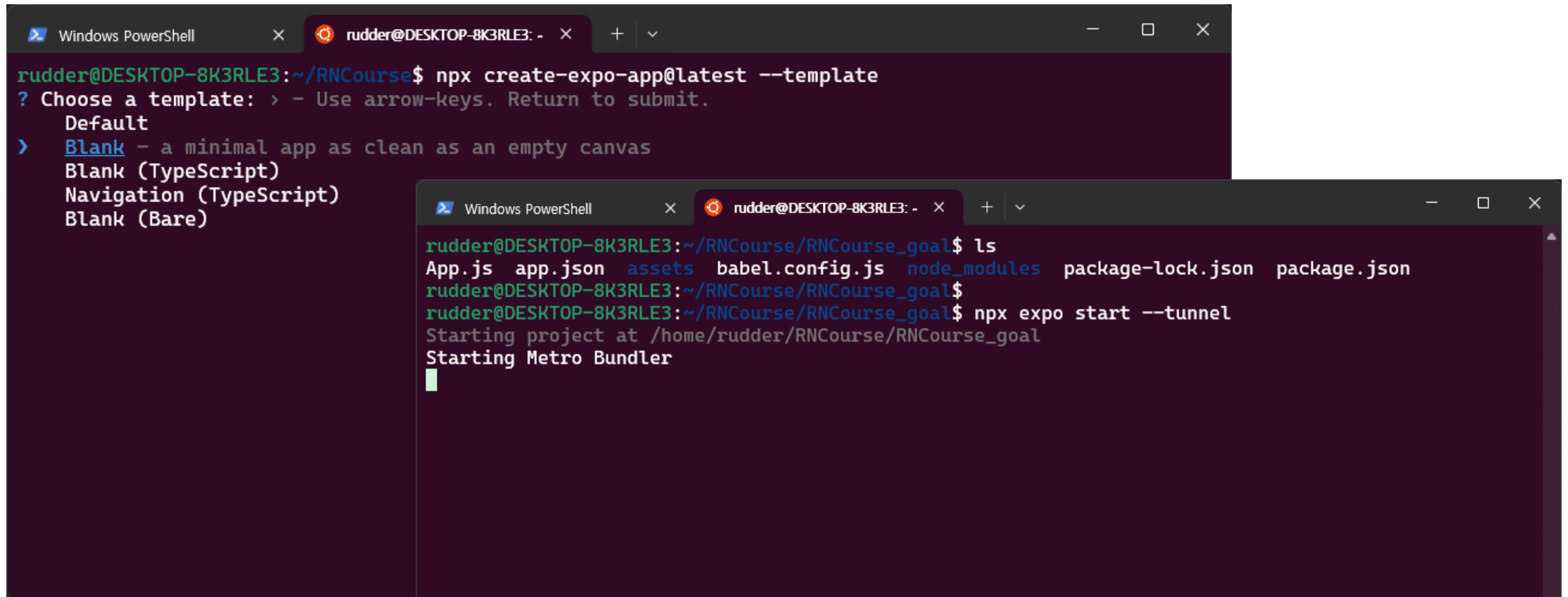
- Modal 컴포넌트를 사용하여 목표 추가 입력을 위한 모달 화면 구현

6. 스타일링 (Styling)

- StyleSheet를 사용하여 각 컴포넌트 스타일링
- Pressable과 android_ripple을 사용하여 클릭 시 시각적 피드백 제공
- 플랫폼별 스타일 차이 고려

0. 기본 앱 생성

- \$ npx create-expo-app@latest --template
- \$ npm install expo



The image shows two screenshots of a Windows PowerShell terminal window. The first screenshot shows the command `npx create-expo-app@latest --template` being executed, followed by a prompt to choose a template. The user selects the `Blank` template. The second screenshot shows the user running `ls` to list the files in the newly created directory, and then `npx expo start --tunnel` to start the development server.

```
Windows PowerShell
rudder@DESKTOP-8K3RLE3: ~\RNCourse$ npx create-expo-app@latest --template
? Choose a template: > - Use arrow-keys. Return to submit.
  Default
> Blank - a minimal app as clean as an empty canvas
  Blank (TypeScript)
  Navigation (TypeScript)
  Blank (Bare)

rudder@DESKTOP-8K3RLE3: ~\RNCourse\RNCourse_goal$ ls
App.js  app.json  assets  babel.config.js  node_modules  package-lock.json  package.json
rudder@DESKTOP-8K3RLE3: ~\RNCourse\RNCourse_goal$
rudder@DESKTOP-8K3RLE3: ~\RNCourse\RNCourse_goal$ npx expo start --tunnel
Starting project at /home/rudder/RNCourse/RNCourse_goal
Starting Metro Bundler
```

1. 기본 레이아웃 구현

- 설명:
 - React Native를 사용하여 간단한 목표 입력 앱의 기본 UI를 구현
 - 사용자가 목표를 입력할 수 있는 텍스트 입력 필드와 목표를 추가하는 버튼을 포함
 - 목표 리스트를 표시할 영역을 준비
 - StyleSheet.create를 사용하여 컴포넌트의 스타일을 정의
- 구현 내용:
 - App 컴포넌트를 생성하여 전체 애플리케이션 구조를 정의
 - 입력 영역(inputContainer)과 목표 리스트 영역(goalsContainer)으로 구성된 레이아웃 설계
 - TextInput 컴포넌트를 사용하여 사용자 입력을 받을 수 있는 필드 생성
 - Button 컴포넌트를 추가하여 목표 추가 기능을 위한 UI 요소 구현
 - 목표 리스트를 표시할 영역에 임시 텍스트 추가
 - StyleSheet.create를 사용하여 각 컴포넌트의 스타일을 정의하고 적용

1. 기본 레이아웃 구현

- 설명:
 - React Native를 사용하여 간단한 목표 입력 앱의 기본 UI를 구현
 - 사용자가 목표를 입력할 수 있는 텍스트 입력 필드와 목표를 추가하는 버튼을 포함
 - 목표 리스트를 표시할 영역을 준비
 - StyleSheet.create를 사용하여 컴포넌트의 스타일을 정의
- 구현 내용:
 - App 컴포넌트를 생성하여 전체 애플리케이션 구조를 정의
 - 입력 영역(inputContainer)과 목표 리스트 영역(goalsContainer)으로 구성된 레이아웃
 - TextInput 컴포넌트를 사용하여 사용자 입력을 받을 수 있는 필드 생성
 - Button 컴포넌트를 추가하여 목표 추가 기능을 위한 UI 요소 구현
 - 목표 리스트를 표시할 영역에 임시 텍스트 추가
 - StyleSheet.create를 사용하여 각 컴포넌트의 스타일을 정의하고 적용



1. 기본 레이아웃 구현

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

export default function App() {
  return (
    <View style={styles.appContainer}>
      <View style={styles.inputContainer}>
        <TextInput style={styles.textInput} placeholder="Your course goal!" />
        <Button title="Add Goal" />
      </View>
      <View style={styles.goalsContainer}>
        <Text>List of goals...</Text>
      </View>
    </View>
  );
}
```

```
const styles = StyleSheet.create({
  appContainer: {
    flex: 1,
    paddingTop: 50,
    paddingHorizontal: 16,
  },
  inputContainer: {
    flex: 1,
    flexDirection: "row",
    justifyContent: "space-between",
    alignItems: "center",
    marginBottom: 24,
    borderBottomWidth: 1,
    borderBottomColor: "#cccccc",
  },
  textInput: {
    borderWidth: 1,
    borderColor: "#cccccc",
    width: "70%",
    marginRight: 8,
    padding: 8,
  },
  goalsContainer: {
    flex: 5,
  },
});
```

2. 강의 목표 리스트 관리하기

- 설명:
 - useState를 사용하여 상태를 관리하고, 사용자가 입력한 목표를 리스트에 추가할 수 있는 기능을 구현
 - 입력된 텍스트를 상태로 저장하고, 버튼 클릭 시 리스트에 목표를 추가하여 화면에 출력
 - useState는 React에서 상태를 관리하는 기본 훅으로, 함수형 컴포넌트에서 상태를 추가할 수 있도록 함
- 구현 내용:
 - useState를 사용하여 enteredGoalText와 courseGoals 상태를 추가
 - 목표 입력 핸들러(goalInputHandler)와 목표 추가 핸들러(addGoalHandler)를 정의
 - 입력된 목표를 저장하고 리스트에 추가하는 기능을 구현
 - UI에 추가된 목표를 동적으로 렌더링

2. 강의 목표 리스트 관리하기

```
diff --git a/RNCourse/App.js b/RNCourse/App.js
index 4e45098..5a9c860 100644
--- a/RNCourse/App.js
+++ b/RNCourse/App.js
@@ -1,14 +1,36 @@
+import { useState } from "react";
import { StyleSheet, Text, TextInput, View, Button } from "react-native";

export default function App() {
+  const [enteredGoalText, setEnteredGoalText] = useState("");
+  const [courseGoals, setCourseGoals] = useState([]);
+
+  function goalInputHandler(enteredText) {
+    setEnteredGoalText(enteredText);
+  }
+
+  function addGoalHandler() {
+    console.log(enteredGoalText);
+    setCourseGoals((currentCourseGoals) => [
+      ...currentCourseGoals,
+      enteredGoalText,
+    ]);
+  }
+}
```

```
return (
  <View style={styles.appContainer}>
    <View style={styles.inputContainer}>
      <TextInput style={styles.textInput} placeholder="Your course goal!" />
      <Button title="Add Goal" />
      <TextInput
        style={styles.textInput}
        placeholder="Your course goal!"
        onChangeText={goalInputHandler}
      />
      <Button title="Add Goal" onPress={addGoalHandler} />
    </View>
    <View style={styles.goalsContainer}>
      <Text>List of goals...</Text>
      {courseGoals.map((goal) => (
        <Text key={goal}>{goal}</Text>
      ))}
    </View>
  </View>
);
```

3. iOS & Android 스타일링의 차이점

- 설명:
 - 각 목표 항목에 스타일을 적용하여 UI를 개선
 - `goalItem` 스타일을 통해 배경색, 패딩, 마진 등을 설정하여 리스트 항목의 가독성을 높임
 - React Native는 플랫폼별 스타일링을 지원하므로, iOS와 Android에서 다른 스타일을 적용할 수 있음
- 구현 내용:
 - `courseGoals` 리스트의 각 목표를 별도의 View 컴포넌트로 감싸고 스타일링
 - 목표 항목에 스타일을 적용하여 배경색과 텍스트 색상을 설정

3. iOS & Android 스타일링의 차이점

```
diff --git a/RNCourse/App.js b/RNCourse/App.js
index 5a9c860..344a963 100644
--- a/RNCourse/App.js
+++ b/RNCourse/App.js
@@ -29,7 +29,9 @@ export default function App() {
  </View>
  <View style={styles.goalsContainer}>
    {courseGoals.map((goal) => (
-     <Text key={goal}>{goal}</Text>
+     <View style={styles.goalItem} key={goal}>
+       <Text style={styles.goalText}>{goal}</Text>
+     </View>
    ))}
  </View>
</View>
@@ -61,4 +63,13 @@ const styles = StyleSheet.create({
  goalsContainer: {
    flex: 5,
  },
+  goalItem: {
+    margin: 8,
+    padding: 8,
+    borderRadius: 6,
+    backgroundColor: "#5e0acc",
+  },
+  goalText: {
+    color: "white",
+  },
});
```

4. ScrollView를 통해 콘텐츠를 스크롤 할 수 있도록 만들기

- 설명:

- ScrollView를 사용하여 리스트가 길어질 경우 스크롤할 수 있도록 기능 추가
- 이를 통해 긴 리스트도 화면에 잘 표시될 수 있도록 함
- ScrollView는 스크롤 가능한 컨테이너를 만들기 위해 사용되며, 모든 자식 요소를 한 번에 렌더링함

- 구현 내용:

- ScrollView를 사용하여 목표 리스트를 스크롤 가능하게 변경
- ScrollView 내부에 courseGoals를 맵핑하여 동적으로 렌더링

4. ScrollView를 통해 콘텐츠를 스크롤 할 수 있도록 만들기

```
diff --git a/RNCourse/App.js b/RNCourse/App.js
index 344a963..af3f257 100644
--- a/RNCourse/App.js
+++ b/RNCourse/App.js
@@ -1,5 +1,12 @@
  import { useState } from "react";
- import { StyleSheet, Text, TextInput, View, Button } from "react-native";
+ import {
+   StyleSheet,
+   Text,
+   TextInput,
+   View,
+   Button,
+   ScrollView,
+ } from "react-native";
```

```
export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState("");
@@ -28,11 +35,13 @@ export default function App() {
    <Button title="Add Goal" onPress={addGoalHandler} />
  </View>
  <View style={styles.goalsContainer}>
    {courseGoals.map((goal) => (
      <View style={styles.goalItem} key={goal}>
        <Text style={styles.goalText}>{goal}</Text>
      </View>
    ))}
  <ScrollView alwaysBounceVertical={false}>
    {courseGoals.map((goal) => (
      <View style={styles.goalItem} key={goal}>
        <Text style={styles.goalText}>{goal}</Text>
      </View>
    ))}
  </ScrollView>
</View>
</View>
);
```

5. FlatList를 통해 리스트 최적화하기

- 설명:

- FlatList는 대량의 데이터 렌더링에 최적화된 컴포넌트로, 스크롤 성능을 향상시킴
- FlatList를 사용하여 목표 리스트를 렌더링하고, keyExtractor로 고유 키를 설정하여 효율적으로 리스트를 관리
- FlatList는 성능 최적화를 위해 화면에 보이는 항목만 렌더링함

- 구현 내용:

- FlatList를 사용하여 목표 리스트를 최적화
- FlatList를 사용하여 성능을 향상시키고, keyExtractor를 사용해 각 항목의 고유 키 설정

5. FlatList를 통해 리스트 최적화하기

```
diff --git a/RNCourse/App.js b/RNCourse/App.js
index af3f257..d92570b 100644
```

```
--- a/RNCourse/App.js
```

```
+++ b/RNCourse/App.js
```

```
@@ -6,6 +6,7 @@ import {
```

```
  View,
  Button,
  ScrollView,
```

```
+ FlatList,
} from "react-native";
```

```
export default function App() {
```

```
@@ -20,7 +21,7 @@ export default function App() {
```

```
  console.log(enteredGoalText);
```

```
  setCourseGoals((currentCourseGoals) => [
```

```
    ...currentCourseGoals,
```

```
-   enteredGoalText,
```

```
+   { text: enteredGoalText, id: Math.random().toString()
  }]);
```

```
}
```

```
@@ -35,13 +36,21 @@ export default function App() {
  <Button title="Add Goal" onPress={addGoalHandler} />
```

```
</View>
```

```
<View style={styles.goalsContainer}>
```

```
-   <ScrollView alwaysBounceVertical={false}>
```

```
-     {courseGoals.map((goal) => (
```

```
-       <View style={styles.goalItem} key={goal}>
```

```
-         <Text style={styles.goalText}>{goal}</Text>
```

```
-       </View>
```

```
-     )})
```

```
-   </ScrollView>
```

```
+   <FlatList
```

```
+     data={courseGoals}
```

```
+     renderItem={({itemData}) => {
```

```
+       return (
```

```
+         <View style={styles.goalItem}>
```

```
+           <Text style={styles.goalText}>{itemData.item.text}</Text>
```

```
+         </View>
```

```
+       );
```

```
+     )}
```

```
+     keyExtractor={(item, index) => {
```

```
+       // return index;
```

```
+       return item.id;
```

```
+     }}
```

```
+     alwaysBounceVertical={false}
```

```
+   />
```

```
</View>
```

```
</View>
```

```
);
```

6. 컴포넌트를 작은 컴포넌트로 쪼개기

- 설명:
 - 컴포넌트를 작은 단위로 분리하여 재사용성과 유지보수성을 높임
 - GoallItem 컴포넌트를 별도로 분리하여 각 리스트 항목을 개별 컴포넌트로 관리
 - 이렇게 하면 코드의 가독성과 재사용성이 증가하고, 유지보수가 쉬워짐
- 구현 내용:
 - GoallItem 컴포넌트를 별도의 파일로 분리
 - GoallItem 컴포넌트를 App 컴포넌트에서 사용하도록 변경

6. 컴포넌트를 작은 컴포넌트로 쪼개기

```
diff --git a/RNCourse/App.js b/RNCourse/App.js
index d92570b..34273da 100644
--- a/RNCourse/App.js
+++ b/RNCourse/App.js
@@ -1,13 +1,7 @@
import { useState } from "react";
-import {
-  StyleSheet,
-  Text,
-  TextInput,
-  View,
-  Button,
-  ScrollView,
-  FlatList,
-} from "react-native";
+import { StyleSheet, TextInput, View, Button, FlatList } from "react-native";
+
+import GoalItem from "../components/GoalItem";
+
export default function App() {
  const [enteredGoalText, setEnteredGoalText] = useState("");
@@ -39,11 +33,7 @@ export default function App() {
  <FlatList
    data={courseGoals}
    renderItem={({itemData}) => {
-      return (
-        <View style={styles.goalItem}>
-          <Text style={styles.goalText}>{itemData.item.text}</Text>
-        </View>
-      );
+      return <GoalItem />;
    }}
    keyExtractor={(item, index) => {
      // return index;
```

```
@@ -81,13 +71,4 @@ const styles = StyleSheet.create({
  goalsContainer: {
    flex: 5,
  },
-  goalItem: {
-    margin: 8,
-    padding: 8,
-    borderRadius: 6,
-    backgroundColor: "#5e0acc",
-  },
-  goalText: {
-    color: "white",
-  },
-});
diff --git a/RNCourse/components/GoalInput.js b/RNCourse/components/GoalInput.js
new file mode 100644
index 0000000..e69de29
diff --git a/RNCourse/components/GoalItem.js b/RNCourse/components/GoalItem.js
new file mode 100644
index 0000000..4ab3694
```

6. 컴포넌트를 작은 컴포넌트로 쪼개기



```
--- /dev/null
+++ b/RNCourse/components/GoalItem.js
@@ -0,0 +1,24 @@
+// import React from "react"; react 및 rn 버전 업에 따라 명시적 선언 필요 없어짐
+import { StyleSheet, View, Text } from "react-native";
+
+function GoalItem() {
+  return (
+    <View style={styles.goalItem}>
+      <Text style={styles.goalText}>{itemData.item.text}</Text>
+    </View>
+  );
+}
+
+export default GoalItem;
+
+const styles = StyleSheet.create({
+  goalItem: {
+    margin: 8,
+    padding: 8,
+    borderRadius: 6,
+    backgroundColor: "#5e0acc",
+  },
+  goalText: {
+    color: "white",
+  },
+});
```

7. 프로퍼티 활용하기

- 설명:

- 프로퍼티(props)를 사용하여 부모 컴포넌트에서 자식 컴포넌트로 데이터를 전달
- Goalltem 컴포넌트에 text prop을 전달하여 목표 텍스트를 동적으로 렌더링
- props는 부모 컴포넌트가 자식 컴포넌트에 데이터를 전달하기 위한 메커니즘

- 구현 내용:

- Goalltem 컴포넌트에 text prop을 전달하여 목표 텍스트를 렌더링
- props를 사용하여 Goalltem 컴포넌트에서 텍스트를 동적으로 표시

7. 프로퍼티 활용하기

```
diff --git a/RNCourse/App.js b/RNCourse/App.js
index 34273da..3e45a84 100644
--- a/RNCourse/App.js
+++ b/RNCourse/App.js
@@ -33,7 +33,7 @@ export default function App() {
  <FlatList
    data={courseGoals}
    renderItem={({itemData}) => {
-     return <GoalItem />;
+     return <GoalItem text={itemData.item.text} />;
    }}
    keyExtractor={(item, index) => {
      // return index;
diff --git a/RNCourse/components/GoalItem.js b/RNCourse/components/GoalItem.js
index 4ab3694..178263d 100644
--- a/RNCourse/components/GoalItem.js
+++ b/RNCourse/components/GoalItem.js
@@ -1,10 +1,10 @@
// import React from "react"; react 및 rn 버전 업에 따라 명시적 선언 필요 없어짐
import { StyleSheet, View, Text } from "react-native";

function GoalItem() {
+function GoalItem(props) {
  return (
    <View style={styles.goalItem}>
-     <Text style={styles.goalText}>{itemData.item.text}</Text>
+     <Text style={styles.goalText}>{props.text}</Text>
    </View>
  );
}
```

8. “Goal Input” 컴포넌트로 작업하기

- 설명:
 - 입력 필드와 버튼을 포함하는 GoalInput 컴포넌트를 생성하여 사용자 입력을 처리
 - 입력된 목표를 부모 컴포넌트로 전달하여 리스트에 추가
 - 컴포넌트를 분리하면 코드가 더 깔끔해지고 재사용 가능성이 높아짐
- 구현 내용:
 - GoalInput 컴포넌트를 생성하여 입력 필드와 버튼을 포함
 - GoalInput 컴포넌트에서 목표 입력을 처리하고 onAddGoal prop을 통해 부모 컴포넌트로 전달

8. “Goal Input” 컴포넌트로 작업하기

```
diff --git a/RNCourse/App.js b/RNCourse/App.js
index 3e45a84..cafbc10 100644
--- a/RNCourse/App.js
+++ b/RNCourse/App.js
@@ -1,17 +1,13 @@
  import { useState } from "react";
- import { StyleSheet, TextInput, View, Button, FlatList } from "react-native";
+ import { StyleSheet, View, FlatList } from "react-native";
  import GoalItem from "../components/GoalItem";
+ import GoalInput from "../components/GoalInput";

  export default function App() {
-   const [enteredGoalText, setEnteredGoalText] = useState("");
-   const [courseGoals, setCourseGoals] = useState([]);
-
-   function goalInputHandler(enteredText) {
-     setEnteredGoalText(enteredText);
-   }
-
-   function addGoalHandler() {
+   function addGoalHandler(enteredGoalText) {
      console.log(enteredGoalText);
      setCourseGoals((currentCourseGoals) => [
        ...currentCourseGoals,
```

```
@@ -21,14 +17,7 @@ export default function App() {
  return (
    <View style={styles.appContainer}>
-     <View style={styles.inputContainer}>
-       <TextInput
-         style={styles.textInput}
-         placeholder="Your course goal!"
-         onChangeText={goalInputHandler}
-       />
-     <Button title="Add Goal" onPress={addGoalHandler} />
-   </View>
+   <GoalInput onAddGoal={addGoalHandler} />
+   <View style={styles.goalsContainer}>
+     <FlatList
+       data={courseGoals}
@@ -52,22 +41,6 @@ const styles = StyleSheet.create({
  padding: 50,
  padding: 16,
  },
- inputContainer: {
-   flex: 1,
-   flex: "row",
-   justify: "space-between",
-   align: "center",
-   margin: 24,
-   border: 1,
-   border: "#cccccc",
- },
- textInput: {
-   border: 1,
-   border: "#cccccc",
-   width: "70%",
-   margin: 8,
-   padding: 8,
- },
  goalsContainer: {
    flex: 5,
```

8. “Goal Input” 컴포넌트로 작업하기

```
diff --git a/RNCourse/components/GoalInput.js b/RNCourse/components/GoalInput.js
index e69de29..07da346 100644
```

```
--- a/RNCourse/components/GoalInput.js
```

```
+++ b/RNCourse/components/GoalInput.js
```

```
@@ -0,0 +1,48 @@
```

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

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

```
+
```

```
+function GoalInput(props) {
```

```
+  const [enteredGoalText, setEnteredGoalText] = useState("");
```

```
+
```

```
+  function goalInputHandler(enteredText) {
```

```
+    setEnteredGoalText(enteredText);
```

```
+  }
```

```
+
```

```
+  function addGoalHandler() {
```

```
+    props.onAddGoal(enteredGoalText);
```

```
+    setEnteredGoalText("");
```

```
+  }
```

```
+
```

```
+  return (
```

```
+    <View style={styles.inputContainer}>
```

```
+      <TextInput
```

```
+        style={styles.textInput}
```

```
+        placeholder="Your course goal!"
```

```
+        onChangeText={goalInputHandler}
```

```
+        value={enteredGoalText}
```

```
+      />
```

```
+      <Button title="Add Goal" onPress={addGoalHandler} />
```

```
+    </View>
```

```
+  );
```

```
+}
```

```
+export default GoalInput;
```

```
+
```

```
+const styles = StyleSheet.create({
```

```
+  inputContainer: {
```

```
+    flex: 1,
```

```
+    flexDirection: "row",
```

```
+    justifyContent: "space-between",
```

```
+    alignItems: "center",
```

```
+    marginBottom: 24,
```

```
+    borderBottomWidth: 1,
```

```
+    borderBottomColor: "#cccccc",
```

```
+  },
```

```
+  textInput: {
```

```
+    borderWidth: 1,
```

```
+    borderColor: "#cccccc",
```

```
+    width: "70%",
```

```
+    marginRight: 8,
```

```
+    padding: 8,
```

```
+  },
```

```
+});
```

9. Pressable 컴포넌트로 누르는 이벤트 처리하기

- 설명:

- Pressable 컴포넌트를 사용하여 목표 항목을 눌렀을 때 삭제 이벤트를 처리
- onDeleteItem prop을 통해 부모 컴포넌트로 이벤트를 전달하여 목표를 삭제할 수 있도록 함
- Pressable은 터치 가능한 UI 요소를 만들기 위한 컴포넌트로, 다양한 터치 이벤트를 처리할 수 있음

- 구현 내용:

- GoalItem 컴포넌트에서 Pressable 컴포넌트를 사용하여 목표 항목을 눌렀을 때 이벤트 처리
- onDeleteItem prop을 통해 삭제 이벤트를 부모 컴포넌트로 전달

9. Pressable 컴포넌트로 누르는 이벤트 처리하기

```
diff --git a/RNCourse/App.js b/RNCourse/App.js
index cafbc10..bc48159 100644
--- a/RNCourse/App.js
+++ b/RNCourse/App.js
@@ -15,6 +15,10 @@ export default function App() {
  });
}
```

```
+ function deleteGoalHandler() {
+   console.log("DELETE");
+ }
+
return (
  <View style={styles.appContainer}>
    <GoalInput onAddGoal={addGoalHandler} />
@@ -22,7 +26,12 @@ export default function App() {
    <FlatList
      data={courseGoals}
      renderItem={({itemData}) => {
-       return <GoalItem text={itemData.item.text} />;
+       return (
+         <GoalItem
+           text={itemData.item.text}
+           onDeleteItem={deleteGoalHandler}
+         />
+       );
      }}
      keyExtractor={(item, index) => {
        // return index;
```

```
diff --git a/RNCourse/components/GoalItem.js b/RNCourse/components/GoalItem.js
index 178263d..b056a54 100644
--- a/RNCourse/components/GoalItem.js
+++ b/RNCourse/components/GoalItem.js
@@ -1,11 +1,13 @@
  // import React from "react"; react 및 rn 버전 업에 따라 명시적 선언 필요 없어짐
- import { StyleSheet, View, Text } from "react-native";
+ import { StyleSheet, View, Text, Pressable } from "react-native";
+
function GoalItem(props) {
  return (
    <View style={styles.goalItem}>
      <Text style={styles.goalText}>{props.text}</Text>
    </View>
+    <Pressable onPress={props.onDeleteItem}>
+      <View style={styles.goalItem}>
+        <Text style={styles.goalText}>{props.text}</Text>
+      </View>
+    </Pressable>
  );
}
```

10. 아이템 삭제할 수 있게 만들기 & ID 사용하기

- 설명:

- 목표 항목에 고유 ID를 추가하여 각 항목을 식별
- deleteGoalHandler 함수를 통해 목표를 삭제하는 기능을 구현하여 사용자가 목표를 삭제할 수 있도록 함

- 구현 내용:

- 목표 항목에 고유 ID를 추가하여 각 항목을 식별
- deleteGoalHandler 함수를 통해 목표를 삭제하는 기능을 구현하여 사용자가 목표를 삭제할 수 있도록 함
- 각 항목에 고유 ID를 부여하면 삭제나 업데이트 시 특정 항목을 정확히 식별할 수 있음

10. 아이템 삭제할 수 있게 만들기 & ID 사용하기

```
diff --git a/RNCourse/App.js b/RNCourse/App.js
index bc48159..d47b3a9 100644
--- a/RNCourse/App.js
+++ b/RNCourse/App.js
@@ -15,8 +15,10 @@ export default function App() {
  });
}

- function deleteGoalHandler() {
-   console.log("DELETE");
+ function deleteGoalHandler(id) {
+   setCourseGoals((currentCourseGoals) => {
+     return currentCourseGoals.filter((goal) => goal.id !== id);
+   });
+ }

return (
@@ -29,6 +31,7 @@ export default function App() {
  return (
    <GoalItem
      text={itemData.item.text}
+     id={itemData.item.id}
      onDeleteItem={deleteGoalHandler}
    />
  );
}
```

```
diff --git a/RNCourse/components/GoalItem.js b/RNCourse/components/GoalItem.js
index b056a54..1b4dd62 100644
--- a/RNCourse/components/GoalItem.js
+++ b/RNCourse/components/GoalItem.js
@@ -3,7 +3,7 @@ import { StyleSheet, View, Text, Pressable } from "react-native";

function GoalItem(props) {
  return (
-    <Pressable onPress={props.onDeleteItem}>
+    <Pressable onPress={props.onDeleteItem.bind(this, props.id)}>
      <View style={styles.goalItem}>
        <Text style={styles.goalText}>{props.text}</Text>
      </View>
    </Pressable>
  );
}
```

11. Android 물결 효과 추가하기 & iOS 대안

- 설명:
 - Pressable 컴포넌트에 Android 물결 효과를 추가하여 클릭 시 시각적 피드백을 제공
 - iOS에서는 pressed 스타일을 사용하여 유사한 효과를 구현
 - android_ripple 속성은 Android에서 터치 시 물결 효과를 제공하는 속성
- 구현 내용:
 - Pressable 컴포넌트에 Android 물결 효과(android_ripple) 추가
 - iOS에서는 pressed 스타일을 적용하여 클릭 효과 제공

11. Android 물결 효과 추가하기 & iOS 대안

```
diff --git a/RNCourse/components/GoalItem.js b/RNCourse/components/GoalItem.js
index 1b4dd62..29a6c2f 100644
```

```
--- a/RNCourse/components/GoalItem.js
```

```
+++ b/RNCourse/components/GoalItem.js
```

```
@@ -3,11 +3,15 @@ import { StyleSheet, View, Text, Pressable } from "react-native";
```

```
function GoalItem(props) {
```

```
  return (
```

```
-   <Pressable onPress={props.onDeleteItem.bind(this, props.id)}>
```

```
-     <View style={styles.goalItem}>
```

```
+     <View style={styles.goalItem}>
```

```
+       <Pressable
```

```
+         android_ripple={{ color: "#dddddd" }}
```

```
+         onPress={props.onDeleteItem.bind(this, props.id)}
```

```
+         style={({ pressed }) => pressed && styles.pressedItem}
```

```
+       >
```

```
        <Text style={styles.goalText}>{props.text}</Text>
```

```
-     </View>
```

```
-   </Pressable>
```

```
+   </Pressable>
```

```
+ </View>
```

```
);
```

```
}
```

```
@@ -16,11 +20,14 @@ export default GoalItem;
```

```
const styles = StyleSheet.create({
```

```
  goalItem: {
```

```
    margin: 8,
```

```
-    padding: 8,
```

```
    borderRadius: 6,
```

```
    backgroundColor: "#5e0acc",
```

```
  },
```

```
+  pressedItem: {
```

```
+    opacity: 0.5,
```

```
+  },
```

```
  goalText: {
```

```
    color: "white",
```

```
+    padding: 8,
```

```
  },
```

```
});
```