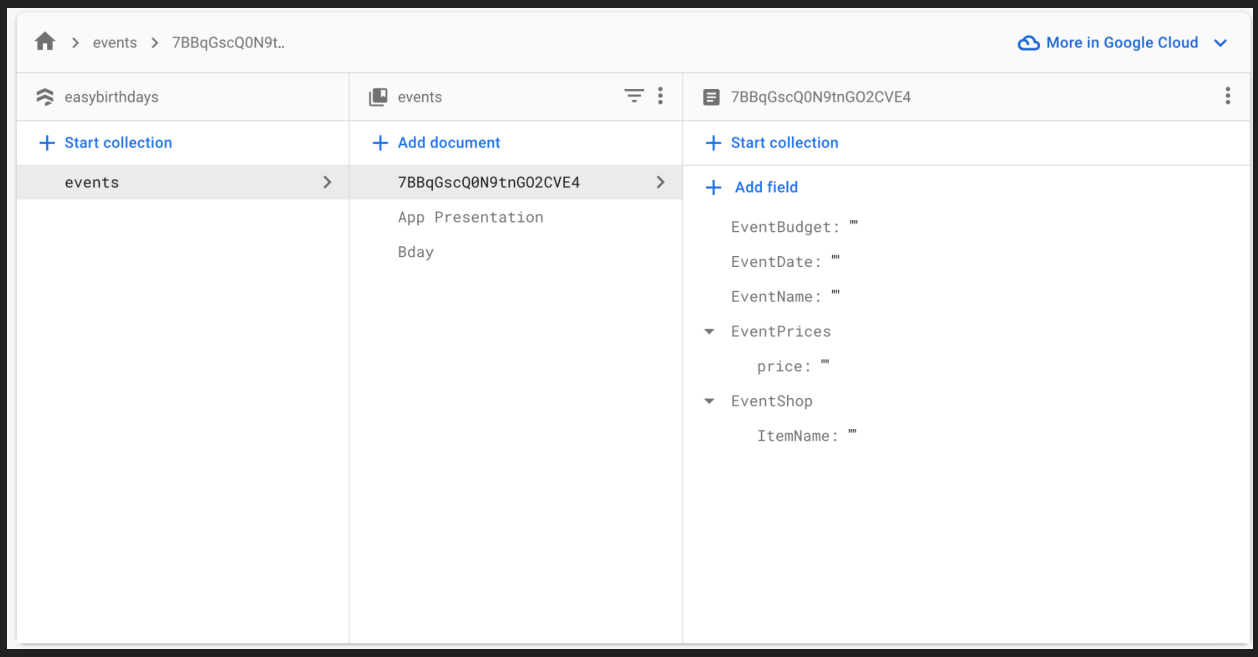
Easy Birthdays

Event Tracker

Firebase Setup



Events stored in Firebase

- strings and map<string>
- map<string, number>
- budgetPrices number
- DateTime

Reading Data

TextFormField
string data read normally
maps…

```
Map<String, String> eventShop = _eventShopController.text
        .split(',')
        .asMap()
        .map((index, value) => MapEntry('Item ${index + 1}', value.trim()))
        .cast<String, String>();
```

Events Listing

- store events in “Card” widget
- listview() for overflow, allows scrolling

```
return ListView.builder(
  itemCount: snapshot.data!.docs.length,
  itemBuilder: (context, index) {
```

list view contains Card children

```
return Card(
  elevation: 4,
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(15),
  ), // RoundedRectangleBorder
  child: ListTile(
```

Card contains ListTile child

```
child: ListTile(
  title: Text('${doc["EventName"]}'),
  subtitle: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text('${doc["EventDate"]}'),
```

Trailing row for edit and delete buttons

```
trailing: Row(
  mainAxisSize: MainAxisSize.min,
  children: [
    ElevatedButton(
      onPressed: () {
        showDialog(
          context: context,
          builder: (BuildContext context) {
            return AlertDialog(
              title:
                  const Text('Confirm Delete'),
              content: const Text(
                  'Are you sure you want to delete this event?'), // Text
              actions: <Widget>[
                TextButton(
                  child: const Text('No'),
                  onPressed: () {
                    Navigator.of(context).pop();
                  },
                ), // TextButton
                TextButton(
                  child: const Text('Yes'),
                  onPressed: () {
                    FirebaseFirestore.instance
                        .collection('events')
                        .doc(
                            '${doc["EventName"]}')
                        .delete();
                    Navigator.of(context).pop();
```

Base event displayed

Don't output if id == base id

```
if (doc.id == '7BBqGscQ0N9tnGO2CVE4') {
  return SizedBox.shrink(); // Return a blank widget
```

Displaying map in for loop

Total Budget Variable

```
for (final entry
    in doc['EventShop'].entries)
  if (entry.value != 'NA')
    (Text('${entry.value}')),

for (final entry
    in doc['EventPrices'].entries)
  if (entry.value != 'NA')
    (Text('\$${entry.value}')),

if (doc["EventBudget"] != 'NA')
  Text(
      'Total Budget: \$${doc["EventBudget"]}'), // Text
```

Providers

```
import 'package:provider/provider.dart';
```

Problems updating color

Providers in Flutter act as a shared container of data that can be accessed and modified by different parts of the application, making it easier to manage and update the state of the user interface.

Provider class sets data

Notify listeners

```
import 'package:flutter/material.dart';

class ColorProvider with ChangeNotifier {
  Color _colorSetting = Colors.blue;

  Color get colorSetting => _colorSetting;

  set colorSetting(Color newColor) {
    _colorSetting = newColor;
    notifyListeners();
  }
}

class PriceProvider with ChangeNotifier {
  bool _priceSetting = true;

  bool get priceSetting => _priceSetting;

  set priceSetting(bool value) {
    _priceSetting = value;
    notifyListeners();
  }
}
```

Listeners Update App

```
void main() async {

 // Print the path to the temp directory

 await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);

 runApp(

   MultiProvider(

     providers: [

       ChangeNotifierProvider(create: (context) => ColorProvider()),

       ChangeNotifierProvider(create: (context) => PriceProvider()),

     ],

     child: const MyApp(),

   ),

 );

}
```

Settings.dart

Drop down color

DropdownButton<String> Set

.colorSetting based on string

```
Text("Color Scheme"),
DropdownButton<String>(
  value: dropdownValue,
  onChanged: (String? newValue) {
    setState(() {
      dropdownValue = newValue!;

      if (dropdownValue == 'Blue') {
        Provider.of<ColorProvider>(context, listen: false)
            .colorSetting = Colors.blue;
      } else if (dropdownValue == 'Green') {
        Provider.of<ColorProvider>(context, listen: false)
            .colorSetting = Colors.green;
      } else if (dropdownValue == 'Red') {
        Provider.of<ColorProvider>(context, listen: false)
            .colorSetting = Colors.red;
      }
      //use the Provider.of method to get the instance of ColorProvider and update the color setting
    });
  },
```

Use Provider.of<ColorProvider>(context) to access color stored in provider

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Easy Birthdays'),
      backgroundColor: Provider.of<ColorProvider>(context)
```