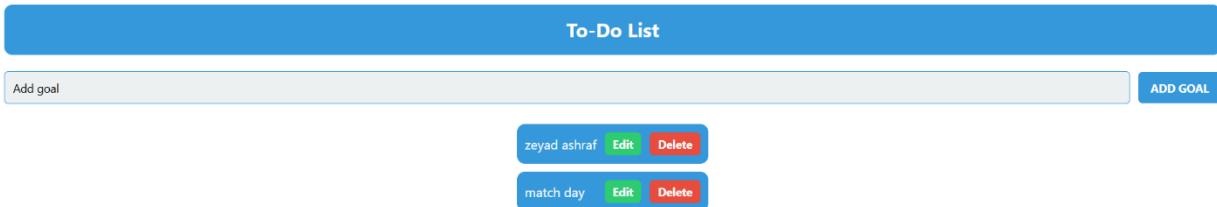
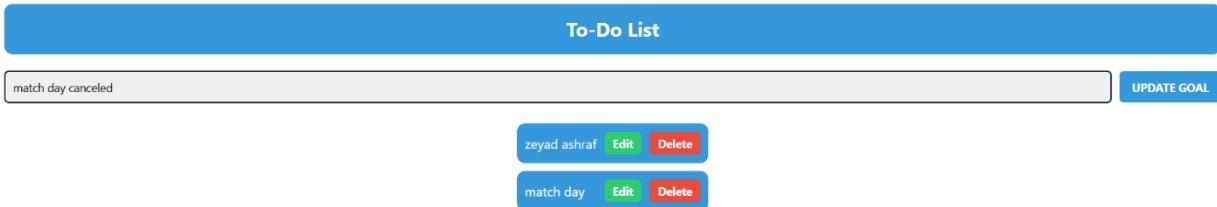


React Native To-Do List App Report

1. Introduction

This report provides an overview of the **React Native To-Do List App**, covering its design, colors, fonts, components, styling, and logic. The app allows users to add, edit, and delete goals dynamically. It is built using React Native and Expo, with a focus on simplicity and usability.

2. Screenshots of the App



3. Colors & Fonts Used

Colors:

Component	Color Code	Description
Background	#ecf0f1	Light Gray
Header	#3498db	Blue
Button	#3498db	Blue
Input Border	#3498db	Blue
Input BG	#ecf0f1	Light Gray
Goal Item BG	#3498db	Blue

Fonts:

- Default React Native fonts used.
 - `fontWeight: 'bold'` for important text elements.
-

4. Code Breakdown

App Component (App.js)

The App.js file contains the main logic and structure of the app. Key features include:

- State Management:
 - goal: Stores the current input value.
 - goals: Stores the list of goals.
 - editingIndex: Tracks the index of the goal being edited.
- Functions:
 - addGoal: Adds a new goal or updates an existing one.
 - deleteGoal: Removes a goal from the list.
 - startEditing: Populates the input field with the goal to be edited.
- Components:
 - TextInput: For entering goals.
 - TouchableOpacity: For buttons (Add/Update, Edit, Delete).
 - FlatList: For rendering the list of goals.

5. Features

Core Features:

1. Add Goal:

- Users can enter a goal in the input field and click "ADD GOAL" to add it to the list.

2. Edit Goal:

- Users can click the "Edit" button next to a goal to populate the input field with the goal's text.
 - Clicking "UPDATE GOAL" updates the goal in the list.

3. Delete Goal:

- Users can click the "Delete" button next to a goal to remove it from the list.

The image shows two side-by-side screenshots of a code editor interface, likely from Visual Studio Code, displaying parts of a React Native application.

Top Screenshot: The current tab is `App.js`. The code defines a functional component `App` that handles adding, editing, and deleting goals. It uses `useState` to manage the goal list and the index of the item being edited. The `addGoal` function adds a new goal or updates an existing one at the specified index. The `deleteGoal` function removes a goal from the list. The `startEditing` function sets the input field to the goal being edited and updates the editing index.

```
File Edit Selection View Go Run Terminal Help ↻ Assignment-1 ⚡ EXPLORER OPEN EDITORS JS App.js U ASSIGNMENT-1 OUTLINE TIMELINE
JS App.js > @ App > addGoal
1 import { StatusBar } from "expo-status-bar";
2 import { View, Text, TextInput, TouchableOpacity, FlatList, StyleSheet, Alert } from "react-native";
3 import React, { useState } from "react";
4
5 export default function App() {
6   const [goal, setGoal] = useState("");
7   const [goals, setGoals] = useState([ ]);
8   const [editingIndex, setEditingIndex] = useState(null); // Track which item is being edited
9
10  const addGoal = () => {
11    if (goal.trim().length === 0) return;
12    if (editingIndex !== null) [
13      // If editing, update the existing goal
14      const updatedGoals = [...goals];
15      updatedGoals[editingIndex] = goal;
16      setGoals(updatedGoals);
17      setEditingIndex(null); // Reset editing state
18    ] else [
19      // If not editing, add a new goal
20      setGoals([...currentGoals, goal]);
21    ]
22    setGoal(""); // Clear the input field
23  };
24
25  const deleteGoal = (index) => {
26    const updatedGoals = goals.filter((_, i) => i !== index);
27    setGoals(updatedGoals);
28  };
29
30  const startEditing = (index) => {
31    setGoal(goals[index]); // Set the input field to the goal being edited
32    setEditingIndex(index); // Set the editing index
33  };
34
35  return (
36    <View style={styles.container}>
37      <View style={styles.container2}>
38        <Text style={styles.header}>To-Do List</Text>
39        <View style={styles.inputContainer}>
40          <TextInput
41            style={styles.input}
42            placeholder="Add goal"
43            value={goal}
44            onChangeText={setGoal}
45          />
46          <TouchableOpacity style={styles.button} onPress={addGoal}>
47            <Text style={styles.buttonText}>
48              | {editingIndex !== null ? "UPDATE GOAL" : "ADD GOAL"}
49            </Text>
50          </TouchableOpacity>
51        </View>
52        <FlatList
53          data={goals}
54          renderItem={({ item, index }) => (
55            <View style={styles.goalItem}>
56              <Text style={styles.goalItemText}>{item}</Text>
57              <View style={styles.buttonContainer}>
58                <TouchableOpacity
59                  style={[styles.actionButton, styles.editButton]}
60                  onPress={() => startEditing(index)}
61                >
62                  <Text style={styles.buttonText}>Edit</Text>
63                </TouchableOpacity>
64                <TouchableOpacity
65                  style={[styles.actionButton, styles.deleteButton]}
66                  onPress={() => deleteGoal(index)}
67                >
68                  <Text style={styles.buttonText}>Delete</Text>
69                </TouchableOpacity>
70              </View>
71            </View>
72          )}
73        </FlatList>
74      </View>
75    </View>
76  );
77}
```

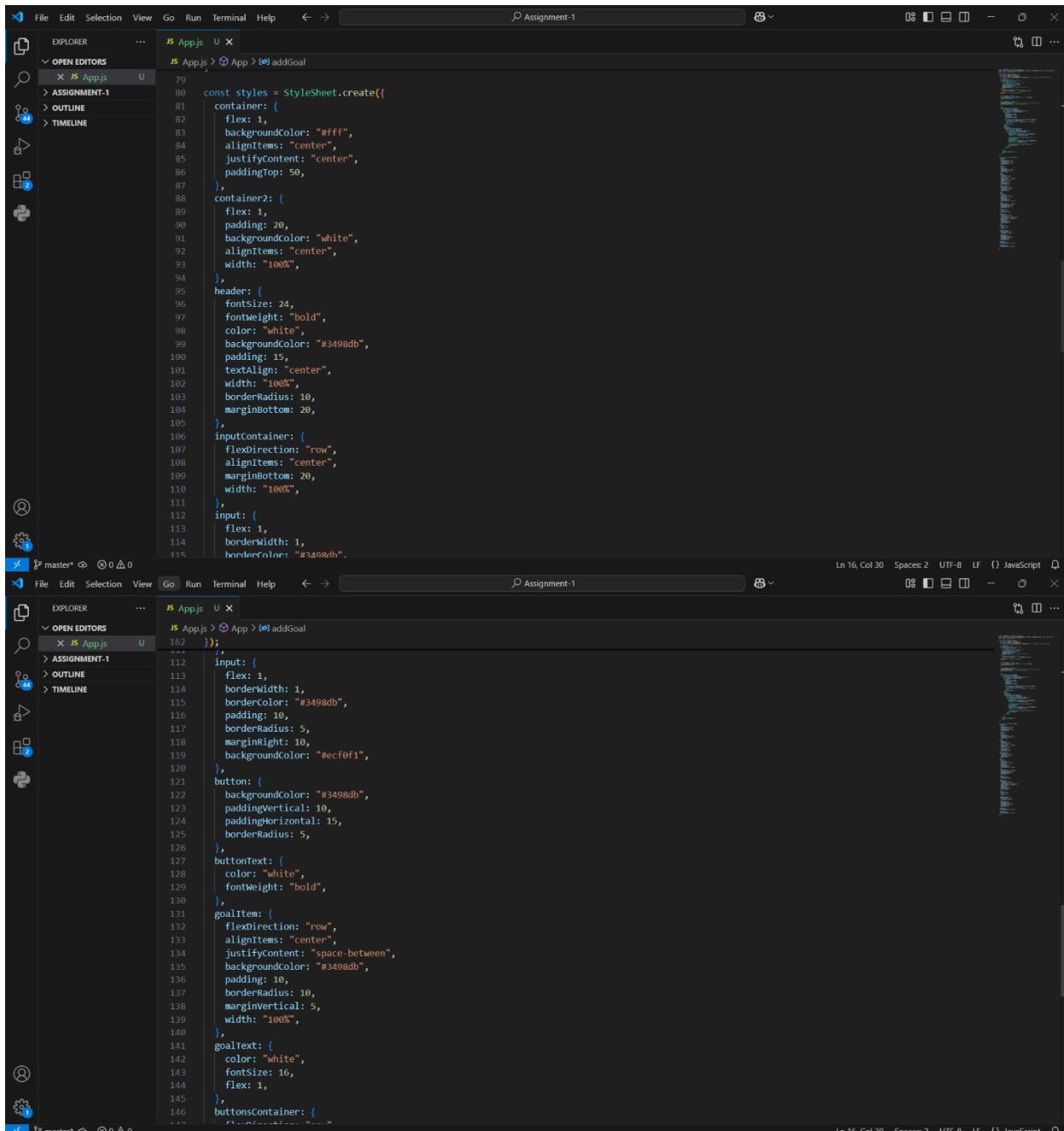
Bottom Screenshot: The current tab is `styles.js`. The code defines styles for the application components, including `header`, `inputContainer`, `button`, `buttonText`, `goalItem`, and `goalItemText`. The styles use `StyleSheet` to define colors, fonts, and layout properties for each element.

```
File Edit Selection View Go Run Terminal Help ↻ Assignment-1 ⚡ EXPLORER OPEN EDITORS JS App.js U ASSIGNMENT-1 OUTLINE TIMELINE
JS App.js > @ App > addGoal
1 import { StyleSheet } from "react-native";
2
3 const styles = StyleSheet.create({
4   header: {
5     color: "#fff",
6     backgroundColor: "#007bff",
7     padding: 10,
8     margin: 10,
9     text-align: "center",
10    },
11   inputContainer: {
12     width: "100%",
13     height: 40,
14     margin: 10,
15     padding: 5,
16     border: 1px solid #ccc,
17     border-radius: 5px,
18   },
19   button: {
20     width: 40,
21     height: 40,
22     margin: 5,
23     padding: 5,
24     border: 1px solid #007bff,
25     border-radius: 5px,
26     background: "#007bff",
27     color: "#fff",
28     display: flex,
29     align-items: center,
30     justify-content: center,
31   },
32   buttonText: {
33     color: "#fff",
34     fontWeight: "bold",
35   },
36   goalItem: {
37     padding: 10,
38     margin: 10,
39     border-bottom: 1px solid #ccc,
40   },
41   goalItemText: {
42     color: "#000",
43     margin: 5,
44   },
45   buttonContainer: {
46     display: flex,
47     align-items: center,
48     gap: 10,
49   },
50   editButton: {
51     border: 1px solid #007bff,
52     border-radius: 5px,
53     padding: 5 10,
54     color: "#007bff",
55   },
56   deleteButton: {
57     border: 1px solid #dc3545,
58     border-radius: 5px,
59     padding: 5 10,
60     color: "#dc3545",
61   }
62 });
63
64 module.exports = styles;
```

Styling (styles object)

The StyleSheet object defines the app's visual design. Key styles include:

- Container: Centers the app content and sets the background color.
- Header: Styles the app title with a blue background and white text.
- Input Container: Arranges the input field and button horizontally.
- Goal Item: Styles each goal item with a blue background and white text.
- Buttons: Styles for the Edit (green) and Delete (red) buttons.



The image shows two side-by-side screenshots of a code editor interface, likely VS Code, displaying two versions of a file named `App.js`. Both screenshots show the same code structure but with different styling blocks visible.

```
JS App.js  U x
JS App.js > ⚡ App > ↗ addGoal
79
80 const styles = StyleSheet.create({
81   container: {
82     flex: 1,
83     backgroundColor: "#fff",
84     alignItems: "center",
85     justifyContent: "center",
86     paddingTop: 50,
87   },
88   container2: {
89     flex: 1,
90     padding: 20,
91     backgroundColor: "white",
92     alignItems: "center",
93     width: "100%",
94   },
95   header: {
96     fontSize: 24,
97     fontWeight: "bold",
98     color: "white",
99     backgroundColor: "#3498db",
100    padding: 15,
101    textAlign: "center",
102    width: "100%",
103    borderRadius: 10,
104    marginBottom: 20,
105  },
106  inputContainer: {
107    flexDirection: "row",
108    alignItems: "center",
109    marginBottom: 20,
110    width: "100%",
111  },
112  input: {
113    flex: 1,
114    borderwidth: 1,
115    bordercolor: "#3498db",
116  },
117  button: {
118    backgroundColor: "#3498db",
119    paddingVertical: 10,
120    paddingHorizontal: 15,
121    borderRadius: 5,
122  },
123  buttonText: {
124    color: "white",
125    fontWeight: "bold",
126  },
127  goalItem: {
128    flexDirection: "row",
129    alignItems: "center",
130    justifyContent: "space-between",
131    backgroundColor: "#3498db",
132    padding: 10,
133    borderRadius: 10,
134    marginVertical: 5,
135    width: "100%",
136  },
137  goalText: {
138    color: "white",
139    fontsize: 16,
140    flex: 1,
141  },
142  buttonsContainer: {
```

The top screenshot shows the first half of the `StyleSheet.create` block, focusing on the `container`, `header`, and `inputContainer` styles. The bottom screenshot shows the second half, focusing on the `button`, `buttonText`, `goalItem`, `goalText`, and `buttonsContainer` styles. Both screenshots include line numbers and syntax highlighting.

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Assignment-1
- Editor Area:** JS App.js (active tab), showing code for styling components like goalItem, goalText, buttonsContainer, actionButton, editButton, and deleteButton.
- Left Sidebar:** OPEN EDITORS (JS App.js), ASSIGNMENT-1, OUTLINE, and TIMELINE.
- Bottom Status Bar:** Ln 16, Col 30, Spaces: 2, UTF-8, LF, JavaScript.

5. Conclusion

The **React Native To-Do List App** is a simple yet powerful tool for managing tasks. It demonstrates the use of React Native components, state management, and styling to create a clean and functional user interface. With the addition of **Edit** and **Delete** functionality, the app provides a more complete user experience. Future enhancements, such as local storage and animations, could further improve the app's usability and appeal.