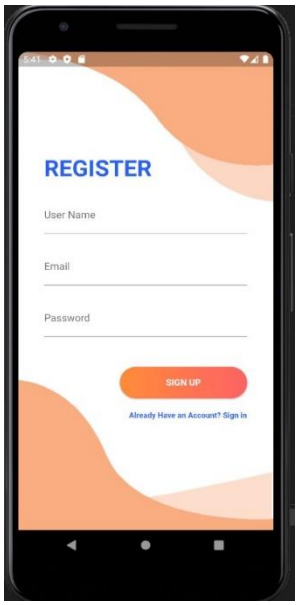
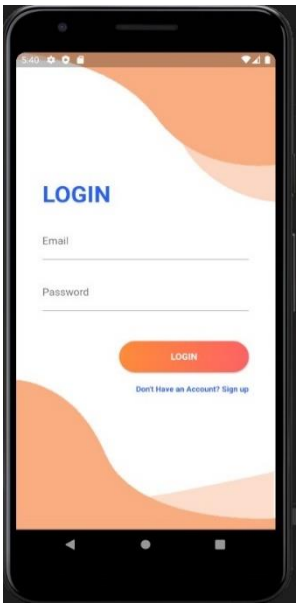
1.1. Proje Adımları ve Uygulanması

Uygulamanın ilk olarak Login ve Register sayfaları hazırlanmıştır. Register sayfasında Şekil 3.7'de görüldüğü üzere kullanıcıdan ad, email ve şifre bilgilerini istenmektedir.

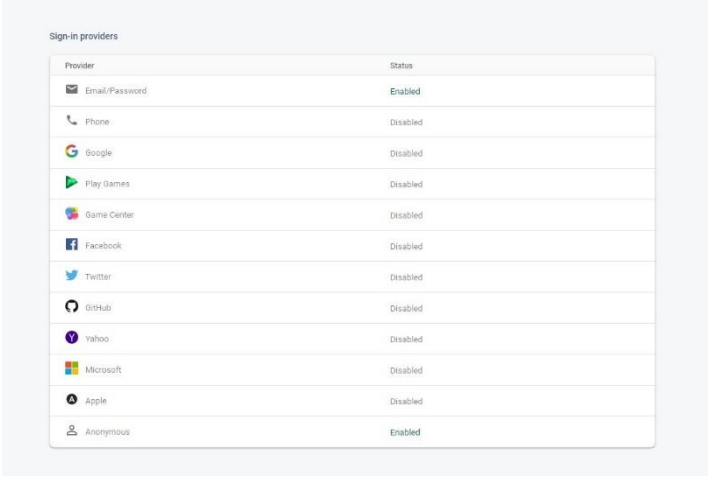


Şekil 3.7. RegisterScreen



Şekil 3.8. LoginScreen

Kullanıcı kayıt olduktan sonra Şekil 3.8.'de ki gibi Login sayfasına yönlendirilmektedir. Kullanıcı kaydı ve kallanıcı girişleri Firebase Authentication sayesinde sağlanmaktadır. Firebase Authentication'da Şekil 3.9.'da görüldüğü üzere birçok oturum açma metodu bulunmaktadır. Uygulama için Email/Password enable yapıldı.

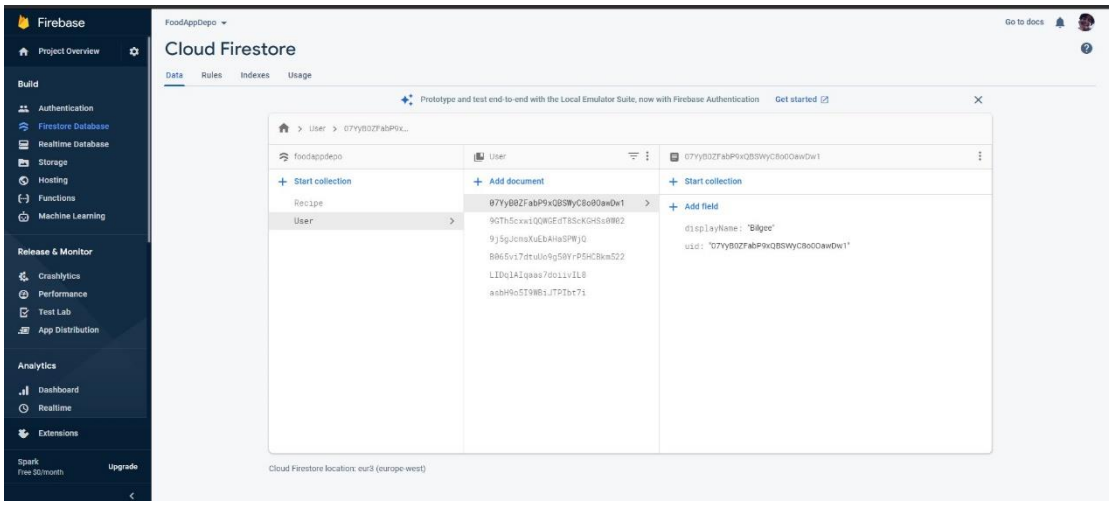


Şekil 3.9. Sign-In Providers

Kullanıcıların kayıt olma durumunun bilgisi Firebase Authentication üzerinden Şekil 3.10.'da görülmektedir.

Şekil 3.10. Kullanıcı Girişi

Kullanıcı adını ve uid bilgisini tutması için Firebase Cloud Firestore'dan yaralanıldı. Şekil 3.11' de görüldüğü gibi User adında collection oluşturuldu. Document bilgisini Firebase Authentication sayesinde kullanıcının user uidsi eklendi. Bu işlemin kodu Şekil 3.12' de gösterilmiştir.



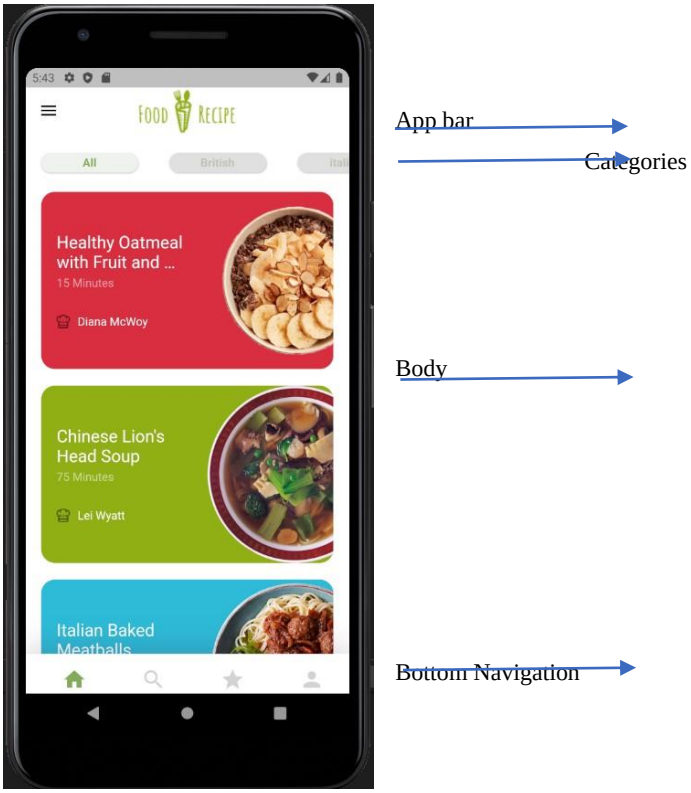
Şekil 3.11. User Collection

```
Future<void> userSetup(String displayName) async {
  CollectionReference users = FirebaseFirestore.instance.collection("User");
  FirebaseAuth auth = FirebaseAuth.instance;
  String uid = auth.currentUser.uid.toString();
  users.doc(uid).set({'displayName': displayName, 'uid': uid});
  return;
}
```

Şekil 3.12. UserSetup

Şekil 3.12.'deki kodda görüldüğü üzere UserSetup fonksiyonu Firebase Cloud Firestore'dan User collectiona erişti. Firebase Authentication sayesinde o anki kullanıcının uidi bilgisi alınıyor. User içerisinde documentte uid bilgisi girilerek kullanıcının displayName(kullanıcı adı) bilgisi alınıyor.

Kullanıcı uygulamaya giriş yaptıktan sonra Home sayfasına yönlendirilmektedir. Şekil 3.13'de Home sayfası mevcuttur.



Şekil 3.13. HomeScreen

Home sayfası dört bölümden oluşmaktadır. İlk kısım uygulamanın logosunun bulunduğu app bar bölümü, ikinci kısım yemek tariflerinin kültürlerine göre sıralamasını sağlayan categories bölümü, üçüncü kısım yemek tariflerinin listelendiği body bölümü ve son kısım sayafalar arası geçişi yapmamızı sağlayan bottom navigation bölümü mevcuttur. Flutterda widget mantığı kullanılmaktadır ve özel widget yazabilme olanağı bulunmaktadır. Home sayfasındaki dört bölüm custom widgettır.

Firebase Cloud Firestore'dan yemek tarifleri için Recipe adında collection oluşturulmuştur. Her document bir yemek tarifini bulundurmaktadır. Her yemek tarifinde aşağıdaki veriler mevcuttur.

- String id: İlgili tarifin idsi
- String chef : Aşçının ad bilgisi
- String description: Tarif bilgisi
- String image: Yemeğin fotoğrafı
- String ingredient: Yemek malzemeleri
- String name: Yemeğin ismi
- String time: Yemeğin aldığı vakit
- String type: Yemeğin hangi kültüre ait olduğu

```
class _RecipeDetailState extends State<RecipeDetail> {
  final _firestore = FirebaseFirestore.instance;

  String name = "";

  @override
  Widget build(BuildContext context) {
    CollectionReference recipes = _firestore.collection('Recipe');
```

Şekil 3.14. RecipeDetailHead

```
body: FutureBuilder<DocumentSnapshot>(
    future: recipes.doc(widget.id).get(),
    builder:
        (BuildContext context, AsyncSnapshot<DocumentSnapshot> snapshot) {
      Map<String, dynamic> data = snapshot.data.data();
      return SingleChildScrollView(
          child: Column(
        children: [
          Image.network(data.cast()['image']),
          Container(
            child: Column(
              children: [
                RecipeMenu(
                  iconSrc: "assets/images/cooking.png",
                  title: data['name'],
                ), // RecipeMenu
                RecipeMenu(
                  iconSrc: "assets/images/clock.png",
                  title: data['time'],
                ), // RecipeMenu
```

Şekil 3.15. RecipeDetailEnd

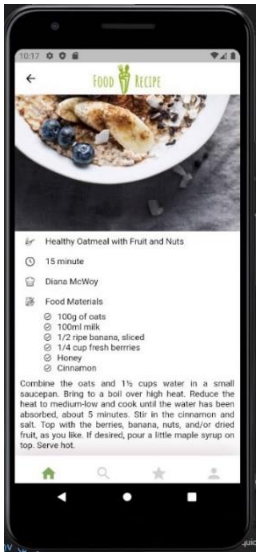
Yemek tariflerinin verileri RecipeDetailScreen sayfasında Firebase Cloud Firestore'dan alınmaktadır. İlgili kod Şekil 3.14. ve Şekil 3.15.'de gösterilmiştir. Firestoredan Recipe collectiona bağlanılmaktadır. Recipedetail classında String türünde id tanımlanılmıştır. Bu id, HomeScreen'de yemek kartlarına ait idler içermektedir. Örneğin HomeScreen'de listelenen yemek tariflerinden ilknin idsi "1" . Bu id bilgisi alınıp, o ilgili yemeğin ayrıntılarının listelendiği sayfaya aktarılır. Böylece id bilgisi alınır. İd bilgisi sayesinde o yemeğin verilerine erişilir. Futurebuilder [5] widgeti ile firestoredan alınan biligiler RecipeScreen'de istenilen yerlere yazılır. FutureBuilder asenkron çalışan bir widgettır. FutureBuilder içerisine 2 tane parametre alır. Bunlar:

FutureBuilder(BuildContext context, AsyncSnapshot snapshot){}

Context uygulamanın ekranında bulunan widgetların yerini belirleyen, widget Tree'de yer alan her elementin barındırdığı bir değişkendir. AsyncSnapshot ise yapacağımız future işleminin dönütünü tutacak değişkendir. Bu değişken, AsyncSnapshot'ı belirtmediğiniz sürece dynamic tiptedir. Değiştirmek isterseniz AsyncSnapshot<String> şeklinde değiştirebilirsiniz.
FutureBuilder ile veriyi, verilerin durumuna göre değişecek widgetları builder: içerisinde belirtmeli ve future: içerisinde network veya servis işlerini yapmalıdır. Firebaseden veri eçekerken kullanılan önemli bir widgettır.

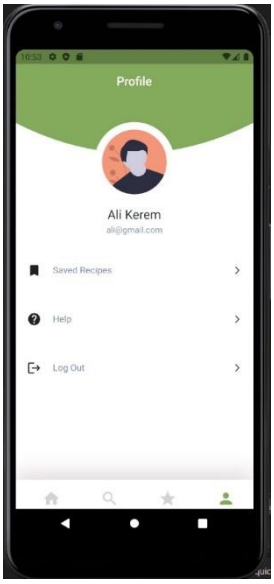
```
FutureBuilder(BuildContext context, AsyncSnapshot snapshot)
{
future: getDocuments(),
builder: ...
}
```

Kullaıncı, ana sayfada listenelen yemeklerden birine tıkladığında ilgili yemeğin ayrıntılarının gösterildiği sayfaya yönlendirilmektedir. RecipeDetailScreen'de ilgili yemeğin fotoğrafı, adı, tarifin kimin tarafından yazıldığı, malzemeleri ve tarif detayı mevcuttur. Şekil 3.16'da tarif detayı sayfası gösterilmiştir.



Şekil 3.16. RecipeDetailScreen

Kullanıcı adı ve email bilgisinin gösterildiği kullanıcı profili sayfası Şekil 3.17'de mevcuttur. Kullanıcı adı User collection sayesinde alınmaktadır. Email bilgisi Firebase Authentication özelliği sayseinde alınmaktadır.



Şekil 3.17. ProfileScreen

Kullanıcı ad bilgisinin alınmasına ait kod Şekil 3.18'de gösterilmiştir. Firebase User collectiona erişilmektedir. O anki kullanıcının uid bilgisi alınmaktadır. Uid biligileri aynı zamanda User collectionda document bilgileridir. Böylece uid ile istenilen kullanıcının bilgilerine erişilebilir. User collectionda o kişinin uidsi bulunup istenilen verileri profil sayfasına yazdırılmaktadır.

```
class _BodyState extends State<Body> {
  final _firestore = FirebaseFirestore.instance;
  FirebaseAuth auth = FirebaseAuth.instance;

  @override
  Widget build(BuildContext context) {
    CollectionReference user = _firestore.collection('User');
    String uid = auth.currentUser.uid.toString();
    return FutureBuilder<DocumentSnapshot>(
        future: user.doc(uid).get(),
        builder:
            (BuildContext context, AsyncSnapshot<DocumentSnapshot> snapshot) {
          Map<String, dynamic> data = snapshot.data.data();

          return SingleChildScrollView(
            child: Column(
              children: [
                Info(
                  image: "assets/images/pic.png",
                  name: data['displayName'],
                  email: auth.currentUser.email.toString(),
                ), // Info
```

Şekil 3.18. ProfileScreenBody