M. Rifki Paranrengi

Advanced State Management

In this assignment I developed an enhanced counter application that demonstrates both local and global state management techniques in Flutter. The project evolved from a basic counter to a sophisticated multi-counter system with advanced UI features.

The approaches implemented are:

1. **Local State Management:** Utilizing StatefulWidget for component-specific state like animations and UI transitions.
2. **Global State Management:** Creating a custom state management solution using ChangeNotifier and InheritedWidget patterns.

Part 1: Local State Management
I implemented local state management within the CounterTile widget using StatefulWidget. This handles individual counter animations and maintains the decrement button's disabled state when counter values reach zero. The AnimatedScale widget provides visual feedback by scaling the decrement button based on counter value state.

Part 2: Global State Management
I built a separate Flutter package named 'global_state' to handle shared application state. The main application imports this package and uses it to manage multiple counters simultaneously. The architecture consists of MyEphemeralApp as the root widget, which creates a GlobalState instance and wraps the entire app with GlobalStateProvider.

The Counter model contains four properties:
1. id: Unique identifier generated using DateTime.now().microsecondsSinceEpoch
2. value: Integer counter that can be incremented or decremented
3. label: String name for each counter (defaults to 'Counter')
4. color: Automatically assigned from kCounterPalette based on ID hash

The GlobalState class manages all counter operations including addCounter(), increment(), decrement(), updateCounter(), removeCounterById(), and moveCounter() for reordering functionality.

Main challenges I encountered were implementing the custom InheritedWidget wrapper that properly manages listener subscriptions and widget lifecycle. The GlobalStateProvider handles adding and removing listeners in initState() and dispose() methods, while the internal _Inherited widget always returns true in updateShouldNotify() since the GlobalState mutates in place.

Advanced features implemented in my application:

1. Dynamic Counter Management: Users can add unlimited counters using addCounter() with unique timestamp IDs. The 'Remove Last' button demonstrates selective counter removal using removeCounterById().

2. Interactive Color Customization: Each counter displays with colors from an 8-color Material Design palette defined in kCounterPalette. The edit dialog shows color selection as circular containers with tap gestures to change counter themes.

3. Label Editing System: Implemented _EditCounterDialog as a modal that allows users to modify counter names using TextField input. The dialog returns a Map containing updated label and color values.

4. Drag-and-Drop Reordering: Built using ReorderableListView.builder with buildDefaultDragHandles set to false. Custom ReorderableDragStartListener widgets provide individual drag handles, and the moveCounter() method handles index calculations for proper list reordering.