Introduction

In this assignment, I built a Global Counter App in Flutter. Before this, I only worked with local state using setState(). For this task, I learned how to use a global state management approach so that multiple counters could share and update data together.

How I Managed State

I created a GlobalState class that keeps track of a list of counters. Each counter has its own id, label, color, and value.

To manage global state, I used the Scoped Model package. This let me pass data down the widget tree and automatically rebuild only the parts of the app that needed to change. Compared to just using setState() everywhere, this approach made the code much cleaner and easier to maintain.

Features I Added

Here are the main features I included in the app:

1. Multiple counters – instead of just one counter, the app supports adding as many counters as the user wants.

2. Add and remove counters – users can create new counters through a dialog, and also delete existing ones.

3. Increment and decrement – each counter has its own buttons to increase or decrease the value.

4. Drag-and-drop reordering – counters can be rearranged with drag-and-drop, thanks to ReorderableListView.

5. Colors and animations – each counter has a unique color, and I added small animations when the value changes to make the UI feel smoother.

What I Learned

At first, I found it tricky to understand the difference between local state and global state. But after implementing the GlobalState class, I started to see how important it is when working with apps that have multiple dynamic parts.

The hardest part was getting drag-and-drop to work while keeping the counters' data in sync with the UI. Once I figured it out, it really showed me how powerful Flutter's widget system can be.

Conclusion

Overall, this assignment helped me learn how to manage global state in Flutter and why it's better than only relying on local state for bigger apps. I also practiced adding useful features like multiple counters, custom colors, animations, and reordering.

This was a big step forward compared to the simple counter app I made before. Now I feel more confident about handling more complex apps that need shared state across different widgets.