MOBILE PROGRAMMING

Tutorial 06

Activity 01: The Time Tracking App

Based on the lecture slides, build a Time Tracking App.

You need to build a **Time Tracking App** using React Native to manage time spent on tasks. This app will allow users to create timers for specific tasks, start/stop tracking time, and edit timer details.

Requirements

Main Interface:

- Display a list of active timers.
- A "+" button to add new timers.
- Each timer shows a title, project name, and elapsed time.

Core Features:

1. **Create Timer:** Users can create a new timer with a title and project name.
2. **Edit Timer:** Users can update the details of an existing timer.
3. **Start/Stop Timer:** Users can toggle the timer on or off.
4. **Delete Timer:** Users can remove a timer from the list.

Component Structure:

- **`EditableTimer`:** Displays a timer with editing capabilities.
- **`Timer`:** Shows timer details and control buttons.
- **`TimerButton`:** A reusable button component.
- **`TimerForm`:** A form component for entering timer details.

- **`ToggleableTimerForm`:** Displays the form when creating a new timer.

State Management:

- Uses **React Hooks** (`useState`, `useEffect`) to handle state.
- Each timer has the following states:
  - `id (unique identifier)`
  - `title (task name)`
  - `project (associated project)`
  - `elapsed (tracked time)`
  - `isRunning (active/inactive state)`

Time Formatting:

- Implement a millisecondsToHuman function to display time in **HH:MM:SS** format.

Technical Requirements:

- Use **React Native** to build the UI.
- Manage state with **React Hooks**.
- Ensure compatibility on both **iOS and Android**.

Optional Enhancements:

- Store timer data using **AsyncStorage**.
- Display a **time usage statistics chart** for each task.
- Support **task categories** for better organization.

Activity 02: Implement a Time Tracking App with Navigation in React Native

The requirement of this activity is to integrate a navigation system into the **Time Tracking App** to improve user experience. The **Time Tracking App** allows users to create, edit, and manage timers for their tasks. It uses **React Navigation** for screen transitions, and timers are managed using useState in App.js.

Detailed Requirements for Each Screen

1. App.js (Main Navigation)

➢ **Requirements:**

- Set up `NavigationContainer` to manage navigation within the app.
- Use `createNativeStackNavigator` to define screen transitions.
- Manage the list of timers using `useState` and pass them as props to child screens.
- Navigation should include:
  - `HomeScreen`: Displays the list of timers.
  - `CreateTimerScreen`: Allows users to create a new timer.
  - `EditTimerScreen`: Allows users to edit an existing timer (**should be added to `Stack.Navigator`**).

To Do:

Add `EditTimerScreen` to the navigation stack.

Use `setTimers` to update the timer list when creating or editing a timer.

---

2. HomeScreen.js (Timer List)

➢ **Requirements:**

- Display all `timers` in a list using `FlatList`.
- Each timer should have an **`Edit`** button that navigates to `EditTimerScreen`.
- Include an **"`Add Timer`"** button that navigates to `CreateTimerScreen`.

To Do:

Use `FlatList` to render timers.

Pass the selected timer when navigating to `EditTimerScreen`.

3. CreateTimerScreen.js (Create Timer)

- **Requirements:**

- Allow users to enter **`Title`** and **`Project`** for a timer.
- When **`Save`** is clicked, create a new timer with:
  - `id`: A unique identifier using `Date.now().toString().`
  - `title`: The user-provided title.
  - `project`: The user-provided project name.
  - `elapsed`: Default value 0.
  - `isRunning`: Default value false.
- After saving, return to `HomeScreen` and update the timer list.

To Do:

Add validation: Prevent saving if the `Title` field is empty.

Update the timer list using `setTimers([...timers, newTimer]).`

Navigate back to `HomeScreen` after saving.

---

4. EditTimerScreen.js (Edit Timer)

Requirements:

- Receive timer data from `route.params` to populate the input fields.
- Allow users to modify `title` and project.
- When **Save Changes** is clicked, update the timer in the list.
- After saving, return to `HomeScreen` and refresh the list.

To Do:

Use `setTimers` to update the selected timer in the list.

Navigate back to `HomeScreen` after editing.

Submission

Submit a zip file of your project (excluding `node_modules` folder) to this tutorial's submission box in the course website on FIT Portal.