**To-Do List App Documentation**

**Overview**

**Link:** **https://snack.expo.dev/@shelbyk342/recipes-tts**

This application is a simple to-do list manager built using React Native. It allows users to add, delete, and mark tasks as completed, while also providing options for filtering tasks by their status (All, Active, or Completed). Additionally, the app includes a text-to-speech feature to read task names aloud. Data is stored persistently using AsyncStorage, ensuring that tasks are saved even after the app is closed and reopened.

**Libraries and Components Used**

1. **React & React Native**:
   * useState: Used to manage the application's state.
   * useEffect: A hook for side effects, used to load tasks from AsyncStorage when the app starts.
   * FlatList: A React Native component used to efficiently render a list of tasks.
   * Text, TextInput, Button, TouchableOpacity, StyleSheet, SafeAreaView, ScrollView: Various React Native components used for UI design.
2. **AsyncStorage**:
   * A library for local storage, enabling persistent data storage (tasks) across app restarts.
3. **Expo Speech**:
   * expo-speech: Provides text-to-speech functionality to read task names aloud.

**App Structure**

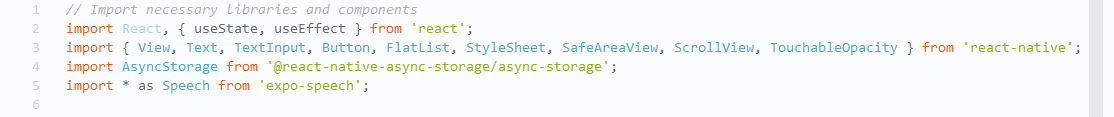
The app's main component is the App function, which contains the following key functionalities:

1. **State Management**:
   * tasks: Stores the list of tasks. Each task has properties like id, name, and completed.
   * taskInput: Holds the value of the input field used to add new tasks.
   * selectedTaskId: Stores the ID of the currently selected task, enabling the text-to-speech feature.
   * filter: Stores the current filter selected by the user (All, Active, or Completed).
2. **useEffect**:
   * The useEffect hook runs when the app first mounts and attempts to load any saved tasks from AsyncStorage. If no tasks are found, it initializes with an empty list.
3. **Task Functions**:
   * addTask: Adds a new task to the list. Each task has a unique ID (generated with Date.now()) and a default completed status of false. Tasks are stored in AsyncStorage after being added to the state.
   * deleteTask: Deletes a task by its ID. After the task is deleted, the tasks list is updated and saved to AsyncStorage.
   * toggleCompletion: Marks a task as completed or incomplete. When the completion status changes, the list is updated and saved.
   * getFilteredTasks: Filters tasks based on the selected filter (All, Active, or Completed).
   * readTask: Uses expo-speech to read the name of a task aloud.
4. **UI Components**:
   * **Task List**: Renders each task with a checkbox (to mark the task as completed), buttons for "Edit" and "Delete", and a "Read Aloud" option.
   * **Filters**: Provides the ability to filter tasks based on their status (All, Active, or Completed).
   * **Add Task Input**: A text input field and button to add new tasks to the list.

**App Flow**

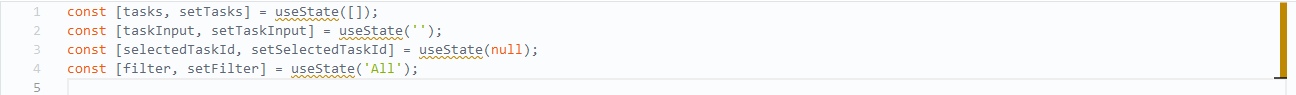
1. **Loading Tasks**:
   * When the app loads, useEffect triggers the loadTasks function, which attempts to fetch the stored tasks from AsyncStorage. If tasks are found, they are loaded into the state.
2. **Adding Tasks**:
   * The user can type a new task in the TextInput field and click the "Add Task" button. If the input is non-empty, a new task is added to the list and saved to AsyncStorage.
3. **Task Completion**:
   * Each task has a checkbox. When clicked, the task's completion status is toggled between completed and incomplete. Tasks are saved after the status update.
4. **Filtering Tasks**:
   * The user can filter tasks using the "All", "Active", or "Completed" buttons. The displayed tasks update based on the current filter selection.
5. **Deleting Tasks**:
   * Each task has a "Delete" button that removes it from the list and updates AsyncStorage to reflect the changes.
6. **Reading Tasks Aloud**:
   * The user can click the "Read Aloud" button to hear the name of the task via text-to-speech using the expo-speech library.
7. **Clear Completed Tasks**:
   * An optional feature (not implemented in the current code) could include a button to clear all completed tasks from the list.

**Code Breakdown**



* **React**: Core library for building the app.
* **React Native components**: Essential UI components for layout and interactivity.
* **AsyncStorage**: Used for saving and loading tasks locally.
* **expo-speech**: Used for the text-to-speech functionality.

**State Management**

****

* **tasks: Stores the list of tasks.**
* **taskInput: Manages the input field for new tasks.**
* **selectedTaskId: Tracks which task is selected for reading aloud.**
* **filter: Tracks the current filter applied to the task list.**
* **tasks: Stores the list of tasks.**
* **taskInput: Manages the input field for new tasks.**
* **selectedTaskId: Tracks which task is selected for reading aloud.**
* **filter: Tracks the current filter applied to the task list.**

**Task Functions**

* **addTask: Adds a new task if the input is not empty.**
* **deleteTask: Deletes a task from the list based on its ID.**
* **toggleCompletion: Toggles the completion status of a task.**
* **getFilteredTasks: Filters tasks based on the selected filter.**
* **readTask: Uses the expo-speech API to read a task aloud.**

**UI Layout**

* **Task Input: A TextInput for entering new tasks.**
* **Filter Buttons: Three buttons for filtering tasks by status.**
* **Task List: A FlatList renders tasks with completion checkboxes and buttons for each task.**
* **Buttons for Actions: Tasks have buttons for "Delete", "Read Aloud", and selecting a task.**

**Styles**

**A close up of text

Description automatically generated**

* **General Styles: Defines padding, margins, font sizes, and text styles.**
* **Task Item Styles: The tasks are styled with checkboxes, text for the task name, and buttons for interactions.**
* **Filter Styles: Ensures the filter buttons are displayed in a row, with the active filter highlighted.**

**Conclusion**

**This is a simple but effective to-do list app that allows users to manage tasks with the ability to add, delete, mark as completed, and filter tasks. The app uses AsyncStorage to persist data and expo-speech to read task names aloud, making it more interactive. The structure is flexible, allowing easy future additions, such as task prioritization or adding due dates.**

**Manual Testing**

**1. Test Setup**

* **Run the app on a device or emulator (Android or iOS).**
* **Ensure the following features are available:**
  + **Adding a new task**
  + **Deleting a task**
  + **Marking a task as completed**
  + **Filtering tasks (All, Active, Completed)**
  + **Reading a task aloud using the "Read Aloud" button**

**Test Scenarios:**

**1. Adding Tasks**

* **Test Steps: Entered tasks using the input field and submitted them.**
* **Expected Outcome: Tasks should appear in the list with the correct names.**
* **Result:**
  + **Task "Buy groceries" added successfully. Displayed correctly.**
  + **Task "Complete project" added successfully. Displayed correctly.**

**2. Deleting Tasks**

* **Test Steps: Deleted a few tasks from the task list.**
* **Expected Outcome: Tasks should be removed from the display and not be present in AsyncStorage.**
* **Result:**
  + **Task "Buy groceries" deleted from display and AsyncStorage.**
  + **Task "Complete project" deleted from display and verified removal from AsyncStorage.**

**3. Marking Tasks as Completed**

* **Test Steps: Toggled tasks to mark as completed.**
* **Expected Outcome: Task appearance should change to strikethrough, and the status should be updated in AsyncStorage.**
* **Result:**
  + **Task "Buy groceries" marked as completed with strikethrough. Status correctly updated in AsyncStorage.**
  + **Task "Complete project" marked as completed with strikethrough. Status correctly updated in AsyncStorage.**

**4. Filtering Tasks**

* **Test Steps: Applied "All," "Active," and "Completed" filters.**
* **Expected Outcome: Correct tasks should display based on the selected filter.**
* **Result:**
  + **"All" filter shows both "Buy groceries" and "Complete project."**
  + **"Active" filter displays only active tasks as expected.**
  + **"Completed" filter displays only completed tasks as expected.**

**5. Text-to-Speech**

* **Test Steps: Pressed the "Read Aloud" button for each task.**
* **Expected Outcome: Task name should be spoken correctly.**
* **Result:**
  + **Task "Buy groceries" was read aloud correctly.**
  + **Task "Complete project" was read aloud correctly.**

**Wire Frame**

**A blank to do list

Description automatically generated**