T.C.
ERCIYES ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

MOBILE APPLICATION DEVELOPMENT DERSİ
ARA RAPORU

ReOrderableList — Öğe Sırasını Değiştirme.

Durum Yönetimi

1030510588 Emirhan Uğurlu

DR. ÖGR. ÜYESİ FEHİM KÖYLÜ

Nisan 2025

DURUM YÖNETİMİ

Flutter’da durum yönetimi (state management), uygulamanın kullanıcı arayüzü (UI) ile veri (state) arasındaki ilişkiyi kontrol etmek için kullanılır. Flutter, reactive (tepkisel) bir framework olduğu için, bir veride (state) değişiklik olduğunda bu değişikliği UI'ye yansıtmak için durum yönetimi mekanizmalarına ihtiyaç duyulur.

Flutter’da durum yönetimi temel olarak ikiye ayrılır:

1. Basit (Yerel) Durum Yönetimi

Genellikle küçük uygulamalar veya tek bir widget ağacında kullanılan durumlar için uygundur.

- setState() (En sık tercih edilen, ben de bu başlığı tercih ettim.)
- InheritedWidget / InheritedModel
- ValueNotifier + ValueListenableBuilder

Örnek olarak verilebilir.

Avantajları:

- Öğrenmesi kolaydır.
- Küçük uygulamalarda yeterlidir.

Dezavantajları:

- Karmaşık uygulamalarda sürdürülebilir değildir.
- State yukarıdan aşağıya taşınmak zorunda kalır ("prop drilling").

2. Gelişmiş Durum Yönetimi

Daha büyük uygulamalar için, farklı sayfalarda (widget ağaçlarında) state'e ulaşmak gerektiğinde tercih edilir.

Popüler Paketleri:

- **Provider** (Google tarafından önerilir)
- **Riverpod** (Provider'ın daha esnek ve modern versiyonu)
- **Bloc / Cubit (flutter_bloc)**
- **GetX**
- **MobX**
- **Redux**

Hangi durumda hangi durum yönetimi aracı kullanılmalı?

Senaryo	**Önerilen Yönetim Yöntemi**
Tek sayfa, basit UI	setState()
Birden fazla widget etkileniyorsa	Provider, Riverpod
İş mantığı karmaşıksa	Bloc, Cubit
Minimalist ama güçlü çözüm	GetX

Tablo 1. Örnek Senaryolar

Basit kod örneği:

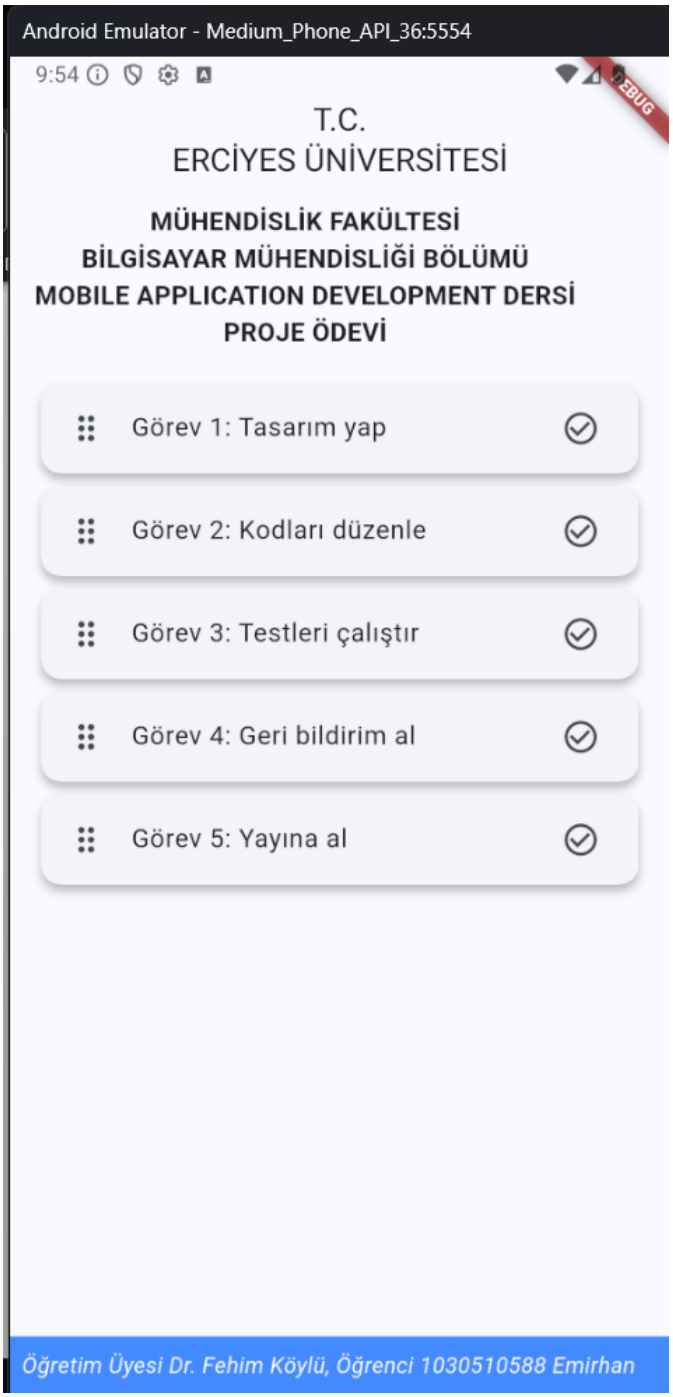
```
import 'package:flutter/material.dart';


// Ana uygulamayı başlatan main fonksiyonu
void main() => runApp(MyApp()); // MyApp widget'ını başlatır


// StatelessWidget kullanan MyApp sınıfı
class MyApp extends StatelessWidget {
  // build metodu, uygulamanın temel yapısını oluşturur.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: CounterPage(),  // Uygulamanın ana sayfasını CounterPage olarak ayarlar
    );
  }
}


// CounterPage, sayacı tutan ve artıran StatefulWidget
class CounterPage extends StatefulWidget {
  @override
  _CounterPageState createState() => _CounterPageState();  // CounterPage için durumu tutacak State sınıfını oluşturur
}


// _CounterPageState, sayacı kontrol eden ve UI'yi güncelleyen sınıf
```

```
class _CounterPageState extends State<CounterPage> {
 int _counter = 0;  // Sayacın başlangıç değeri

 // _increment metodu, sayacı 1 artıran fonksiyondur
 void _increment() {
  setState(() {
   _counter++;  // setState çağrıldığında UI yeniden oluşturulacak ve sayacın değeri artacaktır
  });
 }

 // build metodu, ekranın tasarımını yapar
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   body: Center(
    child: Text(
     'Sayı: $_counter',  // Ekranda sayacın mevcut değerini gösterir
     style: TextStyle(fontSize: 24),  // Yazının boyutunu ayarlar
    ),
   ),
   // Sayacın artırılmasını sağlayan floating action button
   floatingActionButton: FloatingActionButton(
    onPressed: _increment,  // Butona tıklanınca _increment metodunu çalıştırır
    child: Icon(Icons.add),  // Artırma simgesini (artı işareti) gösterir
   ),
  );
 }
}
```

Reorderable list — öğe sırasını değiştirme

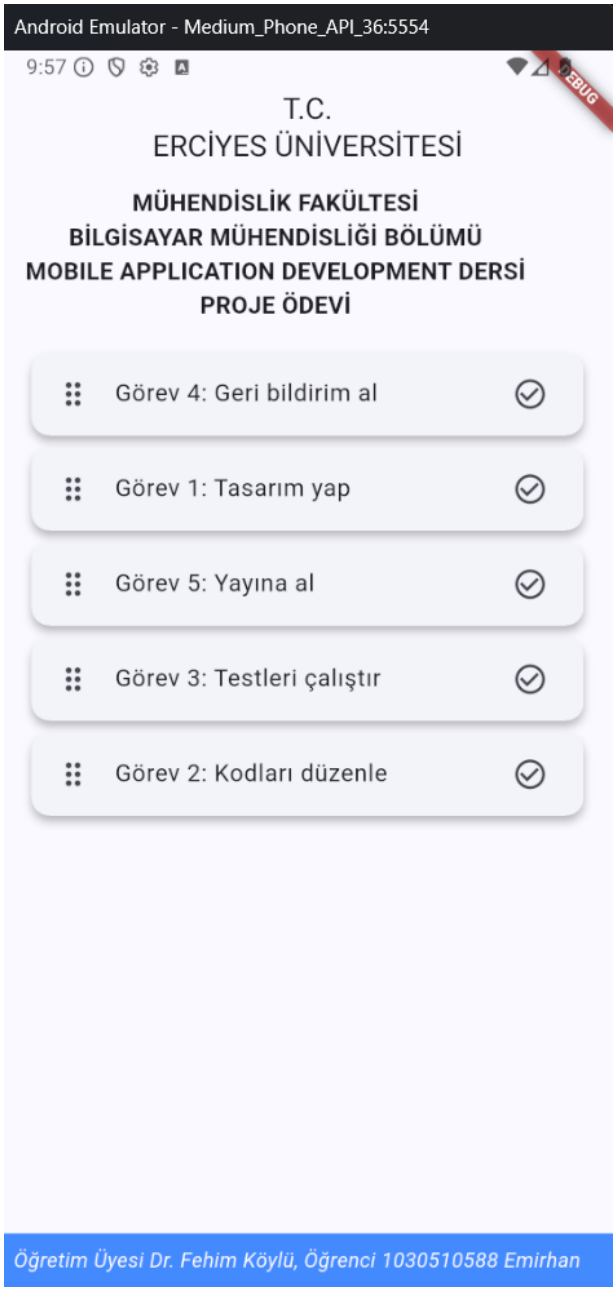
Reorderable list — öğe sırasını değiştirme başlıklı uygulama ödevimde araştırma ödevim konusu olan **durum yönetimini** de kullandım.

Öncelikle MyApp widget'ini çalıştırdım. Akabinde MaterialApp ile temel yapıları kurdum. Bir stateful widget oluşturdum. İçerisinde bir liste yapısı oluşturarak sırası değişecek görevleri belirledim. ReorderableListView ile öğeleri sürükleyerek bırakmayı sağladım. SetState ile UI kendini sayfada sürükleme bırak yaptığımda sayfa kendini güncelliyor.



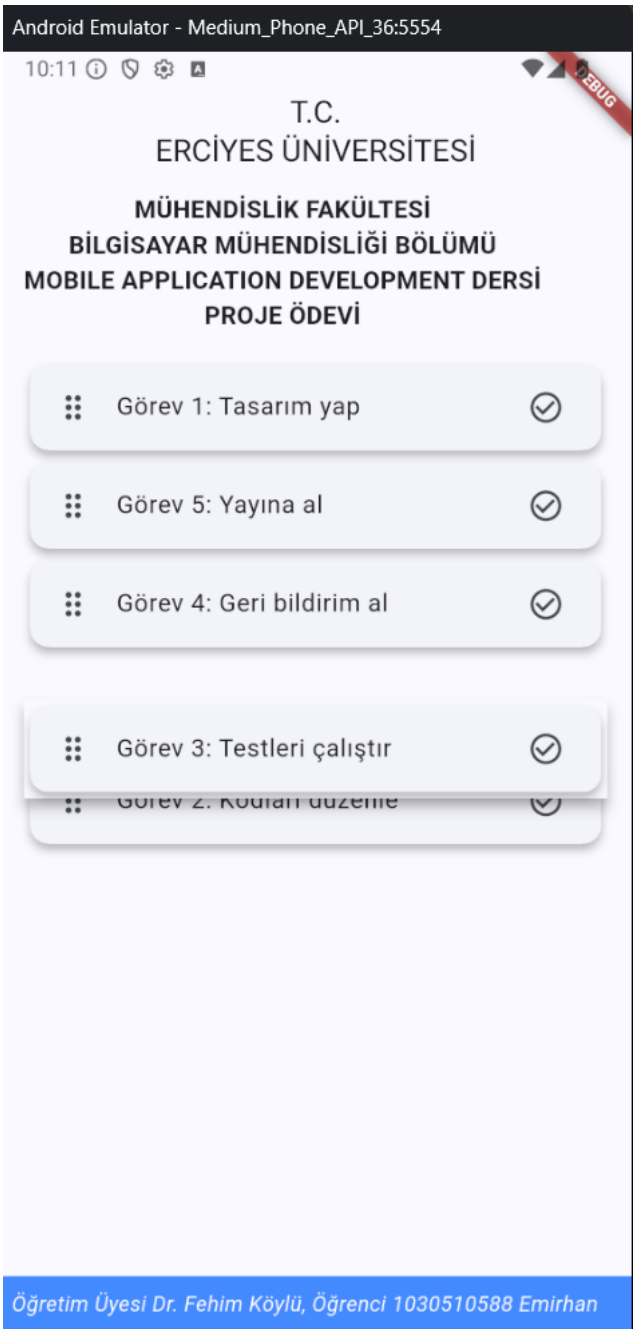
Görsel 1. Sayfa Düzeni

Görsel 1. de gözüktüğü üzere uygulama ilk açıldığında Android Studio kullanılarak emulator aracılığı ile mobil uygulama üzerinden görüntülenebiliyor. Uygulama çalıştığında sayfa düzeni bu şekilde yer almaktadır. Listede yer alan görevlere tıklandığında sürüklenebilme özelliği bulunuyor ve liste istenen sırada organize edilebiliyor.



Görsel 2. Listenin değiştirilmiş hali (ReOrderableList kullanımı)

Görsel 2’de ise ReOrderableList yapısı kullanılarak elde edilen görüntü yer almaktadır.



Görsel 3. ReOrderableList Kullanımı

Görsel 3’de ReOrderableList yapısı kullanıldığı an elde edilen görüntü yer almaktadır. Listedeki elemanların sürükle bırak ile sırasını değiştirirken görsel alınmıştır.

Proje kodları (main.dart) ek kısmında yer almaktadır.

EK.

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


void main() => runApp(MyApp());


class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'UI Prototip - Reorderable List',
   theme: ThemeData(
    colorScheme: ColorScheme.fromSeed(seedColor: Colors.blueAccent),
    useMaterial3: true,
   ),
   home: ReorderableListUI(),
  );
 }
}


class ReorderableListUI extends StatefulWidget {
 @override
 _ReorderableListUIState createState() => _ReorderableListUIState();
}


class _ReorderableListUIState extends State<ReorderableListUI> {
 List<String> items = [
  'Görev 1: Tasarım yap',
  'Görev 2: Kodları düzenle',
  'Görev 3: Testleri çalıştır',
  'Görev 4: Geri bildirim al',
  'Görev 5: Yayına al',
 ];
```

```
@override
Widget build(BuildContext context) {
 return Scaffold(
appBar: AppBar(
 title: Center(
 child: Text(
  'T.C.\nERCİYES ÜNİVERSİTESİ',
  style: TextStyle(fontSize: 20),
  textAlign: TextAlign.center,
 ),
),
),
body: Padding(
 padding: const EdgeInsets.symmetric(vertical: 12, horizontal: 16),
 child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
   Text(
    'MÜHENDİSLİK FAKÜLTESİ\nBİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ\nMOBILE APPLICATION DEVELOPMENT DERSİ\nPROJE ÖDEVİ',
    style: TextStyle(
     fontSize: 16,
     fontWeight: FontWeight.bold,
    ),
     textAlign: TextAlign.center,      ),
   SizedBox(height: 16), // Boşluk
   Expanded(
    child: ReorderableListView(
     buildDefaultDragHandles: false,
     onReorder: (oldIndex, newIndex) {
      setState(() {
       if (newIndex > oldIndex) newIndex -= 1;
       final item = items.removeAt(oldIndex);
```

```
          items.insert(newIndex, item);
        });
      },
      children: List.generate(items.length, (index) {
        return Card(
          key: ValueKey(items[index]),
          elevation: 4,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(12),
          ),
          child: ListTile(
            title: Text(items[index]),
            leading: ReorderableDragStartListener(
              index: index,
              child: Icon(Icons.drag_indicator),
            ),
            trailing: Icon(Icons.check_circle_outline),
          ),
        );
      }),
    ),
  ),
 ],
 ),
),
bottomNavigationBar: BottomNavigationBar(
 items: const <BottomNavigationBarItem>[
  BottomNavigationBarItem(
   icon: Icon(Icons.home),
   label: 'Home',
  ),
  BottomNavigationBarItem(
   icon: Icon(Icons.search),
```

```
      label: 'Search',
    ),
    BottomNavigationBarItem(
      icon: Icon(Icons.account_circle),
      label: 'Profile',
    ),
  ],
  backgroundColor: Colors.blueAccent,
 ),
 bottomSheet: Container(
  padding: EdgeInsets.all(8),
  color: Colors.blueAccent,
  height: 35,
  child: Center(
   child: Text(
    'Öğretim Üyesi Dr. Fehim Köylü, Öğrenci 1030510588 Emirhan Uğurlu',
    style: TextStyle(
     fontSize: 14,
     fontStyle: FontStyle.italic,
     color: Colors.white,
    ),
   ),
  ),
 ),
);
 }
}
```