Timer Management App – Assignment Solution

Objective

The goal of this assignment is to build a React Native app that allows users to create, manage, and interact with multiple customizable timers.

Each timer is associated with a category, and users can perform grouped actions such as starting, pausing, or resetting all timers under a category.

Data should be persisted locally using AsyncStorage to ensure users do not lose timers on app restart.

UI/UX should be clean and intuitive with minimal third-party dependencies.

Requirements

1. Core Features
   1. Add Timer

Functionality: Users can create a new timer with the following fields:

Name: Name of the timer.

Duration: Duration in seconds.

Category: A category assigned to the timer.

Implementation:

A modal form is used for timer creation.

On saving, timer is added to the list and persisted in AsyncStorage.

Code Example:

<TextInput placeholder=”Timer Name” value={name} onChangeText={setName} />

<TextInput placeholder=”Duration (seconds)” value={duration} onChangeText={setDuration} keyboardType=”numeric” />

<TextInput placeholder=”Category” value={category} onChangeText={setCategory} />

<Button title=”Save Timer” onPress={addTimer} />

* 1. Timer List with Grouping

Timers are displayed under expandable/collapsible sections based on their category.

For each timer:

Name, Remaining time, and Status are shown.

Code Example:

{Object.keys(groupedTimers).map(category => (

<CategorySection

Category={category}

Timers={groupedTimers[category]}

...

>

{groupedTimers[category].map(timer => (

<TimerItem key={timer.id} timer={timer} ... />

))}

</CategorySection>

))}

1.3 Timer Management

Start, Pause, and Reset buttons for each timer.

Timers marked as “Completed” when remaining time reaches 0.

* 1. Progress Visualization

A progress bar showing remaining time relative to total duration.

Code Example:

<ProgressBarAndroid styleAttr=”Horizontal” progress={1 – timer.remaining / timer.duration} indeterminate={false} />

1.5 Bulk Actions

“Start All”, “Pause All”, and “Reset All” buttons at the category level.

* 1. User Feedback

On timer completion, an Alert modal congratulates the user with the timer’s name.

1. Enhanced Functionality
   1. Timer History

Each completed timer is logged with:

Name

Completion time

History is displayed on a separate “History” screen.

* 1. Customizable Alerts

Users can optionally set a halfway alert (50% time remaining).

An Alert is triggered at halfway point.

Technical Details

State Management: useState and useReducer

Navigation: @react-navigation/native and @react-navigation/stack

Persistence: @react-native-async-storage/async-storage

Timers: Managed using setInterval

Styling: Handled using StyleSheet

Bonus: Light/Dark mode theming optional

Bonus Features (Optional)

Export completed timer history as a JSON file.

Light and Dark mode support via a theme switcher.

Filter timers based on categories.

Deliverables

A fully functional React Native app with all above features.

GitHub repository containing:

All project code.

README with setup instructions.

Estimated Completion Time

5–6 hours.

Code Structure

/TimerApp

/components

TimerItem.js

CategorySection.js

/screens

HomeScreen.js

HistoryScreen.js

/storage

AsyncStorageHelper.js

/navigation

AppNavigator.js

App.js

Important Code Files

App.js

Import React from ‘react’;

Import { NavigationContainer } from ‘@react-navigation/native’;

Import AppNavigator from ‘./navigation/AppNavigator’;

Export default function App() {

Return (

<NavigationContainer>

<AppNavigator />

</NavigationContainer>

);

}

Navigation/AppNavigator.js

Import React from ‘react’;

Import { createStackNavigator } from ‘@react-navigation/stack’;

Import HomeScreen from ‘../screens/HomeScreen’;

Import HistoryScreen from ‘../screens/HistoryScreen’;

Const Stack = createStackNavigator();

Export default function AppNavigator() {

Return (

<Stack.Navigator>

<Stack.Screen name=”Home” component={HomeScreen} />

<Stack.Screen name=”History” component={HistoryScreen} />

</Stack.Navigator>

);

}

Storage/AsyncStorageHelper.js

Import AsyncStorage from ‘@react-native-async-storage/async-storage’;

Export const storeData = async (key, value) => {

Try {

Await AsyncStorage.setItem(key, JSON.stringify(value));

} catch (e) {

Console.error(‘Error storing data’, e);

}

};

Export const getData = async (key) => {

Try {

Const value = await AsyncStorage.getItem(key);

Return value ? JSON.parse(value) : null;

} catch (e) {

Console.error(‘Error fetching data’, e);

}

};

Components/TimerItem.js

Import React from ‘react’;

Import { View, Text, Button, StyleSheet, ProgressBarAndroid } from ‘react-native’;

Const TimerItem = ({ timer, onStart, onPause, onReset }) => {

Const progress = 1 – timer.remaining / timer.duration;

Return (

<View style={styles.container}>

<Text style={styles.name}>{timer.name}</Text>

<ProgressBarAndroid styleAttr=”Horizontal” progress={progress} />

<Text>Time Left: {timer.remaining} sec</Text>

<View style={styles.buttons}>

<Button title=”Start” onPress={onStart} />

<Button title=”Pause” onPress={onPause} />

<Button title=”Reset” onPress={onReset} />

</View>

</View>

);

};

Const styles = StyleSheet.create({

Container: { padding: 10, marginBottom: 10, backgroundColor: ‘#f1f1f1’, borderRadius: 8 },

Name: { fontSize: 18, fontWeight: ‘bold’ },

Buttons: { flexDirection: ‘row’, justifyContent: ‘space-between’, marginTop: 10 },

});

Export default TimerItem;