

FLUTTER PROVIDER

Aurelia
D.P. Nata





global state.

1 **increment & decrement**

3 **advanced UI**

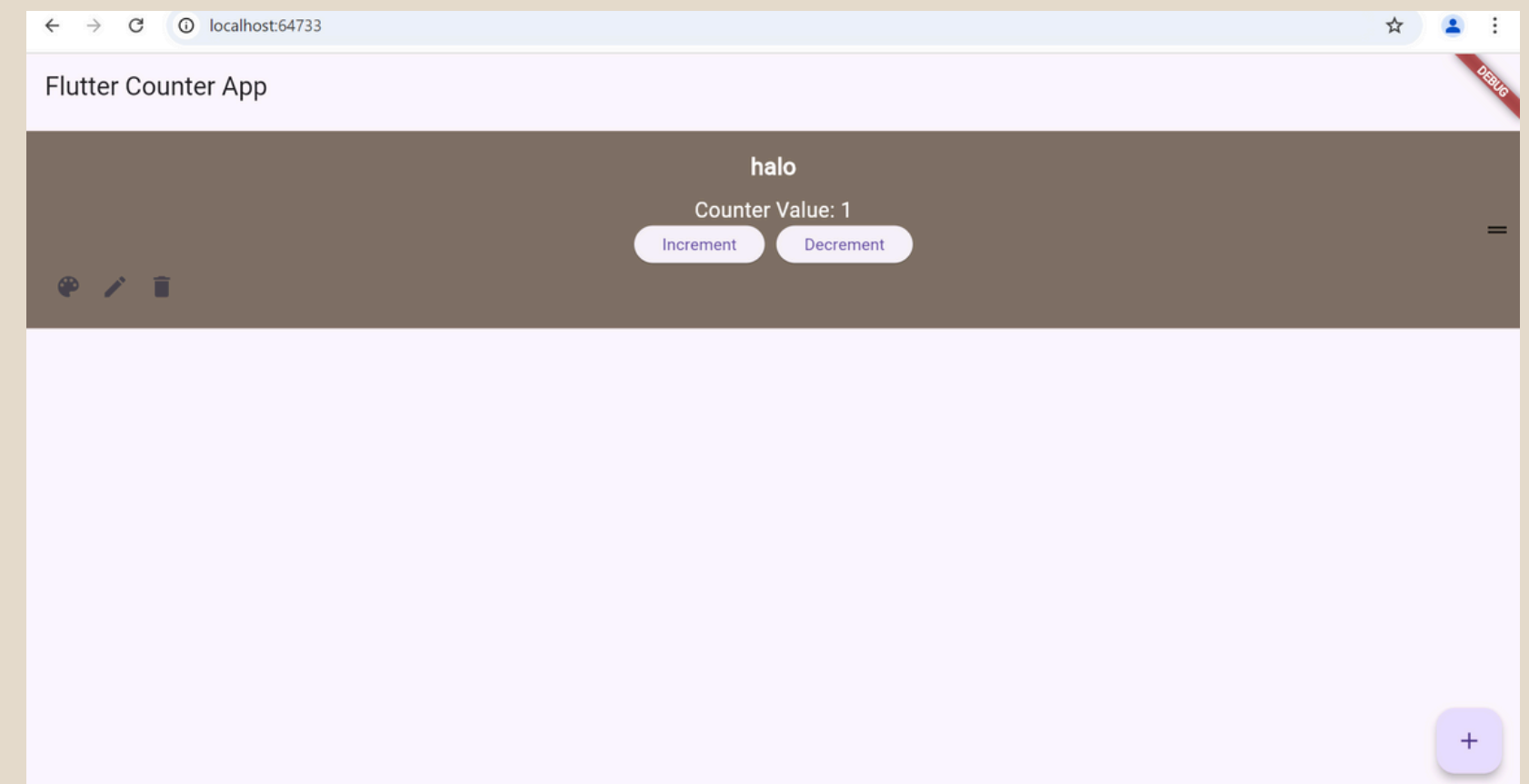
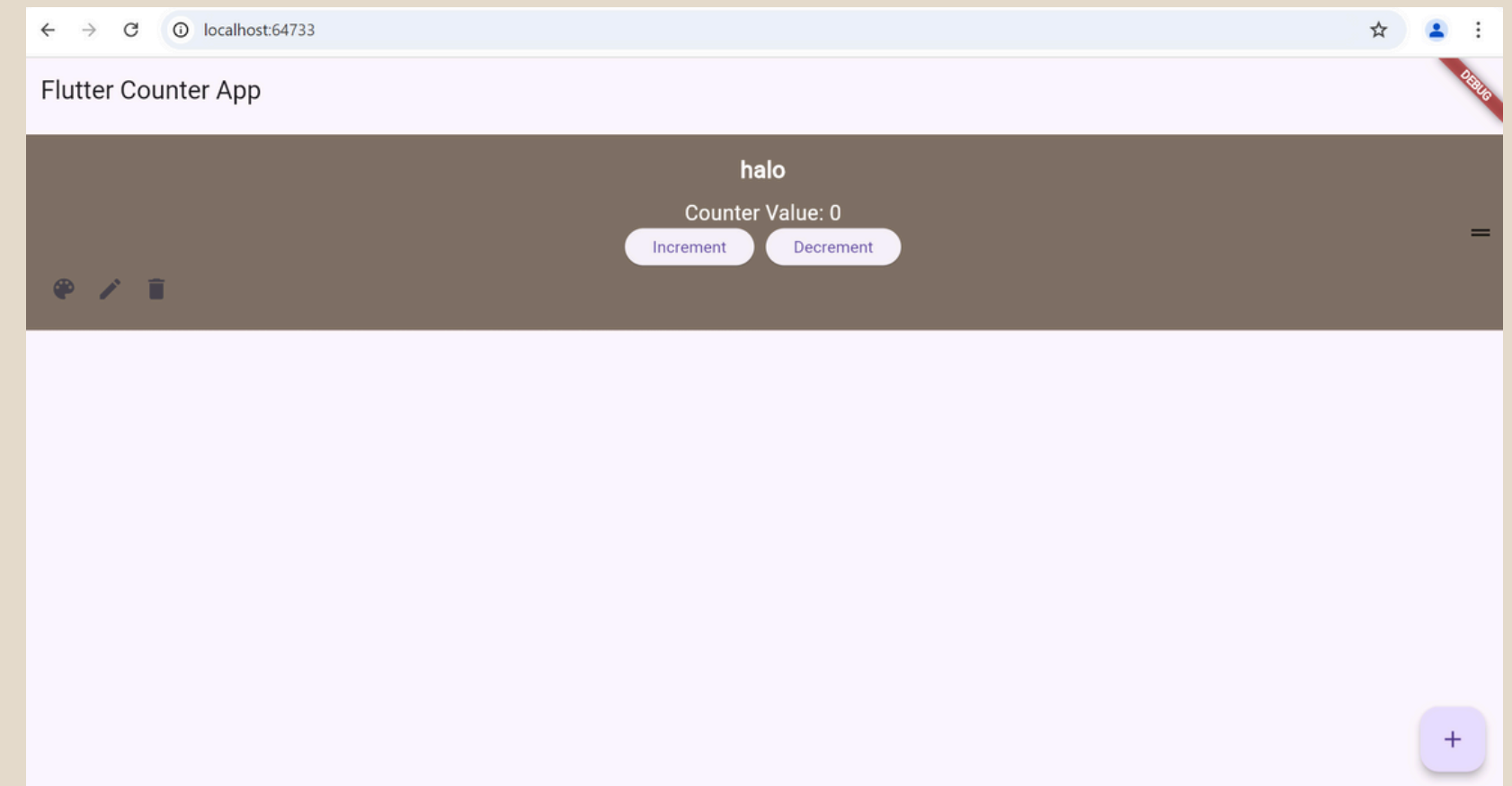
2 **functionality**



Implement Increment and Decrement

Fitur ini memungkinkan pengguna untuk menambah (increment) atau mengurangi (decrement) nilai counter secara individual. Tombol Increment menambahkan 1 ke nilai counter, sedangkan tombol Decrement mengurangi nilai counter (dengan batas nilai minimum 0).

jika dilihat dari hasil dambar dibawah ketika memencet tombol increment nilai yang awalnya 0 akan menjadi 1 dan ketika decrement yang ditekan maka nilai akan kembali menjadi 0. Jika nilai 0 dan decrement yang ditekan, nilai akan tetap 0



Penjelasan kode

```
ElevatedButton(  
  onPressed: onIncrement,  
  child: Text('Increment'),  
), // ElevatedButton  
 SizedBox(width: 10),  
 ElevatedButton(  
  onPressed: onDecrement,  
  child: Text('Decrement'),  
), // ElevatedButton
```

- Pada CounterWidget, dua tombol dibuat menggunakan ElevatedButton. Tombol ini memanggil onIncrement dan onDecrement saat ditekan.

```
onIncrement: () {  
  globalState.changeCounter(index, counterData['value'] + 1);  
},  
onDecrement: () {  
  if (counterData['value'] > 0) {  
    globalState.changeCounter(index, counterData['value'] - 1);  
  }  
},
```

- Fungsi onIncrement dan onDecrement diimplementasikan di Main.dart. Fungsi ini mengakses state global dan memperbarui nilai counter pada indeks tertentu.


```
void changeCounter(int index, int value) {  
  _counters[index]['value'] = value;  
  notifyListeners();  
}
```

- Pada global_state.dart, fungsi changeCounter memperbarui nilai counter di daftar _counters dan memanggil notifyListeners() untuk memperbarui tampilan.



advanced UI

Fitur-fitur ini meningkatkan pengalaman pengguna dengan menambahkan elemen advanced UI dan interaksi yang menarik:

- **Warna Unik untuk Setiap Counter:** Setiap counter memiliki warna yang dapat diubah oleh pengguna.
 - **Mengubah Warna dan Label:** Ikon warna membuka dialog pemilih warna, sedangkan ikon edit membuka dialog untuk mengganti label.
 - **Animasi pada Increment/Decrement:** Efek animasi memberikan pengalaman visual yang lebih menarik.
 - **Drag-and-Drop untuk Reorder:** Fitur ini memungkinkan pengguna untuk mengubah urutan counter melalui interaksi seret-lepas.
- 

Counter App

halo

Counter Value: 0

Increment

Decrement

Counter 2

Counter Value: 0

Increment

Decrement

Counter 3

Counter Value: 0

Increment

Decrement

Flutter Counter App

halo

Counter Value: 0

Increment

Decrement

Counter 3

Counter Value: 0

Increment

Decrement

Flutter Counter App

Counter 2

Counter Value: 0

Increment

Decrement

halo

Counter Value: 0

Increment

Decrement

Counter 3

Counter Value: 0

Increment

Decrement

Flutter Counter App

box 2

Counter Value: 0

Increment

Decrement

halo

Counter Value: 0

Increment

Decrement

Penjelasan kode : warna pada counter

```
IconButton(  
  icon: Icon(Icons.color_lens),  
  onPressed: () async {  
    Color? newColor = await _pickColor(context);  
    if (newColor != null) {  
      onColorChanged(newColor);  
    }  
  },  
)
```

- Mengubah Warna: Ikon palet warna membuka dialog pemilih warna menggunakan ColorPicker dari paket flutter_colorpicker. package color[icker ditambahkan di pubspec.yaml (flutter_colorpicker: ^1.0.0)

```
void changeColor(int index, Color newColor) {  
  _counters[index]['color'] = newColor;  
  notifyListeners();  
}
```

- Fungsi changeColor bertanggung jawab untuk mengubah warna (color) dari salah satu counter dalam daftar _counters. Fungsi ini memperbarui data dan memberi tahu seluruh aplikasi bahwa data telah berubah, sehingga UI dapat diperbarui secara otomatis.

```
return AnimatedContainer(  
  key: key,  
  duration: Duration(milliseconds: 300),  
  color: color,  
  padding: EdgeInsets.all(16.0),  
  margin: EdgeInsets.symmetric(vertical: 10.0),  
  child: Column(  
    children: [
```

- Animasi: Setiap counter ditampilkan menggunakan AnimatedContainer. Komponen ini memberikan animasi transisi saat ada perubahan pada atributnya seperti warna atau padding.

Penjelasan kode : warna pada counter

```
Future<Color?> _pickColor(BuildContext context) async {
  Color tempColor = color;
  return await showDialog<Color>(
    context: context,
    builder: (context) {
      return AlertDialog(
        title: Text('Pick a color'),
        content: SingleChildScrollView(
          child: ColorPicker(
            pickerColor: color,
            onColorChanged: (selectedColor) {
              tempColor = selectedColor;
            },
          ), // ColorPicker
        ), // SingleChildScrollView
        actions: <Widget>[
          TextButton(
            onPressed: () {
              Navigator.pop(context, tempColor);
            },
            child: Text('Select'),
          ), // TextButton
        ], // <Widget>[]
      ); // AlertDialog
    },
  );
}
```

- Fungsi `_pickColor` adalah fungsi asinkron yang menampilkan dialog kepada pengguna untuk memilih warna menggunakan widget `ColorPicker`. Fungsi ini menerima konteks aplikasi (`BuildContext`) sebagai parameter untuk menampilkan dialog dalam hierarki UI.
- Awalnya, warna yang sedang digunakan disimpan dalam variabel sementara `tempColor`. Fungsi ini menggunakan `showDialog<Color>` untuk memunculkan dialog dengan tipe pengembalian `Color`. Dalam dialog tersebut, pengguna dapat memilih warna baru melalui widget `ColorPicker`, yang memperbarui nilai `tempColor` setiap kali warna dipilih. Setelah pengguna selesai memilih dan menekan tombol "Select", dialog ditutup dengan `Navigator.pop`, dan warna yang disimpan dalam `tempColor` dikembalikan.
- Jika dialog ditutup tanpa memilih warna, fungsi akan mengembalikan nilai `null`. Dialog ini memungkinkan pengguna untuk berinteraksi secara langsung dengan antarmuka dan mengubah properti warna pada widget.

Penjelasan kode : mengubah label

```
IconButton(  
  icon: Icon(Icons.edit),  
  onPressed: () async {  
    String? newLabel = await _editLabel(context);  
    if (newLabel != null) {  
      onLabelChanged(newLabel);  
    }  
  },  
) // IconButton
```

- Fungsi `_editLabel` menampilkan dialog dengan sebuah `TextField` di dalamnya, yang memungkinkan pengguna mengetik label baru. Dialog ini dibangun menggunakan `AlertDialog` dan menerima input teks melalui `TextEditingController` yang mengontrol nilai teks pada `TextField`. Ketika pengguna menekan tombol "Save", dialog akan menutup dan mengembalikan nilai yang dimasukkan oleh pengguna (`controller.text`). Nilai ini kemudian dikirimkan kembali ke widget pemanggil untuk digunakan sebagai label baru. Jika pengguna tidak mengubah atau membatalkan input, fungsi ini akan mengembalikan `null`.

```
Future<String?> _editLabel(BuildContext context) async {  
  TextEditingController controller = TextEditingController();  
  return await showDialog<String>(  
    context: context,  
    builder: (context) {  
      return AlertDialog(  
        title: Text('Edit Label'),  
        content: TextField(  
          controller: controller,  
          decoration: InputDecoration(hintText: 'Enter new label'),  
        ), // TextField  
        actions: <Widget>[  
          TextButton(  
            onPressed: () {  
              Navigator.pop(context, controller.text);  
            },  
            child: Text('Save'),  
          ), // TextButton  
        ], // <Widget>[]  
      ); // AlertDialog  
    },  
  );  
}
```

- Kode pertama menunjukkan penggunaan `IconButton` untuk memungkinkan pengguna mengedit label dari sebuah elemen. Ketika tombol ditekan, fungsi `onPressed` memanggil fungsi asinkron `_editLabel`, yang menampilkan dialog untuk meminta pengguna memasukkan label baru. Fungsi ini menggunakan `await` untuk menunggu hasil dari dialog, yang mengembalikan string berisi label baru jika pengguna mengetikkan dan menyimpan label tersebut. Setelah itu, jika label yang baru tidak `null`, fungsi `onLabelChanged` dipanggil untuk memperbarui label pada elemen.

Penjelasan kode : drag and drop

```
void reorderCounters(int oldIndex, int newIndex) {  
    if (oldIndex < newIndex) newIndex -= 1;  
    final counter = _counters.removeAt(oldIndex);  
    _counters.insert(newIndex, counter);  
    notifyListeners();  
}
```

```
body: ReorderableListView(  
  onReorder: (oldIndex, newIndex) {  
    globalState.reorderCounters(oldIndex, newIndex);  
  },  
  children: globalState.counters  
    .asMap()  
    .map((index, counterData) {  
      return MapEntry(  
        index,  
        CounterWidget(  
          key: ValueKey(index),  
          label: counterData['label'],  
          color: counterData['color'],  
          counter: counterData['value'],  
          onIncrement: () {  
            globalState.changeCounter(index, counterData['value'] + 1);  
          },  
          onDecrement: () {  
            if (counterData['value'] > 0) {  
              globalState.changeCounter(index, counterData['value'] - 1);  
            }  
          },  
          onLabelChanged: (newLabel) {  
            globalState.changeLabel(index, newLabel);  
          },  
          onColorChanged: (newColor) {  
            globalState.changeColor(index, newColor);  
          },  
          onDelete: () {  
            globalState.removeCounter(index);  
          },  
        ), // CounterWidget  
      ); // MapEntry  
    })  
    .values  
    .toList(),  
), // ReorderableListView
```

- fungsi `reorderCounters` bertugas untuk mengubah posisi elemen dalam daftar `_counters`. Ketika posisi counter diubah (misalnya dengan drag-and-drop), parameter `oldIndex` dan `newIndex` diterima untuk menunjukkan posisi lama dan baru dari elemen tersebut. Jika `oldIndex` lebih kecil dari `newIndex`, indeks baru dikurangi 1 untuk menyesuaikan perubahan posisi. Kemudian, elemen dihapus dari indeks lama dan disisipkan pada indeks baru. Setelah perubahan, `notifyListeners()` dipanggil untuk memberi tahu pendengar agar memperbarui tampilan.
- `onReorder` memanggil `reorderCounters` saat pengguna melakukan drag-and-drop untuk mengubah urutan elemen dalam daftar. Di bagian `children`, setiap item dalam `globalState.counters` diubah menjadi widget `CounterWidget` menggunakan `asMap()` untuk mendapatkan indeks dan data. Setiap `CounterWidget` diberi kemampuan untuk meningkatkan atau mengurangi nilai counter, mengubah label dan warna, serta menghapus elemen melalui berbagai callback seperti `onIncrement`, `onDecrement`, `onLabelChanged`, `onColorChanged`, dan `onDelete`. Semua perubahan ini diatur melalui state global menggunakan metode yang sesuai di dalam `globalState`.

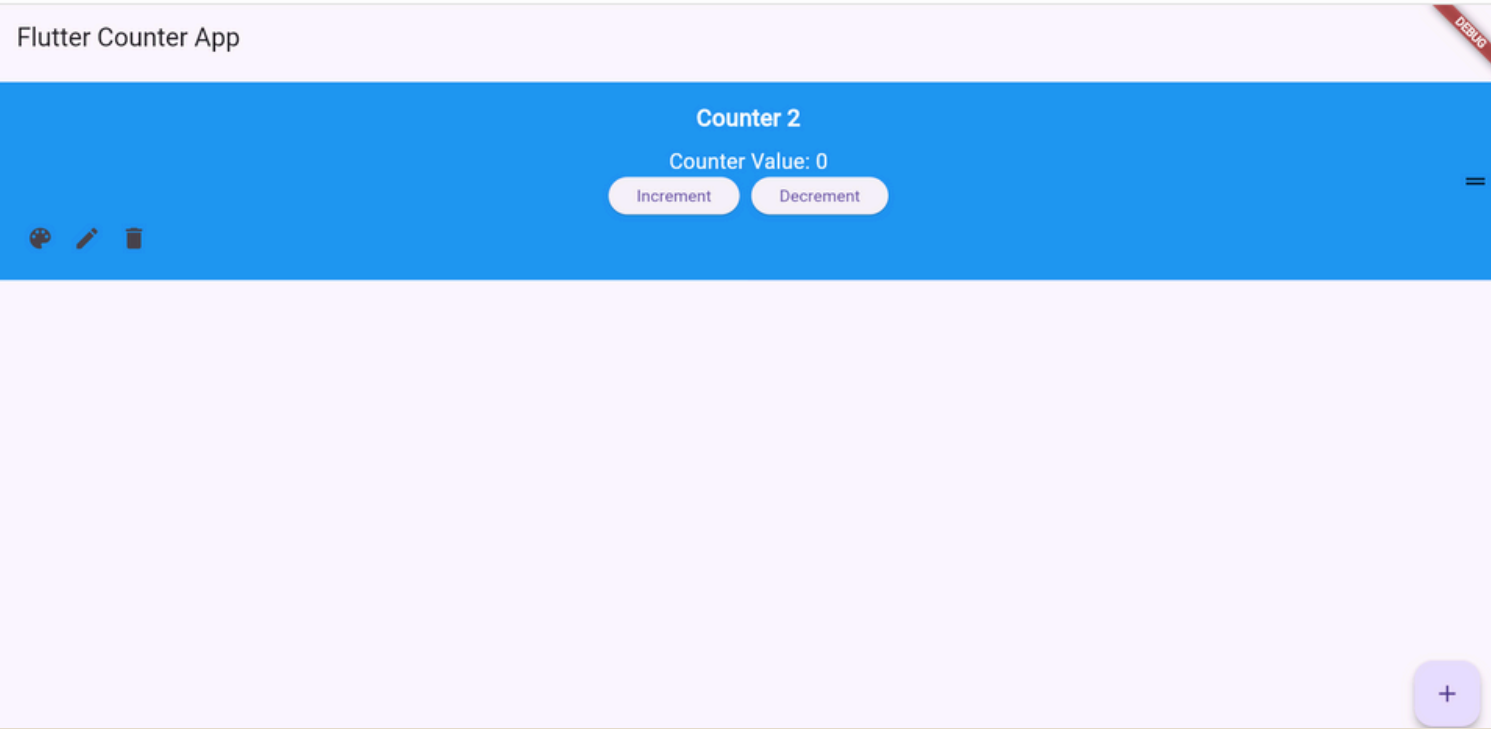
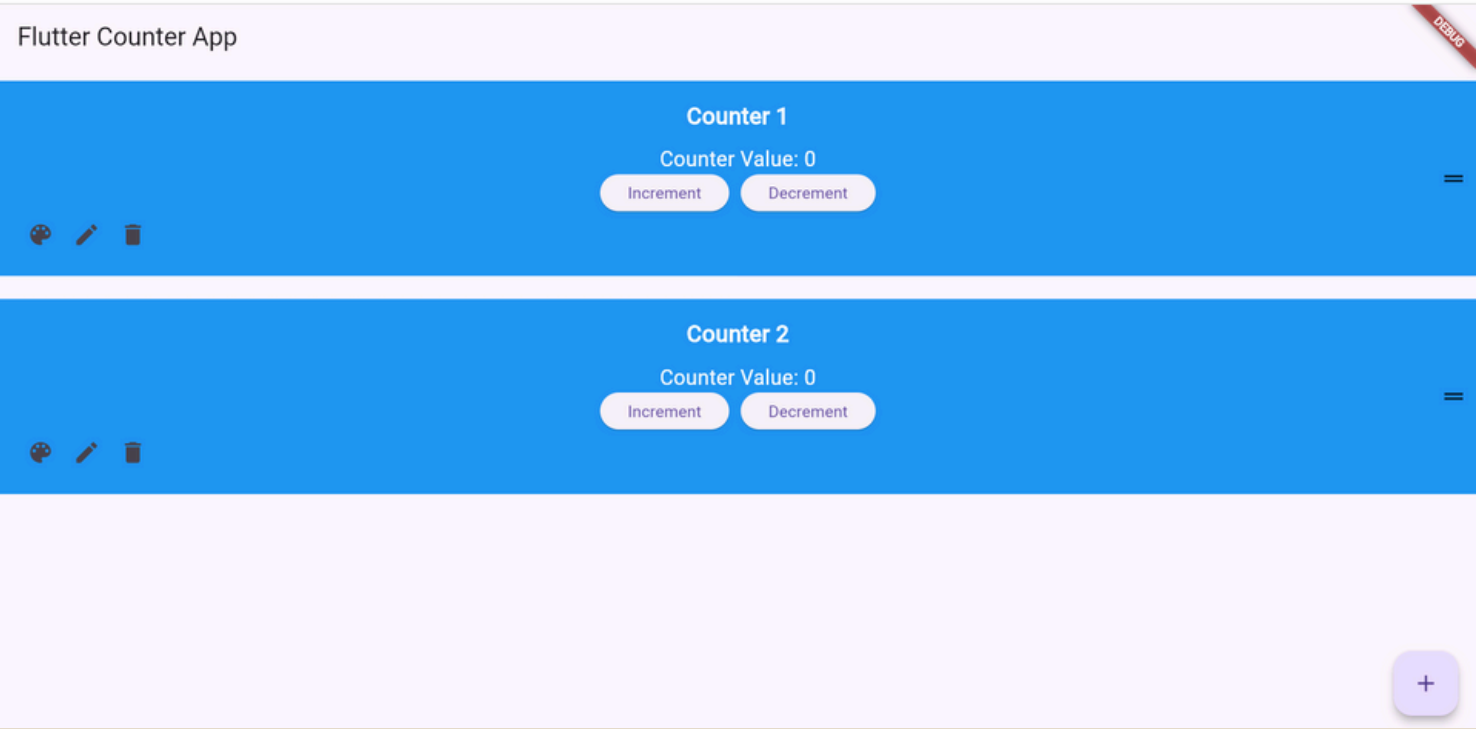


functionality

Semua data counter (label, warna, nilai) dikelola di dalam sebuah global state menggunakan Provider. Fitur yang didukung:

- **Add New Counters:** Tambahkan counter baru dengan atribut default.
- **Remove Counters:** Hapus counter berdasarkan indeksnya.





Penjelasan kode: add & remove

```
void addCounter() {  
    _counters.add({  
        'label': 'Counter ${_counters.length + 1}',  
        'value': 0,  
        'color': Colors.blue,  
    });  
    notifyListeners();  
}
```

- Fungsi `addCounter()` bertanggung jawab untuk menambahkan counter baru ke dalam daftar `_counters`. Setiap counter yang ditambahkan memiliki label default yang dihitung berdasarkan jumlah counter yang sudah ada ('Counter \${_counters.length + 1}'), nilai awal counter diatur menjadi 0 ('value': 0), dan warna default diatur ke biru ('color': `Colors.blue`). Setelah menambahkan elemen baru, fungsi `notifyListeners()` dipanggil untuk memberi tahu komponen yang mendengarkan perubahan state agar memperbarui UI dengan daftar counter yang baru.

```
void removeCounter(int index) {  
    _counters.removeAt(index);  
    notifyListeners();  
}
```

- fungsi `removeCounter(int index)` digunakan untuk menghapus counter berdasarkan indeks yang diberikan. Fungsi ini memanggil `removeAt(index)` untuk menghapus elemen pada posisi yang sesuai dalam daftar `_counters`. Setelah menghapus elemen, `notifyListeners()` dipanggil untuk memberi tahu komponen yang mendengarkan perubahan agar memperbarui UI, mencerminkan daftar counter yang telah diperbarui tanpa elemen yang dihapus.

THANK
YOU.

Brigitte
Schwartz

