Objective

Build a React Native app that allows users to create, manage, and interact with multiple customizable timers. The app will include features like categories, progress visualization, and grouped actions while maintaining clean UI/UX and minimal third-party dependencies.

Requirements

1. Core Features

1. **Add Timer**
    - A screen to create new timers with the following fields:
        - **Name**: The name of the timer (e.g., "Workout Timer").
        - **Duration**: Timer duration in seconds.
        - **Category**: Assign a category to the timer (e.g., "Workout," "Study," "Break").
    - Save the timer to a list and persist the data locally (e.g., using `AsyncStorage`).
2. **Timer List with Grouping**
    - Display all timers grouped by their categories in expandable/collapsible sections.
    - For each timer, show:
        - Name.
        - Remaining time.
        - Status: Running, Paused, or Completed.
    - Users can expand or collapse categories to view timers within them.
3. **Timer Management**
    - Provide controls for each timer to:
        - **Start**: Begin countdown.
        - **Pause**: Pause countdown.
        - **Reset**: Reset to original duration.
    - Mark timers as "Completed" when they reach zero.
4. **Progress Visualization**
    - Show a simple progress bar or percentage for each timer to visualize remaining time relative to the total duration.
5. **Bulk Actions**
    - Add buttons at the category level to:
        - Start all timers in a category.
        - Pause all timers in a category.
        - Reset all timers in a category.
6. **User Feedback**
    - When a timer completes:
        - Show an on-screen modal with a congratulatory message and the timer's name.

Enhanced Functionality

1. **Timer History**
    - Maintain a log of completed timers with:
        - Timer name.
        - Completion time.
    - Display the log on a separate "History" screen.
2. **Customizable Alerts**
    - Allow users to set an optional halfway alert for each timer (e.g., at 50% of the total duration).
    - Display a notification or message when the alert triggers.

Technical Details

- **State Management**: Use `useState` or `useReducer` for managing timers and categories.
- **Navigation**: Use `React Navigation` with at least two screens:
    1. **Home Screen**: Timer list and grouping functionality.
    2. **History Screen**: Log of completed timers.
- **Persistence**: Use `AsyncStorage` for storing timers and logs.
- **Styling**: Use `StyleSheet` for clean and responsive layouts.
- **Timers**: Implement time management logic using `setInterval`.

Bonus Features (Optional)

1. **Export Timer Data**

    - Allow users to export timer history as a JSON file.
2. **Custom Themes**
    - Add support for light and dark modes with a theme switcher.
3. **Category Filtering**
    - Add a filter dropdown to show timers from specific categories only.

Deliverables

1. A React Native app with:
    - Timer creation and management.
    - Grouping and bulk actions.
    - Progress visualization.
    - History tracking.
2. Code hosted in a GitHub repository with:
    - A `README.md` containing:
        - Setup instructions.
        - A list of assumptions made during development.

Estimated Completion Time

5–6 hours

This enhanced assignment evaluates advanced React Native skills, including **state management, user interaction, data persistence, and UI/UX design**. It's detailed enough to showcase a mid-level developer's capabilities while remaining manageable.