# Coding test for React Native

## Description:

This code challenge aims to test the general ability for building React Native apps as well as familiarity with the framework.

Within ./oct0 folder you will find a simple RN TODO app lacking in functionality, your job would be to add some missing functionality & improvements.

The app has a basic navigation setup which consists of 4 screens see ./oct0/src/screens

- TaskLists: For displaying the current ongoing tasks.
- History: For displaying the previously completed tasks.
- Settings: For app-wide settings.
- TaskEditor: For adding / editing a task.

The whole app is based on the `Task` model which is defined in ./oct0/src/models/Task.ts

```
export type Task = {
  id: string;
  title: string;
  description: string | null;
  completed: boolean;
};
```

The state management is handled by xstate, and consists of a basic machine that can be visualized interactively at https://stately.ai/viz. All of which live in ./oct0/src/controllers/globalController.ts

- Copy this code snippet and paste it here https://stately.ai/viz .

```
import { createMachine } from "xstate";

type GlobalContext = {
  currentTasks: [any];
  completedTasks: [any];
  currentTask?: any;
  navigationController: any;
};

type GlobalEvents =
  | { type: "SHOW_EDITOR" }
  | { type: "SAVE"; taskID?: string }
  | { type: "CANCEL" }
  | { type: "EDIT"; data: Partial<any> };

export const globalController = createMachine({
  schema: {
```

```
    context: {} as GlobalContext,
    events: {} as GlobalEvents,
  },
  predictableActionArguments: true,
  id: "globalController",
  initial: "idle",
  states: {
    idle: {
      on: {
        SHOW_EDITOR: {
          target: "editing",
          actions: "openTaskEditor",
        },
      },
    },
    editing: {
      on: {
        EDIT: {
          actions: "editTask",
        },
        SAVE: {
          actions: "saveNewTask",
          target: "idle",
        },
        CANCEL: {
          target: "idle",
          actions: "dismissTaskEditor",
        },
      },
    },
  },
});
```

## Summary:

- The tasks list displays all the tasks whether they're ongoing or completed.
- If you open the file oct0/src/screens/History.tsx, you will find out that it is currently not displaying the completed tasks. And the same goes for oct0/src/screens/Settings.tsx
- Pressing on a row in the list does nothing.
- There is no ability to check/uncheck a task.
- Styling is minimal and ugly.

## Instructions:

### Mandatory:

---

- Update TasksList to not show the completed tasks.
- Implement History to show the completed tasks.
- Add the ability to modify task details by pressing on the corresponding row. (see #Hints)
- Fix the TaskView component to have consistent spacing between the title and the toggle.

Optional:

---

**NOTE: The optional part doesn't restrict you to use xstate, however the mandatory part should not stay functional**

---

These are not in any particular order and certainly not exhaustive:

- Add some test coverage
- Add some settings to the app's settings screen, there is no restriction to what can this screen contains (eg. dark mode / profile ...)
- Add ability to delete tasks.
- Add ability to toggle task to completed/uncompleted.
- Add better styling/icons (No restrictions in using extra deps or styling lib)
- Add basic validation (eg. Title max characters 40 ...) (Error feedback ?) You are also free to add anything that you like whether it is theming or something else, just make sure to highlight either in the email, code comment etc...

## Hints

If you've never used xstate before, you don't need to dig deeper to complete this challenge. Let's take a look at how we create a new Task:

1. Inside ./oct0/src/screens/TaskEditor.tsx we send the `SAVE` event with taskID `undefined`

```
const handleSave = () => send({type: 'SAVE', taskID: undefined});
```

2. We handle that event in the action `saveNewTask` ./oct0/src/controllers/globalController.ts

```
  saveNewTask: assign((ctx, e) => {
    if (e.type !== 'SAVE') {
      return {};
    }
    // We don't have taskID parameter
    if (!e.taskID && ctx.currentTask) {
        // Create a new Task
      const newTask: Task = {
        id: (Math.random() * ctx.currentTask.title.length *
100).toString(),
        title: ctx.currentTask.title,
        description: ctx.currentTask.description,
        completed: false,
      };
      // Append the newly created task
      const updatedTasks = [...ctx.currentTasks, newTask];
      // Return the updated tasks as `currentTasks`
      // Reset the currentTask to allow for new input
      return {
```

```
          currentTasks: updatedTasks as [Task],
          currentTask: undefined,
        };
      }
      // Add code here...
      return {};
    }),
```

For the mandatory part, `saveNewTask` action should handle the case when taskID is defined. And instead of sending the event in this way:

```
const handleSave = () => send({type: 'SAVE', taskID: undefined});
```

we would instead do something like:

```
const handleEditedSave = () => send({type: 'SAVE', taskID: 1});
```

For the optional part you can refer to the xstate docs here: https://stately.ai/docs/xstate/basics/what-is-a-statechart