Proje Raporu: Lezzet Diyarı

1. Proje Tanımı

Lezzet Diyarı, kullanıcıların çeşitli yemek tariflerine göz atabileceği, yeni tarifler keşfedebileceği bir mobil uygulamadır. Uygulama, kullanıcıların öğle yemeği, akşam yemeği veya kahvaltı gibi öğünler için tarif aramasını kolaylaştırmak amacıyla tasarlanmıştır. Kullanıcılar, uygulama içinde tariflere göz atabilir, arama yapabilir ve ilgili tariflere kolayca erişebilirler. Ayrıca uygulamayla alakalı yorumlarını bırakabilir ve diğer kullanıcıların yorumlarını görüntüleyebilirler. Uygulama, Flutter ve Dart kullanılarak geliştirilmiş olup, görsel açıdan çekici ve kullanıcı dostu bir arayüze sahiptir.

2. Uygulama Bileşenleri

2.1 Açılış/ Hoşgeldiniz Ekranı (WelcomeScreen): Uygulamanın açılış ekranı, kullanıcılara uygulamaya giriş yapma veya tariflere göz atma seçenekleri sunar. Ayrıca, görsel olarak çekici bir arayüz sunar ve kullanıcıyı uygulamaya çeker. Home sayfasından geri tuşuna bastığımızda bu ekrana tekrar erişebiliriz. WelcomeScreen sayfasında 'Tariflere Göz At!' butonuna tıkladığımızda SignUp Page sayfasına yönlendiriliyoruz. Giriş yapmadan uygulamaya erişim sağlanmayacaktır.

2.2. Ana Ekranlar

- **Ana Ekran (HomeScreen):** Kullanıcılar ana ekranda kategori seçenekleriyle karşılaşırlar. Kahvaltı, öğle yemeği ve akşam yemeği gibi kategorilere göre tariflere erişim sağlanır.
- **Kategori Ekranı (MealScreen):** Her kategoriye özel olarak listelenen tarifler arasında gezinebilirler ve ilgilerini çeken tariflere tıklayarak detaylarını inceleyebilirler ve favorileyebilirler. Örneğin, kahvaltı ekranında kahvaltı tarifleri listelenir. Koduyla beraber görseli ilerleyen sayfalarda yer alacak. Vize ödevinde BreakfastScreen, LunchScreen ve DinnerScreen olarak ayırdığım sayfaları MealScreen başlığı altında birleştirdim ve fonksiyonlarla bu tüm kategoriler için çağırma işlemi gerçekleştirdim.

Giriş ve Kayıt Ekranları

- **Giriş Sayfası (LogInPage):** Kullanıcılar burada mevcut hesaplarına giriş yapabilirler. Giriş yapmadan önce, email ve şifre alanlarını doldurmaları gerekir.
- **Kayıt Sayfası (SignUpPage):** Kullanıcı bir hesaba sahip değilse 'Henüz hesabınız yok mu?' yazısının yanındaki 'Kayıt Ol' yazısına tıklayarak bir hesap oluşturabilirler. Kayıt olmadan önce kullanıcı adı, e-posta ve şifre gibi bilgileri girmeleri istenir.

2.3. Arama Ekranı

- **Arama Ekranı (SearchScreen):** Kullanıcılar burada "Bugün Ne Yesem?" yazılı ekranda çıkan 'Değiştir' butonuna tıklayarak rastgele bir yemek görseli görebilirler. Bu

durum kullanıcıların karar vermelerini kolaylaştırır ve yeni tarifler keşfetmelerini sağlar.

2.4 Yorum Ekranı

- Bu sayfa kullanıcının yorum ekleyebileceği, listeleyebileceği ve silebileceği bir arayüz sağlar. Yorumlar **SQLite veritabanında** saklanır ve **QuickAlert** paketi kullanılarak başarı mesajları gösterilir.

2.5. Diğer Bileşenler

- **Navigasyon Çubuğu (MyBottomNavigationBar):** Uygulamanın alt kısmında bulunan gezinme çubuğu, kullanıcılara farklı ekranlar arasında kolay geçiş imkânı sağlar. Ana ekrana, arama ekranı, yorum ekranı veya giriş/kayıt ekranlarına erişim sağlar.

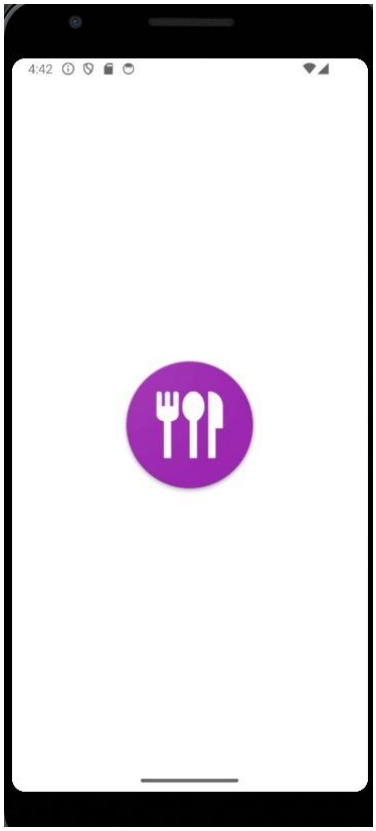
3. Kullanılan Teknolojiler

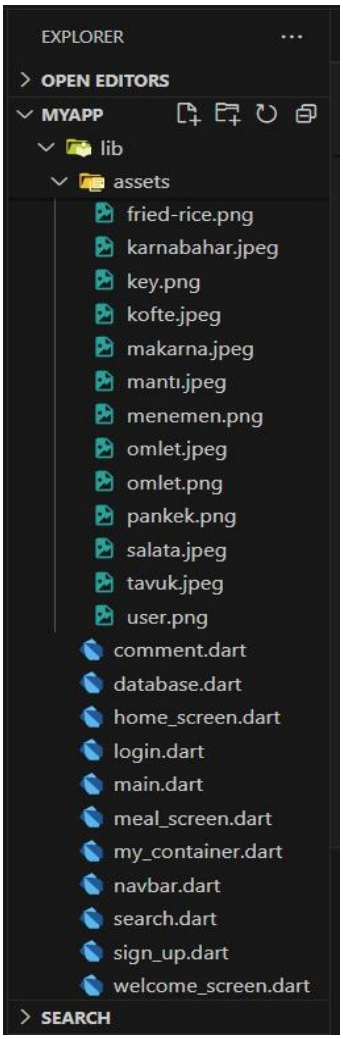
- **Flutter ve Dart:** Uygulama, Google'ın Flutter framework'ü kullanılarak geliştirilmiştir. Dart programlama dili kullanılarak kodlanmıştır.

Package Kullanımı:

- **Google Fonts:** Uygulama içinde metinler için çeşitli Google fontları kullanılmıştır. Bu, uygulamanın görsel estetiğini artırır ve metinleri daha çekici hale getirir.
- **HexColor:** Hex renk kodlarını kolayca kullanmak için HexColor paketi entegre edilmiştir. Bu, uygulama içinde renkleri belirlemeyi ve yönetmeyi kolaylaştırır.
- **Page Transition:** Kullanıcıların açılış ekranından giriş ekranına geçiş yaparken daha akıcı ve görsel olarak çekici bir deneyim yaşamasını sağlar.
- **Quick Alert:** Kullanıcılara hızlı ve özelleştirilebilir uyarılar göstererek uygulama etkileşimini artırır. Yorum ekleme ve silme işlemleri sonrasında bildirimler gösterir.

Uygulama Logosu:





Dosyalarım yandaki gibi gözüküyor.

Main.dart:

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  //Uygulamanın ana sınıfı
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner:
          false, //Hata ayıklama modunda üst köşede bulunan etiketi kaldırır.
      title: 'Yemek Uygulaması',
      home:
          const WelcomeScreen(), //Kullanıcının uygulamayı açtığında ilk
göreceği ekranı
      theme: ThemeData(
```

```
        brightness: Brightness.light, //Uygulamanın tema ayarı
      ),
    );
  }
}
```

Bu **main.dart** dosyası, **Yemek Uygulaması** adlı Flutter uygulamasının giriş noktasını oluşturur. Flutter uygulamasının temel yapılandırmasını ve başlangıç ekranını belirler. Uygulama çalıştırıldığında, kullanıcı ilk olarak **WelcomeScreen** ekranını görecektir.

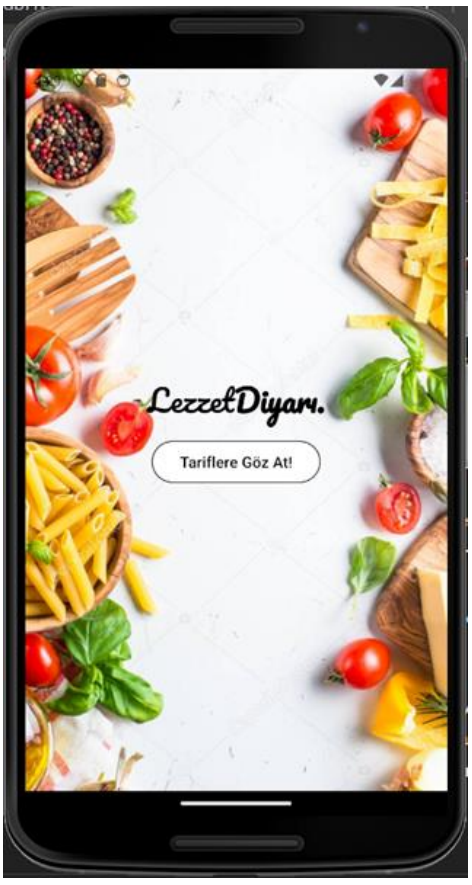
- **import 'welcome_screen.dart';**: Uygulamanın başlangıç noktası olan **WelcomeScreen** sınıfını içeren dosyayı içe aktarır.
- **package:flutter/material.dart**: Flutter'ın temel kullanıcı arayüzü bileşenlerini ve öğelerini içeren pakettir.
- **void main() => runApp(const MyApp());**: **main()** fonksiyonu, uygulamayı çalıştırmak için gereklidir. Bu, **MyApp** sınıfının bir örneğini oluşturur ve bu örneği **runApp()** fonksiyonuna geçirir.
- **class MyApp extends StatelessWidget { ... }**: **MyApp** sınıfı, uygulamanın ana sınıfıdır ve **StatelessWidget** sınıfından türetilir. Bu sınıf, uygulamanın genel yapısını tanımlar.
- **build(BuildContext context) { ... }**: **build** metodu, uygulamanın arayüzünü oluşturur. Bu metod, **MaterialApp** widget'ını döndürür. **MaterialApp**, genel tema ve yönlendirme ayarları gibi uygulamanın genel yapılandırmasını sağlar.
- **debugShowCheckedModeBanner: false,**: Bu özellik, hata ayıklama modunda üst köşede bulunan etiketi kaldırır. Uygulama kullanıcıya sunulurken bu etiketin görüntülenmesi gerekmeyebilir.
- **title: 'Yemek Uygulaması',**: Uygulamanın başlığını belirtir. Bu başlık, uygulama penceresinin üst kısmında görünür.
- **home: WelcomeScreen(),**: Uygulamanın başlangıç ekranını belirtir. **WelcomeScreen**, kullanıcının uygulamayı açtığında ilk göreceği ekranı temsil eder.
- **theme: ThemeData(...),**: Uygulamanın tema ayarlarını tanımlar. Burada, uygulamanın genel olarak açık bir tema kullanacağı belirtilir.

Welcome_screen.dart:

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:page_transition/page_transition.dart'; // Sayfa geçişlerini
yapmak için ekledim
import 'login.dart';

class WelcomeScreen extends StatelessWidget {
  const WelcomeScreen({Key? key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      body: Container(
        width: double.infinity,
        decoration: const BoxDecoration(
          image: DecorationImage(
            image: AssetImage(
              'lib/assets/depositphotos_190518818-stock-photo-food-cooking-
ingredients-background-on.png',
            ),
            fit: BoxFit.cover,
          ),
        ),
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              RichText(
                text: TextSpan(
                  style: GoogleFonts.pacifico(
                    fontSize: 30.0,
                    color: Colors.black,
                  ),
                  children: const [
                    TextSpan(
                      text: "Lezzet",
                    ),
                    TextSpan(
                      text: "Diyarı.",
                      style: TextStyle(fontWeight: FontWeight.bold),
                    ),
                  ],
                ),
              ),
              const SizedBox(
                height: 10,
              ),
              SizedBox(
                width: MediaQuery.of(context).size.width * .4,
                child: TextButton(
                  onPressed: () {
                    Navigator.push(
                      context,
                      PageTransition(
                        type: PageTransitionType
                            .leftToRightWithFade, // Sayfa geçiş animasyonu
belirledik
                        child: LogInPage(),
                      ),
                    );
                  },
```

```
                style: TextButton.styleFrom(
                  backgroundColor: Colors.white, // Butonun arka plan rengi
                  padding: const EdgeInsets.symmetric(
                    vertical: 10, // Butonun dikey iç boşluğu
                    horizontal: 20, // Butonun yatay iç boşluğu
                  ),
                  elevation: 5,
                  side: const BorderSide(
                      color: Colors.black, width: 1), // Kenarlık
                ),
                child: const Text(
                  "Tariflere Göz At!",
                  style: TextStyle(
                    fontSize: 15,
                    color: Colors.black,
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```



Bu **WelcomeScreen** sınıfı, uygulamanın başlangıç ekranını temsil eder.
WelcomeScreen bir StatelessWidget'tir, bu nedenle herhangi bir durum yönetimi gerektirmez ve yapılandırması değiştirilemez. Bu ekran, uygulama başladığında kullanıcıyı karşılamak için kullanılır.

Scaffold widget'ı, bir uygulama sayfasının temel yapı taşını sağlar. Bu durumda, Scaffold'ın gövdesi, tüm ekranı kaplayan bir Container widget'ıdır. Bu container, arka plan olarak bir görüntü kullanır. DecorationImage widget'ı aracılığıyla bu görüntü, AssetImage ile belirtilen bir resim dosyasından alınır ve BoxFit.cover ile ekrana tamamen sığacak şekilde boyutlandırılır.

Ekranın merkezine yerleştirilmiş bir Column widget'ı bulunur. Bu sütun, dikey olarak hizalanmış ve ekranın ortasında bir araya getirilmiş bir dizi widget içerir. İlk olarak, zengin metin widget'ı kullanılarak özel bir yazı tipiyle "Lezzet Diyarı." yazısı görüntülenir. Ardından

kullanıcılar, "Tariflere Göz At!" butonuna tıklayarak giriş sayfasına yönlendirilir. Bu geçiş, PageTransition paketi ile animasyonlu olarak yapılır, böylece kullanıcı deneyimi daha akıcı ve etkileyici hale gelir. Butonun stil özellikleri, arka plan rengi, kenarlık ve iç boşlukları ile özelleştirilmiştir. TextButton widget'ı, kullanıcının tıklamasına yanıt olarak bir işlem gerçekleştirmesini sağlayan basit bir düğmeyi temsil eder. Bu düğme, onPressed özelliği aracılığıyla bir olay dinleyicisine sahiptir. Kullanıcı bu düğmeye tıkladığında, Navigator aracılığıyla LogİnPage ekranına geçilir.

Ekranın tasarımı ve işlevselliği, kullanıcıyı uygulamanın ana işlevselliğine (yani tariflere göz atmaya) yönlendirir.

Log_in.dart:

```
import 'package:flutter/material.dart';
import 'package:myapp/sign_up.dart';
import 'package:myapp/home_screen.dart';
import 'package:myapp/database.dart';
import 'package:page_transition/page_transition.dart'; // Sayfa geçişlerini
yapmak için ekledik

class LogInPage extends StatelessWidget {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Log In'),
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Image.asset(
              'lib/assets/key.png',
              height: 100,
              width: 100,
            ),
            SizedBox(height: 20),
            TextField(
              decoration: InputDecoration(
                labelText: 'Email',
              ),
              controller: emailController,
            ),
            SizedBox(height: 20),
            TextField(
```

```
              obscureText: true,
              decoration: InputDecoration(
                labelText: 'Password',
              ),
              controller: passwordController,
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () async {
                // Giriş işlemini gerçekleştir
                String email = emailController.text;
                String password = passwordController.text;

                // Kullanıcıyı veritabanında kontrol et
                Map<String, dynamic>? user =
                    await DatabaseHelper().getUser(email, password);

                if (user != null) {
                  // Giriş başarılı olduğunda hoş geldiniz alerti göster
                  showDialog(
                    context: context,
                    builder: (_) => AlertDialog(
                      title: Text('Hoş Geldin ${user['username']}'),
                      content: Text('Giriş işlemi başarıyla
gerçekleştirildi.'),
                      actions: <Widget>[
                        TextButton(
                          onPressed: () {
                            Navigator.push(
                              context,
                              PageTransition(
                                type: PageTransitionType
                                    .fade, // Sayfa geçiş animasyonu
belirledim

                                child: HomeScreen(),
                              ),
                            );
                          },
                          child: Text('Tamam'),
                        ),
                      ],
                    ),
                  );
                } else {
                  // Hesabınız yok uyarısı göster
                  showDialog(
                    context: context,
                    builder: (_) => AlertDialog(
                      title: Text('Hata'),
```

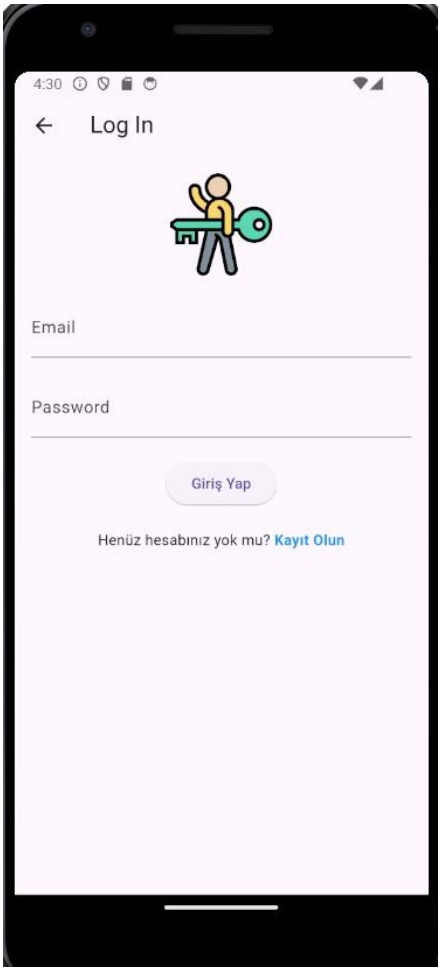
```
                    content: Text(
                        'Kayıtlı hesabınız bulunuyor ise email veya
şifrenizi kontrol ederek tekrar giriniz.'),
                    actions: <Widget>[
                      TextButton(
                        onPressed: () {
                          Navigator.pop(context);
                        },
                        child: Text('Tamam'),
                      ),
                    ],
                  ),
                );
              }
            },
            child: Text('Giriş Yap'),
          ),
          SizedBox(height: 20),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Text("Henüz hesabınız yok mu? "),
              GestureDetector(
                onTap: () {
                  // Sign Up sayfasına yönlendir
                  Navigator.push(
                    context,
                    PageTransition(
                      type: PageTransitionType
                          .rightToLeftWithFade, // Sayfa geçiş animasyonu
belirledik
                      child: SignUpPage(),
                    ),
                  );
                },
                child: Text(
                  "Kayıt Olun",
                  style: TextStyle(
                    color: Colors.blue,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
            ],
          ),
        ],
      ),
    ),
  );
```

```
  }
}
```

LogInPage sınıfı, kullanıcıların uygulamaya giriş yapmasını sağlayan temel ekranı tanımlar. Bu ekran, kullanıcıların email ve şifre bilgilerini girerek uygulamaya erişmelerini sağlayan bir form sunar. Sayfa, AppBar içinde "Log In" başlığıyla kullanıcıyı karşılar ve body kısmında ise SingleChildScrollView ve Column widget'larıyla düzenlenmiştir.

Ekranın merkezine hizalanan Column widget'ı içinde, önce kullanıcıya uygulamanın sembolik resmi olan 'lib/assets/key.png' gösterilir. Ardından kullanıcıdan email ve şifre girmesi istenir; bunun için iki adet TextField kullanılmıştır. Email alanı için "Email" etiketi ve şifre alanı için "Password" etiketiyle kullanıcı girişi yönlendirilir.

"Giriş Yap" butonu, ElevatedButton widget'ı olarak tanımlanmış ve kullanıcıların giriş yapmasını sağlar. Butona tıklandığında, girilen email ve şifre veritabanında kontrol edilir. Başarılı bir giriş durumunda, showDialog ile bir AlertDialog gösterilir. Bu iletişim kutusu, kullanıcının adıyla "Hoş Geldin" mesajı ve giriş işleminin başarıyla gerçekleştirildiğini belirten bir bildirim içerir. "Tamam" butonuna basıldığında, PageTransitionType.fade animasyonuyla kullanıcı HomeScreen'e yönlendirilir.



Başarısız giriş durumunda ise, yine showDialog ile bir AlertDialog gösterilir. Bu durumda kullanıcıya, kayıtlı bir hesapları varsa email veya şifrelerini kontrol etmeleri gerektiğini belirten bir hata mesajı sunulur. "Tamam" butonu ile iletişim kutusu kapatılır ve kullanıcı mevcut sayfada kalır.

Sayfanın alt kısmında, kullanıcıya "Henüz hesabınız yok mu?" mesajı ve bu mesaja bağlı olarak "Kayıt Olun" bağlantısı sunulur. Bu bağlantıya tıklanması durumunda, PageTransitionType.rightToLeftWithFade animasyonu kullanılarak SignUpPage'e yönlendirme yapılır.

LogInPage, kullanıcıların uygulamanın işlevselliğine giriş yapmalarını sağlayan önemli bir arayüz sunar. Kullanıcı dostu bir tasarım ile email ve şifre doğrulaması yapılarak, kullanıcıların güvenli bir şekilde giriş yapmaları ve uygulamanın sunduğu içeriklere erişmeleri sağlanır.

Sign_up.dart:

```
import 'package:flutter/material.dart';
import 'package:myapp/database.dart';
import 'package:myapp/login.dart';
```

```dart
class SignUpPage extends StatelessWidget {
  final TextEditingController usernameController = TextEditingController();
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Sign Up'),
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Image.asset(
              'lib/assets/user.png',
              height: 100,
              width: 100,
            ),
            SizedBox(height: 20),
            TextField(
              decoration: InputDecoration(
                labelText: 'Username',
              ),
              controller: usernameController,
            ),
            TextField(
              decoration: InputDecoration(
                labelText: 'Email',
              ),
              controller: emailController,
            ),
            SizedBox(height: 20),
            TextField(
              obscureText: true,
              decoration: InputDecoration(
                labelText: 'Password',
              ),
              controller: passwordController,
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () async {
                // Kayıt işlemini gerçekleştir
                String username = usernameController.text;
                String email = emailController.text;
                String password = passwordController.text;
```

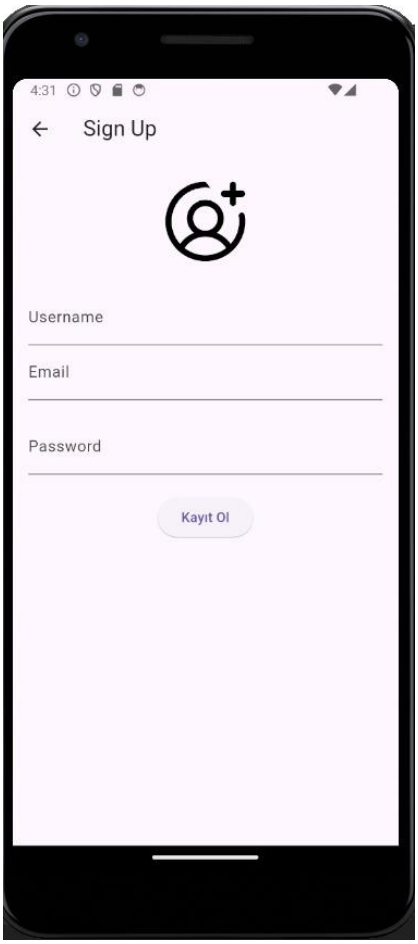
```
                Map<String, dynamic> row = {
                  'username': username,
                  'email': email,
                  'password': password
                };

                int id = await DatabaseHelper().insertUser(row);
                print('Inserted user id: $id');

                // Kayıt işleminden sonra alert göster
                showDialog(
                  context: context,
                  builder: (_) => AlertDialog(
                    title: Text('Kayıt Başarılı'),
                    content: Text('Kayıt işlemi başarıyla gerçekleştirildi.'),
                    actions: <Widget>[
                      TextButton(
                        onPressed: () {
                          // Alert penceresini kapat
                          Navigator.pop(context);
                          // Giriş sayfasına geri dön
                          Navigator.push(
                            context,
                            MaterialPageRoute(
                                builder: (context) => LogInPage()),
                          );
                        },
                        child: Text('Tamam'),
                      ),
                    ],
                  ),
                );
              },
              child: Text('Kayıt Ol'),
            ),
          ],
        ),
      ),
    );
  }
}
```

SignUpPage sınıfı, kullanıcıların uygulamaya kayıt olmalarını sağlayan ekranı temsil eder. Scaffold widget'ı içinde AppBar ile "Sign Up" başlığı kullanıcıyı karşılar ve body kısmında

SingleChildScrollView ve Column widget'larıyla düzenlenmiştir.

Ekranın merkezine hizalanan Column widget'ı içinde, önce kullanıcıya uygulamanın sembolik resmi olan 'lib/assets/user.png' gösterilir. Ardından kullanıcıdan kullanıcı adı (Username), email ve şifre bilgilerini girmesi istenir; bunun için üç adet TextField kullanılmıştır. Email ve şifre alanları için ilgili etiketler belirtilmiştir.

"Kayıt Ol" butonu, ElevatedButton widget'ı olarak tanımlanmış ve kullanıcıların kayıt işlemini gerçekleştirmesini sağlar. Butona tıklandığında, girilen kullanıcı adı, email ve şifre bilgileri bir Map içine alınır ve DatabaseHelper().insertUser(row) metodunu kullanarak veritabanına kaydedilir. Kayıt işlemi başarılı olduğunda, kullanıcıya bir AlertDialog gösterilir. Bu iletişim kutusu, "Kayıt Başarılı" başlığı ve başarı mesajı ile kullanıcıyı bilgilendirir. "Tamam" butonuna basıldığında, AlertDialog kapatılır ve kullanıcı otomatik olarak Giriş sayfasına (LogInPage) yönlendirilir.



Home_screen.dart:

```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'my_container.dart';
import 'meal_screen.dart';
import 'navbar.dart';
import 'login.dart';
import 'search.dart';
import 'comment.dart';
import 'welcome_screen.dart';

class HomeScreen extends StatelessWidget {
  const HomeScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return WillPopScope(
      onWillPop: () async {
        // Geri tuşuna basıldığında WelcomeScreen'e git
        Navigator.pushReplacement(
          context,
```

```
          MaterialPageRoute(builder: (context) => const WelcomeScreen()),
        );
        // Geri tuşunun işlevini iptal etmek için false döndür
        return false;
      },
      child: Scaffold(
        appBar: AppBar(
          title: Text(
            'Lezzet Diyarı',
            style: GoogleFonts.pacifico(
              fontSize: 20.0,
              color: Colors.black,
            ),
          ),
        ),
        body: Column(
          children: <Widget>[
            Expanded(
              child: ListView(
                children: <Widget>[
                  _buildMenuListItem(
                    context,
                    'Kahvaltı',
                    'lib/assets/breakfast.png',
                    'breakfast',
                  ),
                  _buildMenuListItem(
                    context,
                    'Öğle Yemeği',
                    'lib/assets/fried-rice.png',
                    'lunch',
                  ),
                  _buildMenuListItem(
                    context,
                    'Akşam Yemeği',
                    'lib/assets/dinner.png',
                    'dinner',
                  ),
                ],
              ),
            ),
          ],
        ),
        bottomNavigationBar: MyBottomNavigationBar(
          currentIndex: 1,
          onTap: (int index) {
            if (index == 0) {
              // Burada yapılacak bir işlem yoksa
              // geri tuşuna basıldığında önceki sayfaya dönülecek.
```

```
            } else if (index == 1) {
              // Kullanıcı arama öğesine tıkladığında, SearchScreen sayfasına
yönlendirilir.
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => const SearchScreen()),
              );
            } else if (index == 2) {
              //Kullanıcı kayıt ol öğesine tıkladığında, SignUpPage sayfasına
yönlendirilir.
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => FoodPage()),
              );
            } else if (index == 3) {
              //Kullanıcı kayıt ol öğesine tıkladığında, SignUpPage sayfasına
yönlendirilir.
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => LogInPage()),
              );
            }
          },
        ),
      ),
    );
  }

  Widget _buildMenuListItem(
    BuildContext context,
    String content,
    String imagePath,
    String mealType,
  ) {
    return GestureDetector(
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => MealScreen(mealType: mealType),
          ),
        );
      },
      child: MyContainer(
        content: content,
        imagePath: imagePath,
      ),
    );
  }
```

```
}

void main() {
  runApp(const MaterialApp(
    home: HomeScreen(),
  ));
}
```

HomeScreen sınıfı, uygulamanın ana ekranını oluşturan StatelessWidget'ı temsil eder. MaterialApp widget'ı içinde home olarak belirlenmiş ve runApp() fonksiyonuyla uygulamanın başlatılmasını sağlamıştır.

Scaffold widget'ı içinde AppBar ile "Lezzet Diyarı" başlığı kullanıcıya gösterilir. AppBar'ın stilini belirlemek için Google Fonts kullanılarak pacifico fontu ve 20.0 boyutu kullanılmıştır.

Ana ekranın içeriği Column widget'ı içinde düzenlenmiştir. Body kısmında bir ListView bulunur ve içinde üç adet _buildMenuListItem fonksiyonu ile oluşturulmuş liste öğesi vardır. Her bir öğe, kullanıcıların farklı öğünler için seçim yapabileceği resimli ve başlıklı bir konteyneri temsil eder. Öğelere tıklandığında, ilgili öğün türünü temsil eden MealScreen'e yönlendirme yapılır.

BottomNavigationBar widget'ı, uygulamanın altındaki gezinme çubuğunu oluşturur. currentIndex parametresi ile varsayılan olarak seçili olan öğe belirlenmiş ve onTap olayı ile kullanıcının çubuktaki öğelere tıklaması durumunda hangi sayfaya yönlendirileceği belirtilmiştir. Bu öğeler sırasıyla:

- Index 0 (Anasayfa) için herhangi bir işlev belirtilmemiş, bu nedenle geri tuşu varsayılan olarak önceki sayfaya döner.
- Index 1 (Arama) için kullanıcı SearchScreen sayfasına yönlendirilir.
- Index 2 (Yorum) için kullanıcı Comment.dart sayfasına yönlendirilir.
- Index 3 (Giriş Yap) için kullanıcı LogInPage sayfasına yönlendirilir.

Geri tuşuna basıldığında, onWillPop olayı ile belirtilen şekilde, kullanıcı WelcomeScreen'e geri döner ve işlev false döndürülerek geri tuşunun varsayılan işlevi iptal edilmiş olur.

main() fonksiyonu, uygulamanın başlatılmasını sağlar ve MaterialApp içinde HomeScreen sınıfını başlangıç noktası olarak belirler.

Bu kod, kullanıcının uygulamanın ana ekranına erişebileceği bir yol sağlar ve yemek türlerine göz atmasını sağlar.

My_container.dart:

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

class MyContainer extends StatelessWidget {
  final String content;
  final String imagePath;

  const MyContainer({Key? key, required this.content, required
this.imagePath})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      width: double.infinity,
      height: 200,
      decoration: BoxDecoration(
        color: Colors.grey[200],
        borderRadius: BorderRadius.circular(10.0),
      ),
      margin: const EdgeInsets.all(10.0),
      child: Stack(
        children: [
          Positioned.fill(
            child: Image.asset(
              imagePath,
              fit: BoxFit.cover,
            ),
          ),
          Container(
            width: double.infinity,
            height: 200, // Kutunun yüksekliği sabit
            color: Colors.black
                .withOpacity(0.4), // Metin arka plan rengi siyah yarı saydam
            child: Center(
              child: Text(
                content,
                textAlign: TextAlign.center,
                style: TextStyle(
                  fontSize: 20,
                  color: Colors.white,
                ),
              ),
            ),
          ),
        ],
      ),
```

```
    );
  }
}
```

Bu kısımda, MyContainer adında özel bir widget oluşturdum. **MyContainer** sınıfının modüler olarak ayrılması, başka sayfalarda veya bileşenlerde tekrar kullanılabilirliğini sağlar. Bu widget, içeriği ve bir görüntüyü bir araya getirerek bir kutu oluşturur. **content** ve **imagePath** adlı iki zorunlu parametre alır. **content**, kutunun içine yerleştirilecek metni temsil ederken, **imagePath** bir arka plan görüntüsünün yolunu belirtir.

- **Container** widget'ı, belirli bir boyutta ve şekilde bir kutu oluşturur.
- **width** ve **height** özellikleri, konteynerin genişliğini ve yüksekliğini belirler.
- **decoration** özelliği, konteynerin görünümünü özelleştirmek için kullanılır. Burada, gri bir arka plan rengi ve kenarları yuvarlatılmış bir şekil belirlenir.
- **margin** özelliği, konteynerin dış boşluğunu belirtir.
- **child** özelliği olarak bir **Stack** widget'ı kullanılır. **Stack**, birden çok widget'ı üst üste binmiş şekilde yerleştirmek için kullanılır.
- İlk child, tam boyutlu bir **Image.asset** widget'ıdır. Arka plan görüntüsü, tamamen doldurulmuş bir şekilde konteynerın içine yerleştirilir (**BoxFit.cover**).
- İçerik metni, siyah bir yarı saydam arka plan üzerine merkezlenmiş şekilde görüntülenir.

Meal_screen.dart:

```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'my_container.dart';
import 'home_screen.dart';

class MealScreen extends StatefulWidget {
  final String mealType;

  const MealScreen({super.key, required this.mealType});

  @override
  _MealScreenState createState() => _MealScreenState();
}

class _MealScreenState extends State<MealScreen> {
  final List<String> _favoriteRecipes = [];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
```

```
          _getMealTitle(widget.mealType),
          style: GoogleFonts.pacifico(
            fontSize: 20.0,
            color: Colors.black,
          ),
        ),
        leading: IconButton(
          icon: const Icon(Icons.arrow_back),
          onPressed: () {
            Navigator.pop(context); // Önceki ekrana dön
          },
        ),
      ),
      body: Column(
        children: <Widget>[
          Expanded(
            child: ListView(
              children: _buildRecipeList(widget.mealType, context),
            ),
          ),
        ],
      ),
    );
  }

  String _getMealTitle(String mealType) {
    // Öğün türüne göre başlık döndürür
    if (mealType == 'breakfast') {
      return 'Kahvaltı Tarifleri';
    } else if (mealType == 'lunch') {
      return 'Öğle Yemeği Tarifleri';
    } else if (mealType == 'dinner') {
      return 'Akşam Yemeği Tarifleri';
    } else {
      return 'Tarifler';
    }
  }

  List<Widget> _buildRecipeList(String mealType, BuildContext context) {
    // Öğün türüne göre tarifleri içeren widget listesi oluşturur
    List<Widget> recipes = [];
    if (mealType == 'breakfast') {
      recipes = [
        _buildRecipeItem(
          context,
          'Menemen',
          'lib/assets/menemen.png',
          'Kolay Menemen Tarifi:\n\nMalzemeler:\n- 2 adet domates\n- 2 adet
yeşil biber\n- 3 adet yumurta\n- Biraz sıvı yağ\n- Tuz ve
```

```
karabiber\n\nYapılışı:\n1. Domatesleri ve yeşil biberleri küp küp
doğrayın.\n2. Bir tavaya sıvı yağ ekleyin ve ısıtın.\n3. Doğradığınız domates
ve yeşil biberleri tavaya ekleyin, birkaç dakika kavurun.\n4. Üzerine tuz ve
karabiber ekleyin, karıştırın.\n5. Domates ve biberler yumuşadığında çırpılmış
yumurtaları ekleyin.\n6. Yumurtalar pişene kadar karıştırın.\n7. Pişen
menemeni servis yapın. Afiyet olsun!',
        ),
        _buildRecipeItem(
          context,
          'Pankek',
          'lib/assets/pankek.png',
          'Kolay Pankek Tarifi:\n\nMalzemeler:\n- 1 su bardağı un\n- 1 su
bardağı süt\n- 1 adet yumurta\n- 2 yemek kaşığı şeker\n- 1 tatlı kaşığı
kabartma tozu\n- Bir tutam tuz\n- Biraz tereyağı veya sıvı yağ (pişirmek
için)\n\nYapılışı:\n1. Un, süt, yumurta, şeker, kabartma tozu ve tuzu bir
kaseye alıp iyice karıştırın.\n2. Tavaya az miktarda tereyağı veya sıvı yağ
koyun ve ısıtın.\n3. Karışımı kepçe yardımıyla tavaya dökün. Orta ateşte altı
kızarana kadar pişirin.\n4. Altı kızardıktan sonra spatula yardımıyla çevirin
ve diğer tarafını da kızartın.\n5. Pişen pankekleri servis tabağına alıp
istediğiniz şekilde servis yapın. Afiyet olsun!',
        ),
        _buildRecipeItem(
          context,
          'Omlet',
          'lib/assets/omlet.jpeg',
          'Kolay Omlet Tarifi:\n\nMalzemeler:\n- 2 adet yumurta\n- Biraz
süt\n- Tuz ve karabiber\n- Biraz tereyağı veya sıvı yağ (pişirmek
için)\n\nYapılışı:\n1. Yumurtaları bir kaseye kırın ve çırpın.\n2. Biraz süt
ekleyin, tuz ve karabiber ile tatlandırın.\n3. Tavaya biraz tereyağı veya sıvı
yağ ekleyin ve ısıtın.\n4. Yumurta karışımını tavaya dökün.\n5. Altı pişene
kadar spatula yardımıyla karıştırın.\n6. Pişen omleti servis tabağına alıp
istediğiniz şekilde servis yapın. Afiyet olsun!',
        ),
      ];
    } else if (mealType == 'lunch') {
      recipes = [
        _buildRecipeItem(
          context,
          'Tavuklu Salata',
          'lib/assets/salata.jpeg',
          'Malzemeler:\n- 200 gram tavuk göğsü\n- 2 adet domates\n- 1 adet
salatalık\n- Yarım demet maydanoz\n- 1 adet limon\n- Zeytinyağı\n- Tuz,
karabiber\n\nYapılışı:\n1. Tavuk göğsünü haşlayın ve didikleyin.\n2. Domates,
salatalık ve maydanozu doğrayın.\n3. Bir karıştırma kabına tavuk göğsü,
domates, salatalık ve maydanozu ekleyin.\n4. Üzerine limon suyu, zeytinyağı,
tuz ve karabiber ekleyip karıştırın.\n5. Servis yapın. Afiyet olsun!',
        ),
        _buildRecipeItem(
          context,
```

```
          'Karnabahar Graten',
          'lib/assets/karnabahar.jpeg',
          'Malzemeler:\n- 1 adet karnabahar\n- 1 su bardağı rendelenmiş kaşar
peyniri\n- 1 su bardağı süt\n- 2 yemek kaşığı un\n- 2 yemek kaşığı tereyağı\n-
Tuz, karabiber\n\nYapılışı:\n1. Karnabaharı çiçeklerine ayırın ve
haşlayın.\n2. Haşlanmış karnabaharı fırın kabına yerleştirin.\n3. Bir
tencerede tereyağını eritin, unu ekleyip kavurun.\n4. Unun rengi dönene kadar
kavurduktan sonra sütü ekleyin ve karıştırarak pişirin.\n5. Piştikten sonra
tuz ve karabiber ekleyin.\n6. Karnabaharların üzerine sostan dökün,
rendelenmiş kaşar peyniri serpin.\n7. Önceden ısıtılmış 180 derece fırında
peynirler kızarana kadar pişirin. Afiyet olsun!',
        ),
        _buildRecipeItem(
          context,
          'Mantı',
          'lib/assets/mantı.jpeg',
          'Malzemeler:\n- 500 gram kıyma\n- 1 su bardağı un\n- 1 adet
yumurta\n- 1 su bardağı yoğurt\n- 2 diş sarımsak\n- Tereyağı\n- Pul biber,
nane\n\nYapılışı:\n1. Kıymayı yoğurun ve küçük köfteler hazırlayın.\n2. Un,
yumurta ve biraz su ile hamur yoğurun.\n3. Hamuru açın ve küçük kareler
halinde kesin.\n4. Her bir kareye köfte yerleştirin ve kenarlarını birleştirip
mantı şekli verin.\n5. Kaynayan suda mantıları haşlayın.\n6. Sarımsaklı yoğurt
hazırlayın ve mantıların üzerine dökün.\n7. Üzerine kızdırılmış tereyağında
pul biber ve nane ile servis yapın. Afiyet olsun!',
        ),
      ];
    } else if (mealType == 'dinner') {
      recipes = [
        _buildRecipeItem(
          context,
          'Fırında Tavuk',
          'lib/assets/tavuk.jpeg',
          'Malzemeler:\n- 4 adet tavuk but\n- 2 adet patates\n- 2 adet
havuç\n- 1 adet soğan\n- 2 diş sarımsak\n- Zeytinyağı\n- Tuz, karabiber,
kekik\n\nYapılışı:\n1. Tavuk butlarını yıkayın ve temizleyin.\n2. Patatesleri,
havuçları ve soğanı iri iri doğrayın.\n3. Bir fırın kabına tavuk butlarını ve
doğranmış sebzeleri yerleştirin.\n4. Üzerine ezilmiş sarımsak, zeytinyağı,
tuz, karabiber ve kekik serpin.\n5. Önceden ısıtılmış 200 derece fırında
yaklaşık 45 dakika pişirin. Afiyet olsun!',
        ),
        _buildRecipeItem(
          context,
          'Köfte',
          'lib/assets/kofte.jpeg',
          'Malzemeler:\n- 500 gram kıyma\n- 1 adet soğan\n- 1 su bardağı
galeta unu\n- 1 adet yumurta\n- Tuz, karabiber, pul biber\n- Sıvı
yağ\n\nYapılışı:\n1. Kıymayı bir kaseye alın.\n2. Soğanı rendeleyip kıymaya
ekleyin.\n3. Galeta unu, yumurta ve baharatları da ekleyip yoğurun.\n4.
```

```
Yoğrulan kıymadan cevizden biraz büyük parçalar koparıp yuvarlayarak köfte
şekli verin.\n5. Tavaya sıvı yağ koyup köfteleri kızartın. Afiyet olsun!',
        ),
        _buildRecipeItem(
          context,
          'Sebzeli Makarna',
          'lib/assets/makarna.jpeg',
          'Malzemeler:\n- 250 gram makarna\n- 1 adet kabak\n- 1 adet havuç\n-
1 adet kırmızı biber\n- 1 adet yeşil biber\n- 2 diş sarımsak\n- Zeytinyağı\n-
Tuz, karabiber\n\nYapılışı:\n1. Makarnayı haşlayın ve süzün.\n2. Kabak, havuç,
biberleri doğrayın.\n3. Bir tavada zeytinyağında sarımsakları kavurun.\n4.
Doğranmış sebzeleri ekleyip kavurun.\n5. Haşlanmış makarnayı sebzelerin
üzerine ekleyin.\n6. Tuz ve karabiber ekleyip karıştırın. Afiyet olsun!',
        ),
      ];
    }
    return recipes;
  }

  Widget _buildRecipeItem(
    BuildContext context,
    String title,
    String imagePath,
    String recipe,
  ) {
    final isFavorite = _favoriteRecipes.contains(title);

    return GestureDetector(
      onTap: () {
        _showRecipeDialog(context, title, recipe);
      },
      child: Column(
        children: [
          Row(
            children: [
              Expanded(
                child: MyContainer(
                  content: title,
                  imagePath: imagePath,
                ),
              ),
              IconButton(
                icon: Icon(
                  isFavorite ? Icons.favorite : Icons.favorite_border,
                  color: isFavorite ? Colors.red : null,
                ),
                onPressed: () {
                  _toggleFavorite(title);
                },
```

```
              ),
            ],
          ),
          const SizedBox(height: 10),
        ],
      ),
    );
  }

  void _toggleFavorite(String recipeTitle) {
    setState(() {
      if (_favoriteRecipes.contains(recipeTitle)) {
        _favoriteRecipes.remove(recipeTitle);
      } else {
        _favoriteRecipes.add(recipeTitle);
      }
    });
  }

  void _showRecipeDialog(BuildContext context, String title, String recipe) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text(title),
          content: Text(recipe),
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: const Text('Kapat'),
            ),
          ],
        );
      },
    );
  }
}

void main() {
  runApp(const MaterialApp(
    home: HomeScreen(),
  ));
}
```

MealScreen sınıfı, kullanıcının seçtiği öğün türüne göre tarifleri listeleyen bir StatefulWidget'ı temsil eder. MaterialApp widget'ı içinde home olarak belirlenmiş ve runApp() fonksiyonuyla uygulamanın başlatılmasını sağlamıştır.

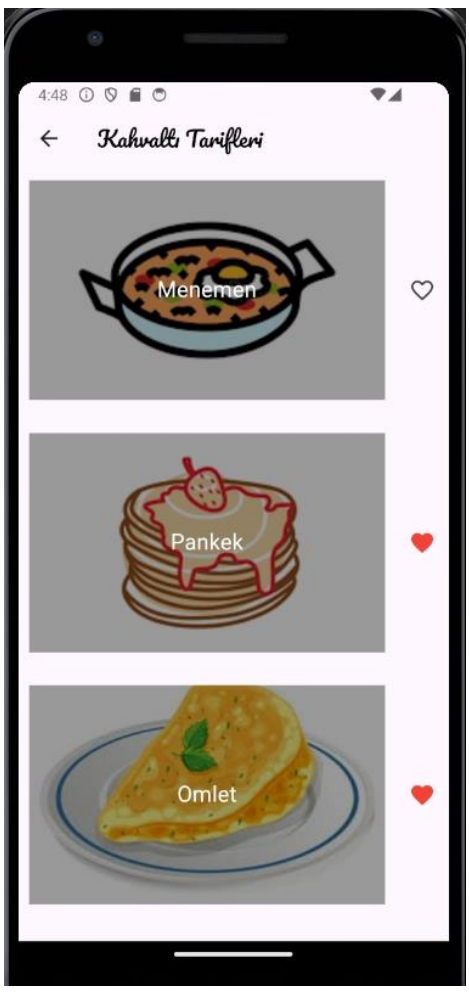
MealScreen sınıfı, State özelliği taşıyan bir sınıf olan _MealScreenState sınıfından türetilmiştir. mealType parametresi ile öğün türünü alır ve başlık olarak AppBar'ın içinde bu türe göre bir başlık gösterir. Başlığın stili için Google Fonts kullanılarak pacifico fontu ve 20.0 boyutu belirlenmiştir.

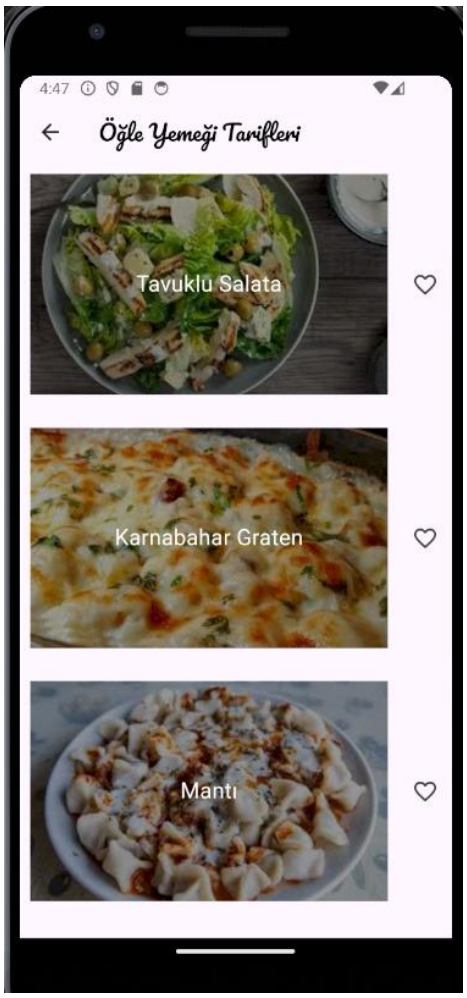
Scaffold widget'ı içinde appBar ve body olarak iki ana bölüm bulunur:

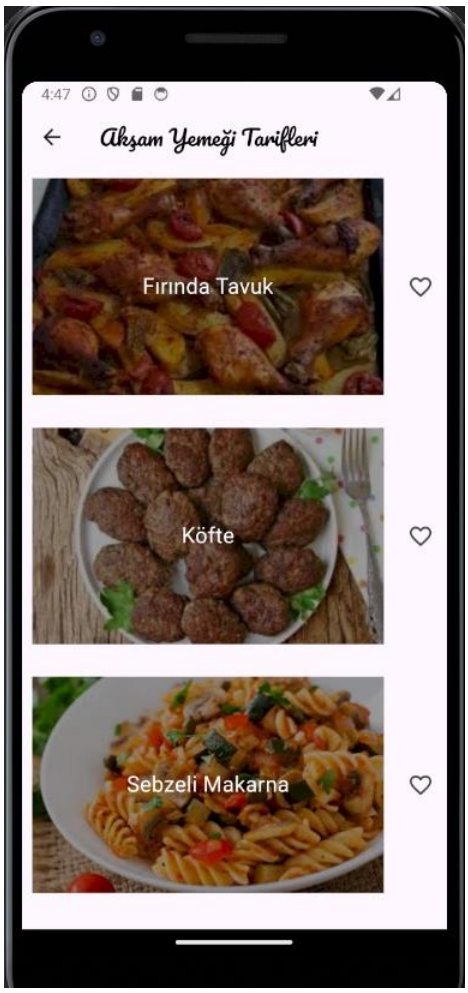
- appBar, kullanıcının seçtiği öğün türünün başlığını ve geri gitmek için kullanılan bir IconButton'ı içerir.
- body kısmında ise ListView widget'ı içinde, _buildRecipeList fonksiyonu kullanılarak seçilen öğün türüne göre tarifler listelenir.

_buildRecipeList fonksiyonu, mealType parametresine göre bir liste oluşturur. Her bir liste öğesi, _buildRecipeItem fonksiyonu ile oluşturulmuş bir widget'dır. Bu öğeler, kullanıcıların tıkladığında tarif detaylarını gösteren bir AlertDialog açan GestureDetector ile sarılmıştır. Ayrıca, her bir tarifin sağ üst köşesinde favorilere ekleme veya çıkarma işlevi sağlayan bir IconButton bulunur.

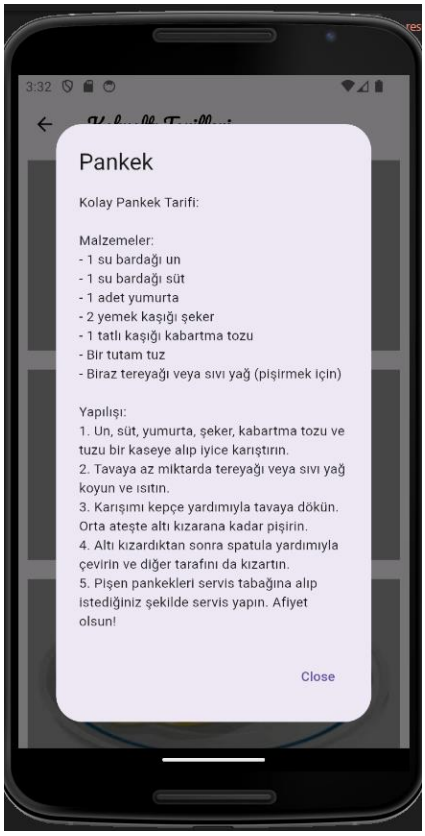
MealScreen sınıfı, kullanıcıların belirli bir öğün türüne göre tariflere erişebildiği, tarifleri görüntüleyebildiği ve favorilere ekleyebildiği interaktif bir ekran sağlar. Bu yapı, uygulamanın kullanıcı deneyimini artıran ve kullanıcıların istedikleri yemeği bulup tarifine ulaşabileceği önemli bir parçadır.





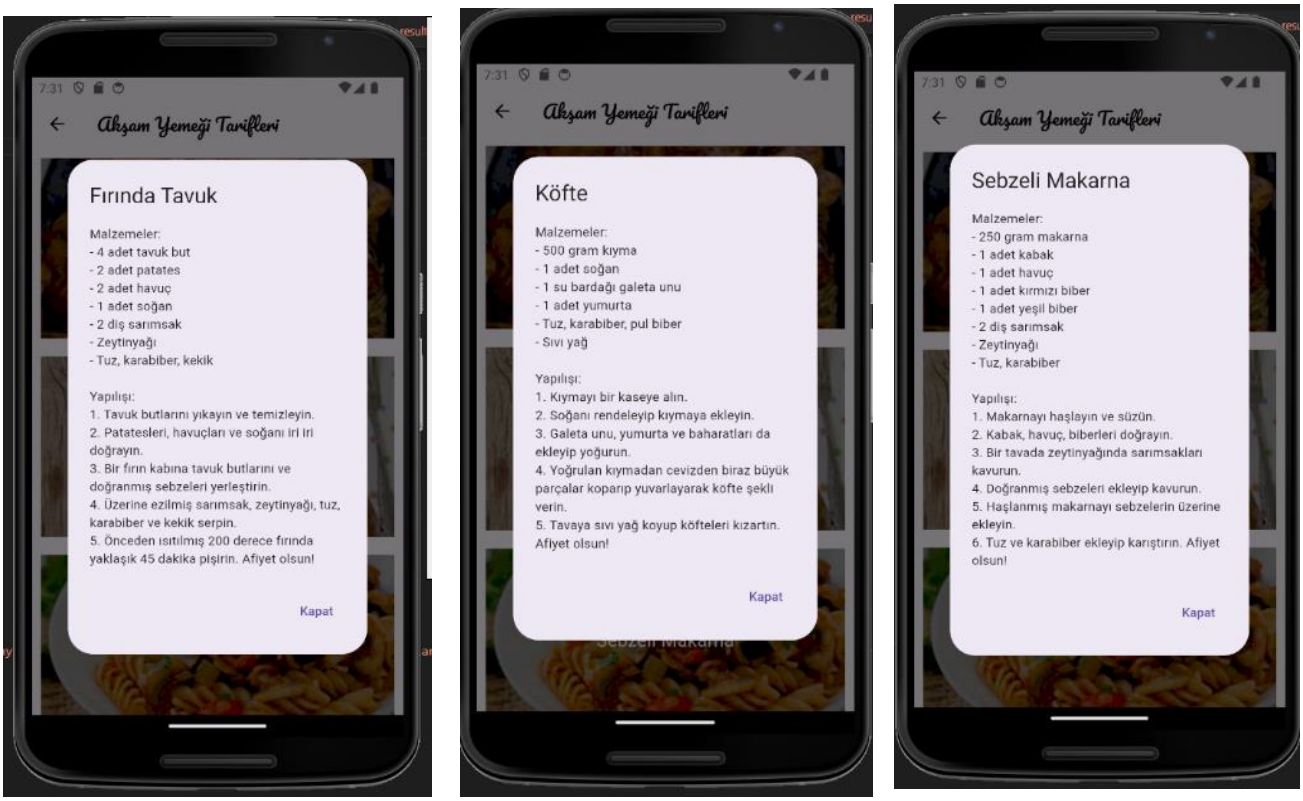


Kahvaltı Tarifleri Kısmı:



Öğle Yemeği Tarifleri Kısmı:

Akşam Yemeği Tarifleri Kısmı:



Search.dart:

```dart
import 'dart:math';
import 'package:hexcolor/hexcolor.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:myapp/comment.dart';
import 'package:myapp/login.dart';
import 'package:myapp/navbar.dart';
import 'home_screen.dart';

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

  @override
  _SearchScreenState createState() => _SearchScreenState();
}

class _SearchScreenState extends State<SearchScreen> {
  String _randomFoodImage = '';

  @override
  void initState() {
```

```dart
    super.initState();
    _setRandomFoodImage(); // Başlangıçta rastgele bir resim belirle
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      backgroundColor: HexColor('#F5F5F5'),
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Center(
            child: Text(
              'Bugün Ne Yesem?',
              style: GoogleFonts.pacifico(
                fontSize: 35.0,
              ),
            ),
          ),
          const SizedBox(
            height: 10.0,
          ),
          Expanded(
            child: Center(
              child: Padding(
                padding: const EdgeInsets.all(16.0),
                child: SizedBox(
                  width: 350,
                  height: 400,
                  child: Image.asset(
                    _randomFoodImage,
                    fit: BoxFit.contain, // Resmin sığdırılma modu
                  ),
                ),
              ),
            ),
          ),
          ElevatedButton(
            onPressed: _setRandomFoodImage,
            child:
                const Text('Değiştir', style: TextStyle(color: Colors.black)),
          ),
          const SizedBox(height: 20.0),
        ],
      ),
      bottomNavigationBar: MyBottomNavigationBar(
        currentIndex: 1,
        onTap: (int index) {
```

```
          if (index == 0) {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const HomeScreen()),
            );
          } else if (index == 1) {
            // Current screen
          } else if (index == 2) {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => FoodPage()),
            );
          } else if (index == 3) {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => LogInPage()),
            );
          }
        },
      ),
    );
  }

  void _setRandomFoodImage() {
    // Rastgele bir yemek resmi seçmek için bir liste oluştur
    final List<String> foodImages = [
      'lib/assets/menemen.png',
      'lib/assets/pankek.png',
      'lib/assets/omlet.jpeg',
      'lib/assets/salata.jpeg',
      'lib/assets/mantı.jpeg',
      'lib/assets/karnabahar.jpeg',
      'lib/assets/kofte.jpeg',
      'lib/assets/makarna.jpeg',
      'lib/assets/tavuk.jpeg',
    ];

    // Rastgele bir resim seçmek için random bir indeks al
    final Random random = Random();
    final int randomIndex = random.nextInt(foodImages.length);

    // Seçilen rastgele resmi ayarla ve setState kullanarak yeniden çiz
    setState(() {
      _randomFoodImage = foodImages[randomIndex];
    });
  }
}
```

SearchScreen sınıfı, kullanıcıların günlük menüye göz atabilecekleri ve rastgele yemek resimlerini görebilecekleri bir ekranı temsil eder. Flutter'da StatefulWidget olarak tanımlanmıştır, bu da içindeki durumun (state) değişebileceği anlamına gelir.

_SearchScreenState sınıfı, SearchScreen widget'ının durumunu yöneten sınıftır. Bu sınıf, ekranın mevcut durumunu (örneğin, gösterilen rastgele yemek resmi) saklar ve kullanıcının etkileşimlerine yanıt olarak arka planda verilerin değişmesini ve kullanıcı arayüzünün güncellenmesini sağlar.

initState metodu, _SearchScreenState sınıfının bir yaşam döngüsü metodudur ve widget'ın oluşturulduğu anda otomatik olarak çağrılır. Bu metot içinde _setRandomFoodImage() fonksiyonu çağrılarak, başlangıçta rastgele bir yemek resmi seçilir ve _randomFoodImage değişkeni güncellenir. Bu sayede, ekran ilk açıldığında kullanıcıya rastgele bir yemek resmi gösterilir. **build** metodu, widget'ın kullanıcı arayüzünü (UI) oluşturur ve her durum güncellendiğinde yeniden çağrılır.

Scaffold Widget: Ana yapıyı oluşturur. AppBar, backgroundColor ve body gibi önemli bileşenleri içerir.

- AppBar: Ekranın üst kısmında yer alır ve başlık barındaki başlık kısmını gösterir.
- backgroundColor: Arka plan rengini belirtir.
- body: Ekranın ana içeriğini oluşturur. Column widget'ı içinde dikey bir düzen oluşturarak, merkezi hizalama ve içerikleri düzenler.

  Gövde İçeriği: Column widget'ı içindeki başlıca bileşenler:

- Başlık: Text widget'ıyla 'Bugün Ne Yesem?' başlığı gösterilir. GoogleFonts kullanılarak Pacifico fontu ve 35.0 font boyutu ile stilize edilmiştir.
- Rastgele Yemek Resmi: _randomFoodImage değişkeni üzerinden Image.asset ile gösterilir. Bu resim, _setRandomFoodImage fonksiyonu ile rastgele seçilir ve ekran boyunca genişletilmiş şekilde Expanded içinde gösterilir.
- Değiştir Butonu: ElevatedButton widget'ı kullanılarak "Değiştir" butonu oluşturulmuştur. Bu butona tıklandığında _setRandomFoodImage fonksiyonu çağrılır ve yeni bir rastgele yemek resmi seçilerek gösterim güncellenir.

bottomNavigationBar: MyBottomNavigationBar widget'ı kullanılarak ekranın alt kısmında gezinme çubuğu oluşturulur. Bu çubuk, farklı sayfalara geçiş yapmayı sağlar.

Bu yapı, kullanıcının rastgele yemek resimlerini görebileceği, arama ekranının temel işlevselliğini sağlayan bir Flutter ekranını tanımlar.

Comment.dart:

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

class FoodPage extends StatefulWidget {
  @override
  _FoodPageState createState() => _FoodPageState();
}

class _FoodPageState extends State<FoodPage> {
  TextEditingController _textEditingController = TextEditingController();
  List<Map<String, dynamic>> foods = [];

  @override
  void initState() {
    super.initState();
    _loadFoods();
  }

  void _loadFoods() async {
    List<Map<String, dynamic>> data = await DatabaseHelper().getAllFoods();
    setState(() {
      foods = data;
    });
  }

  void _addFood(String foodName) async {
    if (foodName.isNotEmpty) {
      await DatabaseHelper().insertFood({'food': foodName});
      _loadFoods();
      QuickAlert.show(
        context: context,
        type: QuickAlertType.success,
        title: 'Success',
        text: 'Comment added successfully!',
      );
    }
  }

  void _deleteFood(int id) async {
    await DatabaseHelper().deleteFood(id);
    _loadFoods();
    QuickAlert.show(
      context: context,
      type: QuickAlertType.success,
      title: 'Success',
      text: 'Comment deleted successfully!',
```

```
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Comments'),
    ),
    body: Column(
      children: [
        Expanded(
          child: ListView.builder(
            itemCount: foods.length,
            itemBuilder: (context, index) {
              return ListTile(
                title: Text(foods[index]['food']),
                trailing: IconButton(
                  icon: Icon(Icons.delete),
                  onPressed: () {
                    _deleteFood(foods[index]['id']);
                  },
                ),
              );
            },
          ),
        ),
        Padding(
          padding: EdgeInsets.all(8.0),
          child: Row(
            children: [
              Expanded(
                child: TextField(
                  controller: _textEditingController,
                  decoration: InputDecoration(
                    hintText: 'Enter comment!',
                  ),
                ),
              ),
              IconButton(
                icon: Icon(Icons.add),
                onPressed: () {
                  String foodName = _textEditingController.text;
                  if (foodName.isNotEmpty) {
                    _addFood(foodName);
                    _textEditingController.clear();
                  }
                },
              ),
```

```
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

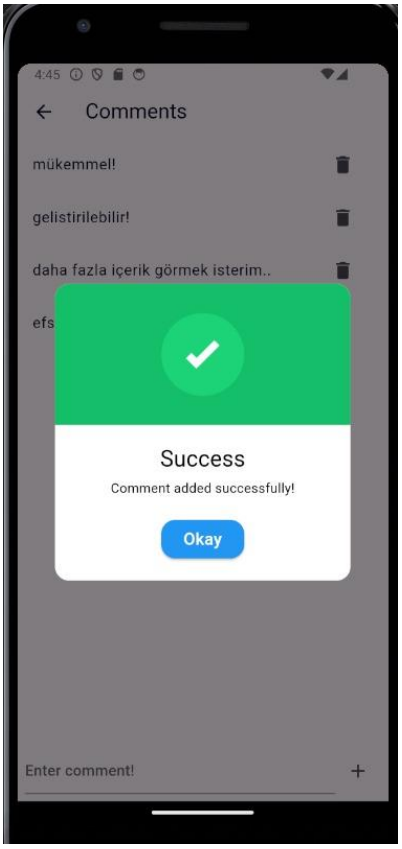
Bu sınıf, Flutter uygulamalarında kullanıcıların uygulama hakkında yorum yapabileceği bir ekranı temsil eder. Bu sınıf, kullanıcıların yeni yorumlar ekleyebileceği, mevcut yorumları görüntüleyebileceği ve listeden yorum silebileceği işlevselliği sunar.

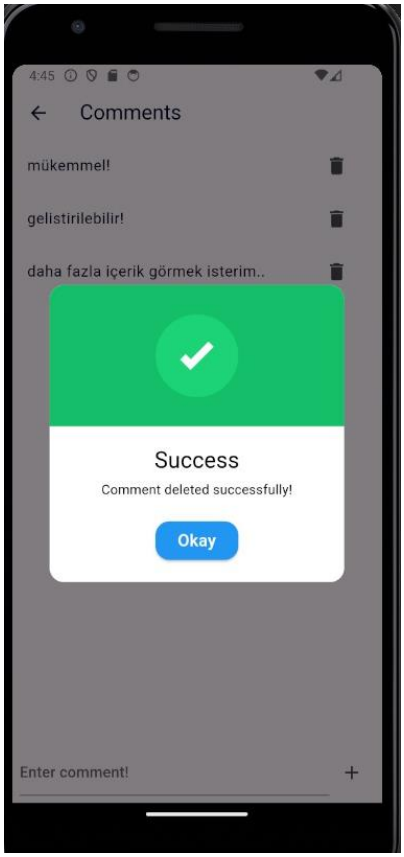
Sınıfın **initState** yöntemi, ekran oluşturulduğunda çağrılır ve **_loadFoods** fonksiyonu ile mevcut yorum listesini yükleme işlemini başlatır. **_loadFoods** fonksiyonu, veritabanından tüm yorum verilerini alır ve **comments** listesini güncelleyerek ekranın yeniden çizilmesini sağlar.

Kullanıcılar, ekranın alt kısmında bulunan bir **TextField** aracılığıyla yeni yorumlar ekleyebilirler. Eklemek istedikleri yorumu girdikten sonra **IconButton** üzerinden ekleyebilirler. Yeni yorum eklenirken **_addFood** fonksiyonu devreye girer. Bu fonksiyon, girilen yorum adını veritabanına ekler ve işlem başarıyla tamamlandığında bir bildirim gösterir.

Ekranın ana bölümünde ise **ListView.builder** kullanılarak **foods** listesi üzerinde döngü oluşturulur. Her bir yorum öğesi, **ListTile** bileşeni içinde listelenir. Her öğenin sağındaki silme işlemi için bir **IconButton** bulunur. Kullanıcı bu düğmeye bastığında seçilen yorumu **_deleteFood** fonksiyonu aracılığıyla veritabanından siler ve işlem başarıyla tamamlandığında bir bildirim gösterir.

Bu yapı, kullanıcı etkileşimleriyle dinamik olarak güncellenen bir yorum listesi yönetim ekranını sağlar. Kullanıcılar, uygulama içindeki yorum verilerini kolaylıkla yönetebilir ve güncelleyebilirler.





Database.dart:

```
import 'dart:async';
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

class DatabaseHelper {
  static final DatabaseHelper _instance = DatabaseHelper.internal();

  factory DatabaseHelper() => _instance;

  static Database? _db;

  Future<Database?> get db async {
    if (_db != null) return _db;
    _db = await initDb();
    return _db;
  }

  DatabaseHelper.internal();

  Future<Database> initDb() async {
    String databasePath = await getDatabasesPath();
    String path = join(databasePath, 'app.db');
    var db = await openDatabase(path, version: 1, onCreate: _onCreate);
    return db;
  }

  void _onCreate(Database db, int newVersion) async {
    await db.execute(
        'CREATE TABLE Users (id INTEGER PRIMARY KEY, username TEXT, email
TEXT, password TEXT)');
    await db.execute(
        'CREATE TABLE Foods (id INTEGER PRIMARY KEY, food TEXT, rating
REAL)');
  }

  // User related methods
  Future<int> insertUser(Map<String, dynamic> row) async {
    Database? dbClient = await db;
    return await dbClient!.insert('Users', row);
  }

  Future<List<Map<String, dynamic>>> getAllUsers() async {
    Database? dbClient = await db;
    return await dbClient!.query('Users');
  }

  Future<Map<String, dynamic>?> getUser(String email, String password) async {
```

```
    Database? dbClient = await db;
    List<Map<String, dynamic>> users = await dbClient!.query(
      'Users',
      where: 'email = ? AND password = ?',
      whereArgs: [email, password],
    );
    if (users.length > 0) {
      return users.first;
    } else {
      return null;
    }
  }

  // Food related methods
  Future<int> insertFood(Map<String, dynamic> row) async {
    Database? dbClient = await db;
    return await dbClient!.insert('Foods', row);
  }

  Future<List<Map<String, dynamic>>> getAllFoods() async {
    Database? dbClient = await db;
    return await dbClient!.query('Foods');
  }

  Future<int> deleteFood(int id) async {
    Database? dbClient = await db;
    return await dbClient!.delete('Foods', where: 'id = ?', whereArgs: [id]);
  }
}
```

DatabaseHelper sınıfı, Flutter uygulamalarında SQLite veritabanıyla etkileşim sağlamak için kullanılır. Bu sınıf, uygulama içinde kullanıcı ve yorum verilerini yönetmek için gerekli veritabanı işlemlerini gerçekleştirir.

İlk olarak, sınıf DatabaseHelper.internal() yöntemiyle tek bir örnekleme oluşturur. Bu örnekleme factory yöntemiyle _instance üzerinden erişilebilir hale getirilir. _db değişkeni, SQLite veritabanı bağlantısını temsil eden bir Database nesnesini saklar.

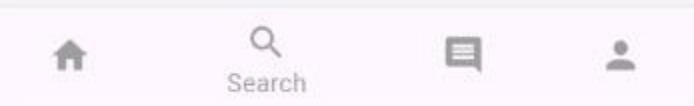
Veritabanı bağlantısı db yöntemiyle asenkron olarak alınır. _db null değilse, mevcut bağlantıyı döner. Eğer null ise, initDb yöntemi çağrılarak veritabanı dosyası oluşturulur veya varsa mevcut dosya açılır.

initDb yöntemi, getDatabasesPath ve join fonksiyonlarıyla cihazın yerel veritabanı dizinine (databasePath) ve app.db adlı dosyaya veritabanı yolunu oluşturur. openDatabase fonksiyonuyla da bu dosya açılır veya oluşturulur. onCreate parametresiyle veritabanı ilk oluşturulduğunda yapılacak işlemler belirtilir. Bu durumda Users ve Foods tabloları oluşturulur.

DatabaseHelper sınıfının insertUser, getAllUsers, ve getUser gibi yöntemleri, kullanıcı verilerini yönetir. insertUser yöntemi, Users tablosuna yeni bir kullanıcı ekler. getAllUsers yöntemi, Users tablosundaki tüm kullanıcıları listeler. getUser yöntemi ise verilen e-posta ve şifre ile eşleşen kullanıcıyı bulur.

Yemek verileriyle ilgili işlemler insertFood, getAllFoods, ve deleteFood yöntemleri aracılığıyla gerçekleştirilir. insertFood yöntemi, Foods tablosuna yeni bir yorum ekler. getAllFoods yöntemi, Foods tablosundaki tüm yorumları listeler. deleteFood yöntemi ise verilen bir yorum ID'sine göre yorumu Foods tablosundan siler.

Bu yapı, Flutter uygulamalarında yerel veritabanı kullanımını kolaylaştırarak, veri yönetimini güvenilir ve etkili bir şekilde sağlar. Kullanıcı ve yorum verileri gibi verilerin saklanması, güncellenmesi ve silinmesi gibi temel veritabanı işlemleri bu sınıf aracılığıyla yönetilir, böylece uygulamanın veri tabanlı işlevselliği sağlanır.



Navbar.dart:

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

class MyBottomNavigationBar extends StatelessWidget {
  final int currentIndex;
  final Function(int) onTap;

  const MyBottomNavigationBar({
    Key? key,
    required this.currentIndex,
    required this.onTap,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: BoxDecoration(
        color: Colors.white,
        boxShadow: [
          BoxShadow(
            blurRadius: 20,
            color: Colors.black.withOpacity(.1),
          )
        ],
      ),
      child: SafeArea(
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 15.0, vertical: 8),
```

```
          child: GNav(
            gap: 8,
            activeColor: Colors.grey,
            iconSize: 24,
            padding: EdgeInsets.symmetric(horizontal: 20, vertical: 5),
            duration: Duration(milliseconds: 400),
            tabBackgroundColor: Colors.grey.withOpacity(0.1),
            color: Colors.grey,
            selectedIndex: currentIndex,
            onTabChange: onTap,
            tabs: [
              GButton(
                icon: Icons.home,
                text: 'Home',
              ),
              GButton(
                icon: Icons.search,
                text: 'Search',
              ),
              GButton(
                icon: Icons.comment,
                text: 'Comment',
              ),
              GButton(
                icon: Icons.person,
                text: 'Profile',
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

MyBottomNavigationBar widget'i, Flutter'da alt navigasyon çubuğu oluşturmak için kullanılan özel bir widget'tır. Bu widget, Google'ın stilize edilmiş navigasyon çubuğunu kullanarak uygulamanın farklı bölümleri arasında geçiş yapmayı sağlar.

Widget, currentIndex ve onTap parametreleri alarak özelleştirilir. currentIndex, şu anda seçili olan sekmenin endeksini belirtirken, onTap ise kullanıcının bir sekme üzerine tıkladığında çağrılacak fonksiyonu işaret eder. Bu şekilde, kullanıcılar uygulamanın farklı ekranları arasında kolayca geçiş yapabilirler.

Görsel tasarımı ise şu özelliklere sahiptir: Container ile arka plan rengi ve gölge efekti ayarlanırken, SafeArea ve Padding kullanılarak içerik düzenlenir ve kenarlardan boşluk bırakılır. Ana bileşen olan GNav, Google Nav Bar için özelleştirilmiş bir widget olarak sekme özelliklerini ve görünümünü belirler. İkon boyutları, boşluklar, animasyon süresi gibi

özelliklerin yanı sıra, her bir sekme için ikon ve metin içeren GButton öğeleri tanımlanarak navigasyon çubuğu oluşturulur.

Bu yapı, uygulamanın kullanıcı arayüzünde hızlı ve etkili bir navigasyon sağlamak amacıyla kullanılır. Kullanıcıların ana ekranlar arasında gezinirken uygulamanın işlevselliğini kolayca keşfetmelerine yardımcı olur.

Pubspec.yaml:

```
name: myapp
description: "A new Flutter project."
# The following line prevents the package from being accidentally published to
# pub.dev using `flutter pub publish`. This is preferred for private packages.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as
versionCode.
# Read more about Android versioning at
https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-number
is used as CFBundleVersion.
# Read more about iOS versioning at
#
https://developer.apple.com/library/archive/documentation/General/Reference/In
foPlistKeyReference/Articles/CoreFoundationKeys.html
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the build
suffix.
version: 1.0.0+1

environment:
  sdk: '>=3.3.4 <4.0.0'

# Dependencies specify other packages that your package needs in order to
work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below
to
# the latest version available on pub.dev. To see which dependencies have
newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
```

```
    sdk: flutter


  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.6
  google_fonts: ^6.2.1
  sqflite: ^2.3.3+1
  hexcolor: ^3.0.1
  quickalert: ^1.1.0
  page_transition: ^2.1.0


dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended lints to
  # encourage good coding practices. The lint set provided by the package is
  # activated in the `analysis_options.yaml` file located at the root of your
  # package. See that file for information about deactivating specific lint
  # rules and activating additional ones.
  flutter_lints: ^3.0.0



flutter:
  uses-material-design: true

  assets:
    - lib/assets/depositphotos_190518818-stock-photo-food-cooking-ingredients-
background-on.png
    - lib/assets/breakfast.png
    - lib/assets/fried-rice.png
    - lib/assets/dinner.png
    - lib/assets/key.png
    - lib/assets/user.png
    - lib/assets/menemen.png
    - lib/assets/pankek.png
    - lib/assets/omlet.jpeg
    - lib/assets/decision.png
    - lib/assets/salata.jpeg
    - lib/assets/mantı.jpeg
    - lib/assets/karnabahar.jpeg
    - lib/assets/kofte.jpeg
    - lib/assets/makarna.jpeg
    - lib/assets/tavuk.jpeg
```

```
  # An image asset can refer to one or more resolution-specific "variants",
see
  # https://flutter.dev/assets-and-images/#resolution-aware

  # For details regarding adding assets from package dependencies, see
  # https://flutter.dev/assets-and-images/#from-packages

  # To add custom fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
  #       - asset: fonts/Schyler-Regular.ttf
  #       - asset: fonts/Schyler-Italic.ttf
  #         style: italic
  #   - family: Trajan Pro
  #     fonts:
  #       - asset: fonts/TrajanPro.ttf
  #       - asset: fonts/TrajanPro_Bold.ttf
  #         weight: 700
  #
  # For details regarding fonts from package dependencies,
  # see https://flutter.dev/custom-fonts/#from-packages
```

Package Linkleri:

https://pub.dev/packages/google_fonts/install
https://pub.dev/packages/hexcolor/install
https://pub.dev/packages/quickalert
https://pub.dev/packages/page_transition