Mobile computing task 1 Report

My-To-Do-List

Submitted to:

Eng. Rania El-Sayed

Presented by:

El-Sherif Adel Abdelhamid

Link: https://snack.expo.dev/@el-sherif_adel/my-to-do-list

Short Pref: -

This code is a simple React Native to-do list application that allows users to manage a list of goals.

Here's a conceptual breakdown: -

1. **State Management:**
    - **goal:** Holds the current text input from the user.
    - **goalList:** Stores an array of goal objects, each with a unique ID and text.
    - **editingGoal:** Keeps track of whether an existing goal is being edited.
2. **User Input & Submission:**
    - A **TextInput** component lets the user type a goal.
    - The input field listens for two events to submit a goal:
        - **Pressing the "done" or enter key:** via the onSubmitEditing prop, which triggers the addition or update function.

        - **Tapping the "ADD" button:** a TouchableOpacity that also triggers the same function.

3. **Adding or Updating Goals:**
    - When a user submits the goal, the function checks if it's an update (editing an existing goal) or a new addition.
    - If updating, it finds the corresponding goal in the list and changes its text.
    - If adding, it creates a new goal with a unique ID (using the current timestamp) and appends it to the list.
4. **Editing and Deleting Goals:**
    - Each goal is rendered using a custom **GoalItem** component that displays the goal text along with edit and delete icons.
    - Tapping the **edit icon** sets the goal text back into the input field for modification.
    - Tapping the **delete icon** removes the goal from the list.
5. **Display:**
    - A **FlatList** is used to efficiently render the list of goals.
    - Each goal is styled using a StyleSheet to provide a consistent look and feel.

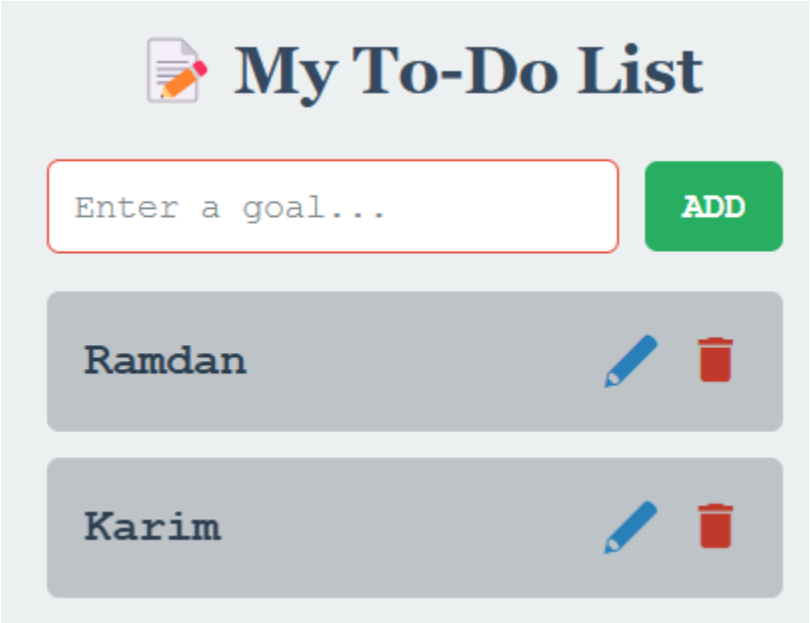
Overall:

- The code demonstrates key concepts of React Native development, including state management, user interaction handling (text input, button press, and keyboard events), conditional rendering, and component-based design. This modular approach makes it easier to understand, maintain, and extend the app as needed.

Screenshots of my design

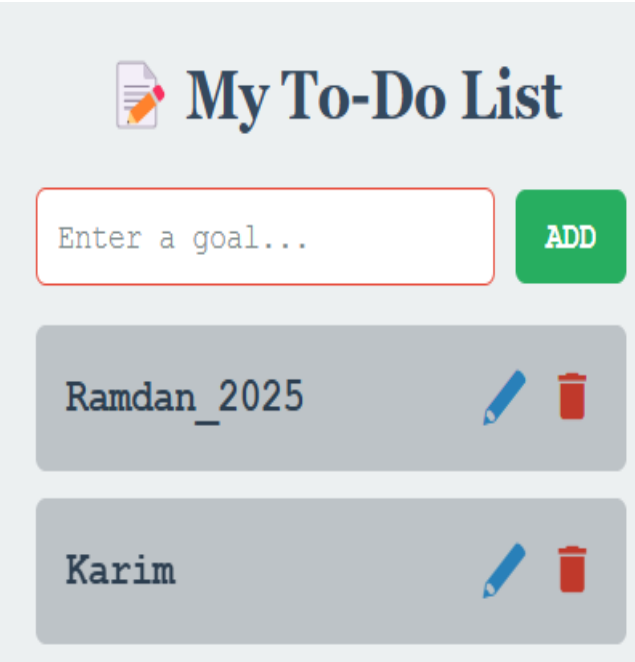
1- Main page:

2- Adding a Task:

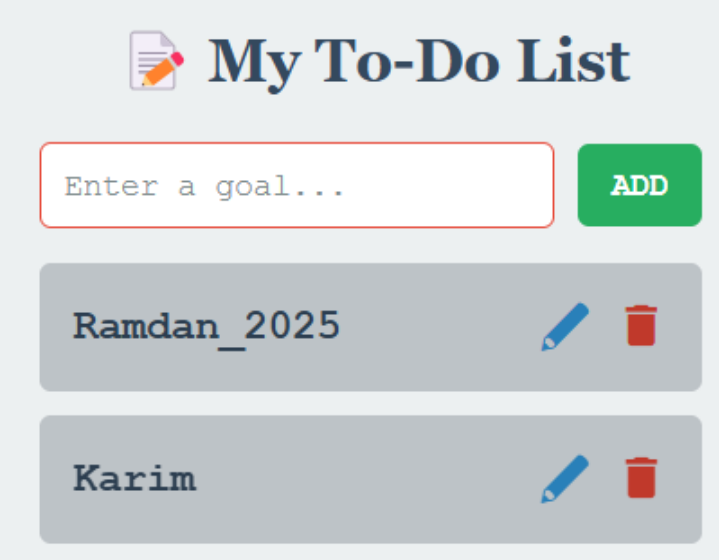


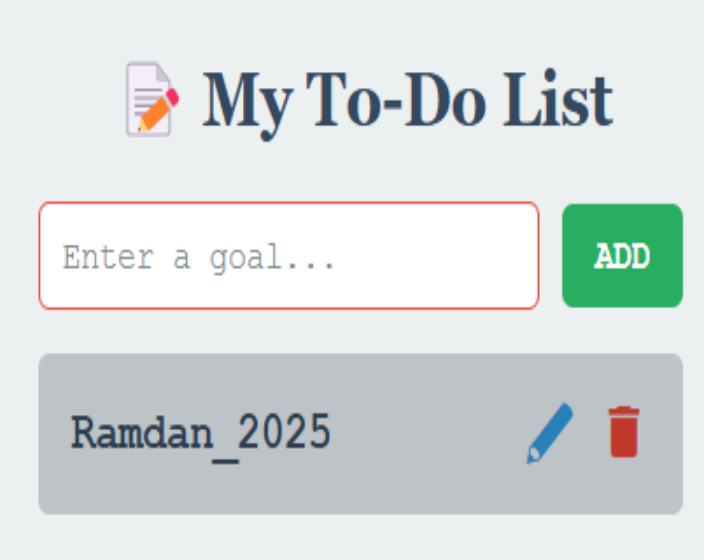
Note: Submitting a goal to the to-do list can be done either by pressing the Enter key on the keyboard or by tapping the ADD button.

3- Editing a Task – Press the edit icon (✎), the text will appear in the input field:



4- Deleting a Task – Press the delete icon (🗑)





Colors and Fonts used in your project:

Colors:

- **Container Background:** #ecf0f1
- **Title Text:** #34495e
- **Input Field:**
  - Border: #e74c3c
  - Text: #2c3e50
  - Placeholder: #7f8c8d
- **Add Button:** #27ae60
- **Goal Item Background:** #bdc3c7
- **Goal Text:** #2c3e50
- **Icons:**
  - Edit Icon: #2980b9
  - Delete Icon: #c0392b

Code Snap shoot

```
import React, { useState } from "react";
import {
  View,
  Text,
  TextInput,
  TouchableOpacity,
  FlatList,
  StyleSheet,
} from "react-native";
import { MaterialIcons, Entypo } from "@expo/vector-icons";

// Component to render each individual goal item in the list
const GoalItem = ({ item, onEdit, onDelete }) => (
  <View style={styles.goalItem}>
    {/* Display the goal text */}
    <Text style={styles.goalText}>{item.text}</Text>
    {/* Container for the edit and delete icons */}
    <View style={styles.iconContainer}>
      {/* When pressed, triggers editing mode for this goal */}
      <TouchableOpacity onPress={() => onEdit(item)}>
        <Entypo name="edit" size={24} color="#2980b9" />
      </TouchableOpacity>
      {/* When pressed, deletes this goal from the list */}
      <TouchableOpacity onPress={() => onDelete(item.id)}>
        <MaterialIcons name="delete" size={24} color="#c0392b" />
      </TouchableOpacity>
    </View>
  </View>
);

// Main application component
export default function App() {
  // State to hold the current text input value (i.e., the goal text)
  const [goal, setGoal] = useState("");
```

```
// State to hold the list of goals; each goal has an id and text
const [goalList, setGoalList] = useState([]);
// State to track if we're currently editing an existing goal
const [editingGoal, setEditingGoal] = useState(null);

// Function to add a new goal or update an existing one
const handleAddOrUpdateGoal = () => {
  // Ignore if the input text is empty or only spaces
  if (goal.trim().length === 0) return;

  // Check if we are in edit mode
  if (editingGoal) {
    // Update the existing goal by mapping through the current list
    setGoalList((prevGoals) =>
      prevGoals.map((item) =>
        // If the id matches, update its text; otherwise, keep the goal as is
        item.id === editingGoal.id ? { ...item, text: goal } : item
      )
    );
    // Clear editing state since the update is complete
    setEditingGoal(null);
  } else {
    // Create a new goal object and add it to the list
    // Date.now() is used here to generate a unique id based on the current timestamp
    setGoalList((prevGoals) => [
      ...prevGoals,
      { id: Date.now().toString(), text: goal },
    ]);
  }
  // Clear the input field after adding or updating the goal
  setGoal("");
};
```

```
68    // Function to delete a goal based on its id
69    const handleDeleteGoal = (id) => {
70      // Filter out the goal with the matching id from the list
71      setGoalList((prevGoals) => prevGoals.filter((goal) => goal.id !== id));
72    };
73
74    // Function to enable edit mode for a selected goal
75    const handleEditGoal = (goalItem) => {
76      // Set the current input to the goal's text so it can be edited
77      setGoal(goalItem.text);
78      // Set the editingGoal state to the selected goal to indicate edit mode
79      setEditingGoal(goalItem);
80    };
81
82    return (
83      <View style={styles.container}>
84        {/* App title */}
85        <Text style={styles.title}>📝 My To-Do List</Text>
86
87        {/* Input container for adding a new goal */}
88        <View style={styles.inputContainer}>
89          <TextInput
90            style={styles.input}
91            placeholder="Enter a goal..."
92            placeholderTextColor="#7f8c8d"
93            value={goal}
94            // Updates the 'goal' state as the user types
95            onChangeText={setGoal}
96            // When the user presses the "enter" or "done" key on the keyboard,
97            // this event is triggered to add or update the goal.
98            onSubmitEditing={handleAddOrUpdateGoal}
```

```
          onSubmitEditing={handleAddOrUpdateGoal}
          // Change the return key on the keyboard to display "done"
          returnKeyType="done"
        />
        {/* Button to add or update the goal */}
        <TouchableOpacity
          style={styles.addButton}
          onPress={handleAddOrUpdateGoal}
        >
          {/* Button text changes based on whether we're editing or adding */}
          <Text style={styles.addButtonText}>
            {editingGoal ? "UPDATE" : "ADD"}
          </Text>
        </TouchableOpacity>
      </View>

      {/* List displaying all the goals */}
      <FlatList
        data={goalList}
        keyExtractor={(item) => item.id}
        renderItem={({ item }) => (
          // Render each goal item using the GoalItem component
          <GoalItem
            item={item}
            onEdit={handleEditGoal}
            onDelete={handleDeleteGoal}
          />
        )}
        style={styles.listContainer}
      />
    </View>
  );
```

```
// Styles for the components
const styles = StyleSheet.create({
  container: {
    flex: 1, // Occupies the full screen
    padding: 20, // Adds padding around the container
    backgroundColor: "#ecf0f1", // Light gray background
  },
  title: {
    fontSize: 26, // Larger font size for the title
    fontWeight: "bold", // Bold text for the title
    textAlign: "center", // Center the title horizontally
    marginBottom: 20, // Space below the title
    color: "#34495e", // Dark blue-gray color
    fontFamily: "Georgia", // Custom font family for the title
  },
  inputContainer: {
    flexDirection: "row", // Arranges TextInput and button side by side
    alignItems: "center", // Aligns children vertically centered
    marginBottom: 10, // Space below the input container
  },
  input: {
    flex: 1, // Takes up available horizontal space
    borderWidth: 1, // Border width around the input field
    borderColor: "#e74c3c", // Red border color
    padding: 10, // Padding inside the input field
    borderRadius: 5, // Rounded corners for the input
    backgroundColor: "#fff", // White background for the input
    fontFamily: "Courier New", // Custom font for input text
    color: "#2c3e50", // Text color inside the input
  },
  addButton: {
    backgroundColor: "#27ae60", // Green background for the button
    paddingVertical: 10, // Vertical padding for the button
    paddingHorizontal: 15, // Horizontal padding for the button
```

```
    borderRadius: 5, // Rounded corners for the button
    marginLeft: 10, // Space between the input and button
  },
  addButtonText: {
    color: "#fff", // White text color for the button
    fontWeight: "bold", // Bold text in the button
    fontFamily: "Courier New", // Custom font for the button text
  },
  listContainer: {
    flex: 1, // Takes up remaining space for the list
  },
  goalItem: {
    flexDirection: "row", // Arranges goal text and icons in a row
    justifyContent: "space-between", // Distributes space between text and icons
    alignItems: "center", // Centers items vertically
    padding: 15, // Padding inside each goal item
    backgroundColor: "#bdc3c7", // Background color for the goal item
    borderRadius: 5, // Rounded corners for each goal item
    marginVertical: 5, // Vertical spacing between items
  },
  goalText: {
    fontSize: 18, // Font size for the goal text
    fontWeight: "bold", // Bold text for the goal
    color: "#2c3e50", // Text color for the goal
    fontFamily: "Courier New", // Custom font for the goal text
  },
  iconContainer: {
    flexDirection: "row", // Arranges icons side by side
    gap: 10, // Space between the icons (supported in newer React Native versions)
  },
});

```