T.C.
BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ
PAZARYERi MESLEK YÜKSEKOKULU
BİLGİSAYAR PROGRAMCILIĞI BÖLÜMÜ

UYGULAMA TASARIM RAPORU

Öğrenci Zeliha Alcık
Öğrenci Numarası [redacted]

ÖĞRETİM GÖREVLİSİ
ZAFER SERİN

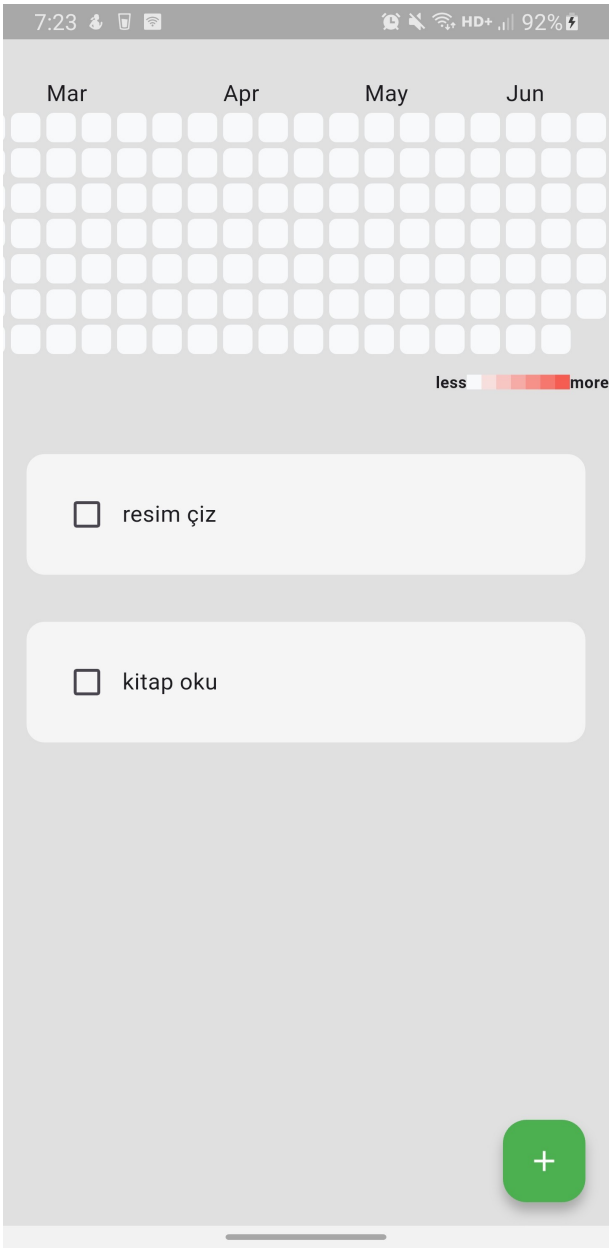
BLP231 Mobil Uygulamalara Giriş

BİLECİK 2026

İÇİNDEKİLER

1 Giriş ve Proje Amacı

Bu proje, kullanıcıların günlük alışkanlıklarını düzenli olarak takip etmelerini, yeni olumlu alışkanlıklar edinmelerini ve mevcut alışkanlıklarını sürdürmelerini sağlamak amacıyla geliştirilmiş bir "Alışkanlık Takip Uygulaması"dır. Modern yaşamın getirdiği yoğunlukta, bireylerin hedeflerine ulaşmaları ve kişisel gelişimlerini desteklemeleri için alışkanlık takibinin önemi büyüktür. Bu uygulama, kullanıcı dostu arayüzü ve sezgisel özellikleriyle bu süreci kolaylaştırmayı hedeflemektedir.



Şekil 1.1: Yeni Alışkanlık Uygulama

2 Kullanılan Teknolojiler

Uygulamanın geliştirilmesinde aşağıdaki teknolojiler ve kütüphaneler kullanılmıştır:

Flutter

Google tarafından geliştirilen, iOS ve Android dahil olmak üzere çoklu platformlar için hızlı uygulama geliştirmeye olanak tanıyan açık kaynaklı UI yazılım geliştirme kitidir. Uygulamanın modern ve akıcı kullanıcı arayüzü Flutter ile oluşturulmuştur.

Hive

Dart için geliştirilmiş, hafif ve hızlı bir NoSQL veritabanı çözümüdür. Uygulama verilerinin (alışkanlık listeleri, tamamlama durumları vb.) yerel olarak ve çevrimdışı çalışacak şekilde depolanması için kullanılmıştır. Hive, performanslı yerel veri saklama ihtiyaçlarımızı karşılamıştır.

Flutter Slidable

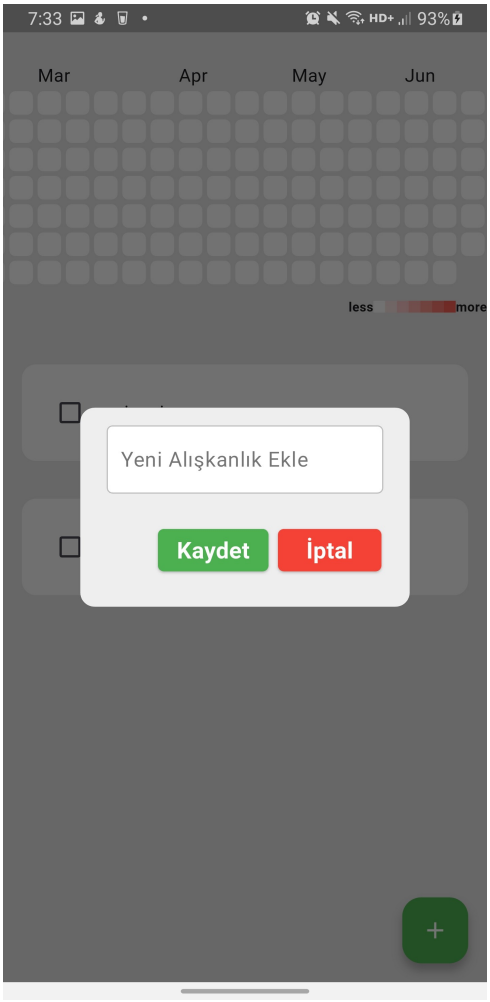
Listelerdeki öğeleri kaydırarak ek eylemler (düzenleme, silme) gerçekleştirmeye olanak tanıyan bir Flutter kütüphanesidir. Bu, alışkanlık listesinin etkileşimini ve kullanışlılığını artırmıştır.

3 Uygulamanın Temel Özellikleri

Uygulama, kullanıcıların alışkanlıklarını etkili bir şekilde yönetmelerine olanak tanıyan aşağıdaki temel özelliklere sahiptir:

Alışkanlık Ekleme

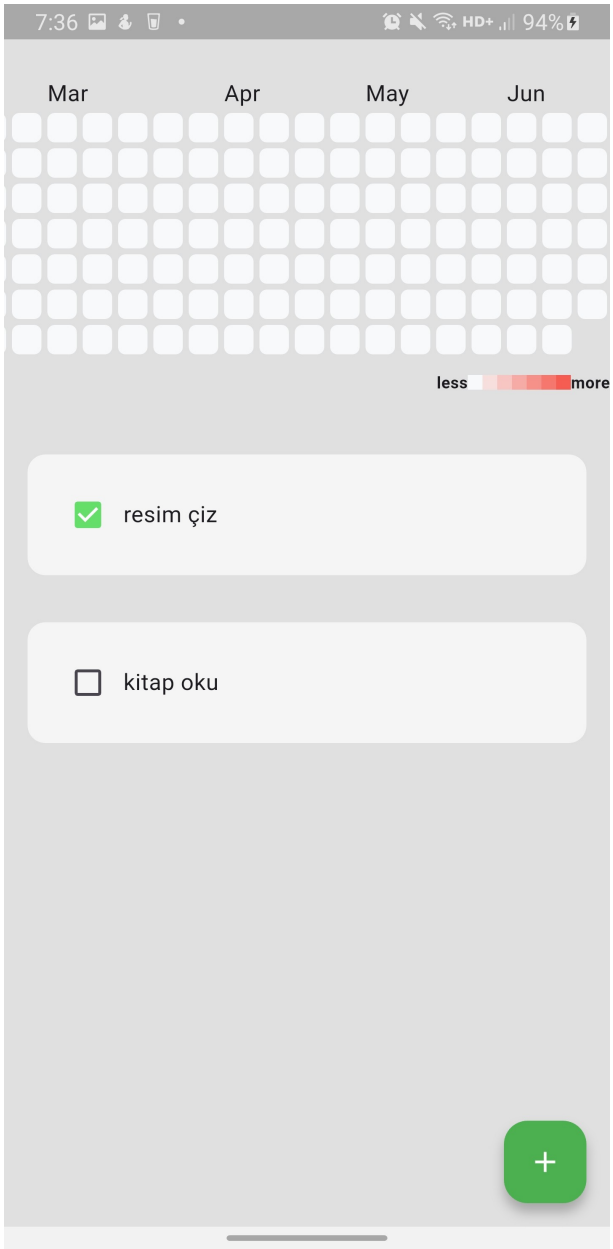
Kullanıcılar, ana ekrandaki sağ alt köşede bulunan '+' (Floating Action Button) simgesine dokunarak açılan diyalog kutusuna alışkanlık adını girerek yeni bir alışkanlık oluşturabilirler. Bu sayede kişisel hedeflerine uygun, özelleştirilebilir alışkanlıklar tanımlayabilirler.



Şekil 3.1: Yeni Alışkanlık Ekleme Ekranı

Alışkanlık Tamamlama Takibi

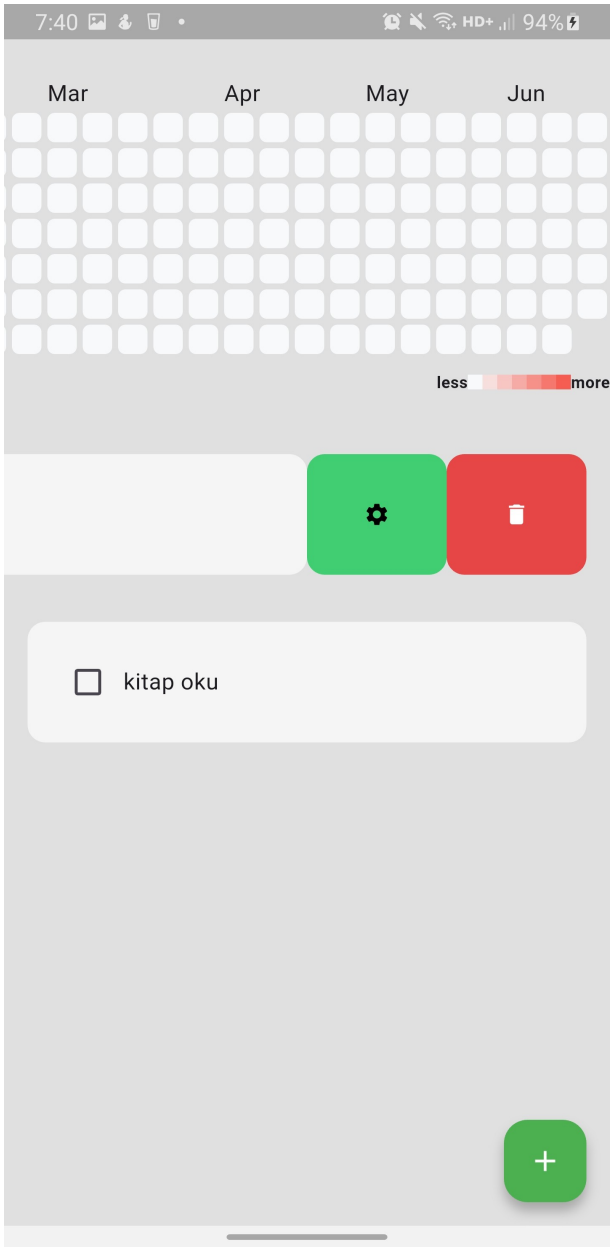
Her bir alışkanlık, yanında bulunan onay kutucuğu (checkbox) aracılığıyla günlük olarak "tamamlandı" veya "tamamlanmadı" olarak işaretlenebilir. Uygulama, her yeni günde alışkanlıkların durumunu otomatik olarak sıfırlayarak, kullanıcının o güne özel takibini kolaylaştırır ve temiz bir başlangıç sunar.



Şekil 3.2: Alışkanlık Listesi ve Tamamlama Durumu

Alışkanlık Düzenleme ve Silme

Kullanıcılar, listedeki bir alışkanlığı sağa kaydırarak çıkan seçenekler (ayarlar ve çöp kutusu simgeleri) aracılığıyla kolayca düzenleyebilir veya silebilirler. Bu etkileşimli özellik, alışkanlık listesinin dinamik yönetimini sağlar.



Şekil 3.3: Alışkanlık Düzenleme/Silme Seçenekleri (Slidable)

Yerel Veri Saklama

Tüm kullanıcı verileri (alışkanlık listeleri, günlük tamamlama durumları ve geçmiş yüzdeler), Hive NoSQL veritabanı kullanılarak cihazda güvenli ve hızlı bir şekilde saklanır. Bu sayede uygulama, internet bağlantısı olmasa bile sorunsuz bir şekilde çalışmaya devam eder ve kullanıcı verilerinin gizliliğini korur.

```
import 'package:flutter/material.dart';
import 'package:hive/hive.dart';
import 'package:hive_todo_app/components/habit_tile.dart';
import 'package:hive_todo_app/components/my_fab.dart';
import 'package:hive_todo_app/components/my_alert_box.dart';
import 'package:hive_todo_app/data/habit_database.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  HabitDatabase db = HabitDatabase();
  final _myBox = Hive.box('Habit_Database');

  @override
  void initState() {
    // Uygulama ilk kez açılıyorsa varsayılan veriyi oluştur
    if (_myBox.get("START_DATE") == null) {
      db.createDefaultData();
    } else {
      // Zaten veri varsa yükle
      db.loadData();
    }
    // Veritabanını güncelle (günlük sıfırlama vb. için)
    db.updateDatabase();
    super.initState();
```

```
31  }
32
33  // Checkbox'a dokunulduğunda
34  void checkBoxTapped(bool? value, int index) {
35    setState(() {
36      db.todaysHabitList[index][1] = value;
37    });
38    db.updateDatabase(); // Veritabanını güncelle
39  }
40
41  // Yeni alışkanlık ekleme metin denetleyicisi
42  final _newHabitController = TextEditingController();
43
44  // Diyalog kutusunu temizleyip kapatan yardımcı fonksiyon
45  void _clearAndPopDialog() {
46    _newHabitController.clear();
47    Navigator.of(context).pop();
48  }
49
50  // Yeni bir alışkanlık ekleme diyalogunu aç
51  void createNewHabit() {
52    showDialog(
53      context: context,
54      builder: (context) {
55        return MyAlertBox(
56          controller: _newHabitController,
57          hintText: 'Yeni Alışkanlık Ekle',
58          onSave: savedNewHabit,
59          onCancel: cancelDialogBox,
60        );
61      },
62    );
63  }
64
65  // Yeni alışkanlığı kaydet
66  void savedNewHabit() {
67    setState(() {
```

```
      db.todaysHabitList.add([
        _newHabitController.text,
        false, // Varsayılan olarak tamamlanmadı
      ]);
      _clearAndPopDialog();
      db.updateDatabase();
    });
  }

  // Diyalog kutusunu iptal et
  void cancelDialogBox() {
    _clearAndPopDialog();
  }

  // Alışkanlık ayarları diyalogunu aç
  void openHabitSettings(int index) {
    _newHabitController.text = db.todaysHabitList[index][0];
    showDialog(
      context: context,
      builder: (context) {
        return MyAlertBox(
          controller: _newHabitController,
          hintText: 'Alışkanlık Adını Güncelle',
          onSave: () => savedExistingHabit(index),
          onCancel: cancelDialogBox,
        );
      },
    );
  }

  // Mevcut alışkanlığı yeni adıyla kaydet
  void savedExistingHabit(int index) {
    setState(() {
      db.todaysHabitList[index][0] = _newHabitController.text;
    });
    _clearAndPopDialog();
    db.updateDatabase();
```

```
105   }
106
107   // Alışkanlığı sil
108   void deleteHabit(int index) {
109     setState(() {
110       db.todaysHabitList.removeAt(index);
111     });
112     db.updateDatabase();
113   }
114
115   @override
116   Widget build(BuildContext context) {
117     return Scaffold(
118       backgroundColor: Colors.grey[300],
119       floatingActionButton: MyFloatingActionButton(onPressed:
            createNewHabit),
120       body: ListView(
121         children: [
122           // Alışkanlıkları listeleyen Builder
123           ListView.builder(
124             shrinkWrap: true,
125             physics: const NeverScrollableScrollPhysics(),
126             itemCount: db.todaysHabitList.length,
127             itemBuilder: (context, index) {
128               return HabitTile(
129                 habitName: db.todaysHabitList[index][0],
130                 habitCompleted: db.todaysHabitList[index][1],
131                 onChanged: (value) => checkBoxTapped(value, index),
132                 settingsTapped: () => openHabitSettings(index),
133                 deleteTapped: () => deleteHabit(index),
134               );
135             },
136           ),
137         ],
138       ),
139     );
140   }
```

```
141 }
```

4 Uygulamanın Mimarisi ve Veri Yönetimi

Uygulama, temel Flutter widget yapısı üzerine inşa edilmiştir. Veri yönetimi için `HabitDatabase` adında özel bir sınıf kullanılmıştır. Bu sınıf:

- Alışkanlık listesini yönetir (`todaysHabitList`).
- Günlük tamamlama yüzdelerini hesaplar ve kaydeder.
- Hive kutusu (`Habit_Database`) ile tüm verileri yerel olarak kaydeder ve yükler.
- `data_time.dart` yardımcı dosyası ile tarih formatlama ve `DateTime` objeleri oluşturma işlemleri merkezi olarak yönetilir.

5 Sonuç ve Gelecek Geliştirmeler

Bu alışkanlık takip uygulaması, basit ve etkili bir kişisel gelişim aracı olarak tasarlanmıştır. Temel özellikleriyle kullanıcıların alışkanlıklarını verimli bir şekilde yönetmelerine olanak tanır.

Gelecekteki geliştirmeler arasında hatırlatıcı bildirimler, daha detaylı istatistikler, alışkanlık kategorileri ve farklı zaman dilimlerine göre (haftalık, yıllık) raporlama seçenekleri eklenebilir.

Kaynaklar

Kaynaklar

[1] Flutter Documentation. (Erişim tarihi: 20 Haziran 2025). https://flutter.dev/

[2] Hive - A Lightweight and Blazing Fast Key-Value Database. (Erişim tarihi: 20 Haziran 2025). https://pub.dev/packages/hive

[3] flutter_slidable - Pub.dev. (Erişim tarihi: 20 Haziran 2025). https://pub.dev/packages/flutter_slidable

[4] Mitch Koko - HABIT TRACKER • Flutter Tutorial Hive Local Storage Youtube Videosu (Erişim tarihi: 20 Haziran 2025) https://www.youtube.com/watch?v=2VKpq4h3Sdw&t=64s.