

**ANKARA ÜNİVERSİTESİ**  
**MÜHENDİSLİK FAKÜLTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**(BLM4538) IOS İle Mobil Uygulama Geliştirme II**

**Github:**

**[https://github.com/OmerFarukKaragoz/BLM4538-](https://github.com/OmerFarukKaragoz/BLM4538-21290585)**

**[21290585](https://github.com/OmerFarukKaragoz/BLM4538-21290585)**

**Video: <https://youtube.com/shorts/rrC-isTzHzs>**

**Ömer Faruk Karagöz**  
**21290585**

## 1. Recipe Recommender

Geliştirdiğim mobil uygulama, kullanıcıların çeşitli tariflere kolayca ulaşmasını, bu tarifleri filtreleyerek aramasını, tarifler arasında karşılaştırma yapmasını ve hem kendi değerlendirmelerini ekleyip hem de diğer kullanıcıların yorum ve puanlarını dikkate alarak en uygun tarife karar vermesini amaçlamaktadır.

Uygulamanın veri yapısı, Firebase üzerinde tutulmaktadır. Kullanıcı profilleri, favori tarifler, kayıt ve giriş işlemleri için gerekli e-posta ve şifre bilgileri ile tarif değerlendirme verileri bu veritabanında saklanmaktadır. Tarif içerikleri ise harici bir kaynak olan Spoonacular API aracılığıyla, API anahtarı kullanılarak elde edilmektedir.

Identifier	Providers	Created ↓	Signed In	User UID
asdasd@gmail.com	✉	Jan 14, 2025	Jan 14, 2025	jkh8KkUhGHeGFTdIEUiFvBYtc...
deve@gmail.com	✉	Jan 14, 2025	Jan 14, 2025	i9IZ2BGiZ3eZeCIaFImcFE1mJ...
gebze@gmail.com	✉	Jan 13, 2025	Jan 13, 2025	whnQ7gJIDFU2caT1uXQlOyD...
omer@gmail.com	✉	Jan 13, 2025	Jan 13, 2025	DALYKcOilecv8BhOdSweb0sd...

Şekil 1.1. Firebase Authentication yapısı

Şekil 1.1’de kullanıcı kaydı ve girişi için gerekli olan kullanıcı mailleri ve şifreleri Firebase Authentication’da depolanmaktadır. Kayıt sırasında mail ve şifre buradaki veri tabanına eklenir, giriş sırasında ise mail ve şifre buradaki veri tabanında sorgulanır.

ratings	DALYKcOilecv8BhOdSweb0sdZH33	+ Add field
users	YMqrHjkjDhZWSmuWcYuQrk1G0T32	age: 18
usersfavorites	i9IZ2BGiZ3eZeCIaFImcFE1mJop1	firstName: "FName"
	jkh8KkUhGHeGFTdIEUiFvBYtcBZ2	gender: "Gender"
	⋮ userId >	lastName: "LName"
	whnQ7gJIDFU2caT1uXQlOyDpUmD2	username: "UserName"

Şekil 1.2. Kullanıcı profilleri için gerekli Firestore database yapısı

Şekil 1.2’de kullanıcı profil bilgisi için gerekli olan Firestore database yapısı bulunmaktadır. Kullanıcıların profilleri “firstName”, “lastName”, “username”, “age” ve “gender” özelliklerinden oluşuyor. Kullanıcı Idsi ile eşleştirilerek depolanmaktadır.

ratings	DALYKcOilecv8BhOdSweb0sdZH33	+ Add field
users	YMqrHjkjDhZWSmuWcYuQrk1G0T32	favorites
usersfavorites	i9IZ2BGiZ3eZeCIaFImcFE1mJop1	0 "tarif id"
	jkh8KkUhGHeGFTdIEUiFvBYtcBZ2	
	userId >	

Şekil 1.3. Kullanıcı favorileri için gerekli Firestore database yapısı

Şekil 1.3’te kullanıcı favorileri için gerekli olan Firestore database yapısı bulunmaktadır. Favori tariflerin idleri, kullanıcı idleri ile eşleştirilerek depolanır (Kullanıcı idsine ait favorites listesinde tarif idleri depolanır).

ratings	640941	+ Add field
users	644387	▼ comments
usersfavorites	664147	0
	715415	▼ rate
	715446	0 2.5
	716627	▼ users
	782601	0
	recipeID	

Şekil 1.4. Tariflere ait değerlendirmeler için gerekli Firestore database yapısı

Şekil 1.4’te tarif değerlendirmeleri için gerekli Firestore database yapısı verilmektedir. Tarif idleri ile “comments”, “rate”, “users” listelerini eşleştirilir, daha sonrasında tarif değerlendirmeleri uygulama içinde veritabanından çekilirken aynı indekse ait yorum, değerlendirme ve kullanıcı bilgisini birleştirir (Bir kullanıcı değerlendirme eklediği zaman aynı anda “comments”, “rate”, “users” listelerine eleman eklendiği için indeksler her zaman eşleşir).

## 2. Uygulama İçi Ekranları

- Kullanıcı Kayıt Ekranı:

### Kayıt Ol

Email

Şifre

Ad

Soyad

Kullanıcı Adı

Yaş

Cinsiyet

Kayıt Ol

Zaten bir hesabınız var mı? Giriş yapın

Şekil 2.1. Kullanıcı Kayıt Ekranı

Şekil 2.1’de kullanıcı kayıt ekranı yapısı verilmiştir. Kayıt sırasında kullanıcıdan mail, şifre, ad, soyad, kullanıcı adı, yaş ve cinsiyet bilgileri istenilmektedir. Bu bölümler doldurulduktan sonra register butonuna basılarak Firebase Authentication’a kullanıcı eklenir (Eğer kullanıcı maili ile daha önceden bir hesap oluşturulmuşsa, uyarı gönderilir ve başka bir mail adresi kullanılması istenilir).

- Kullanıcı Giriş Ekranı:

## Tarif Uygulaması

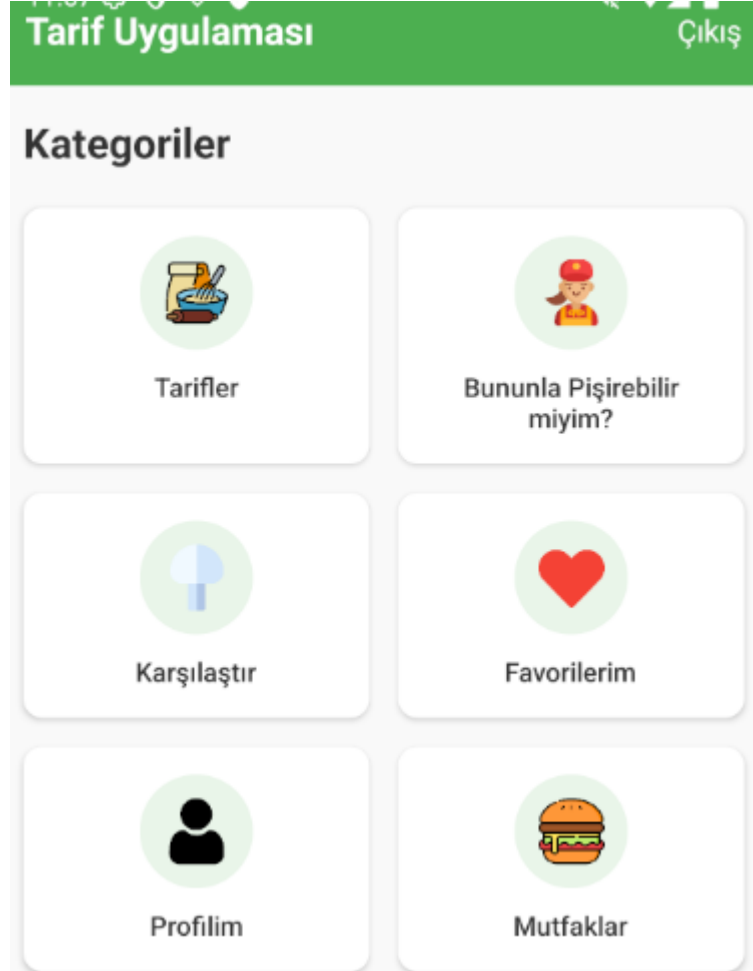
Giriş Yap

Hesabınız yok mu? Kaydolun

Şekil 2.2. Kullanıcı Giriş Ekranı

Şekil.2.2’de kullanıcı giriş ekranı yapısı verilmiştir. Giriş için kullanıcı mail ve şifresi istenilmektedir. Girilen mail ve şifre Firebase Authentication’da var ve eşleşiyorlar ise kullanıcı uygulamaya giriş yapabilir. Eğer mail ve şifre uyuşmuyorsa veya kayıt yapılmamış ise ekrana bir uyarı verilir. Login butonu ile uygulamaya giriş yapılır ve butonun altında bulunan yazıya tıklanarak ise kayıt ekranına gidilir.

- Uygulamanın Ana Ekranı:



Şekil 2.3. Uygulama Ana Ekranı

Şekil 2.3’te uygulamanın ana ekranının yapısı verilmiştir. Profil ikonuna basıldığında kullanıcı profil ekranına gidilir. Favoriler ikonuna basıldığında kullanıcının favori tariflerinin listelendiği favoriler ekranı açılır. Tarifler ikonuna basıldığında tariflerin filtrelenecek şekilde aratılabildiği tarif arama ekranı açılır. Bununla

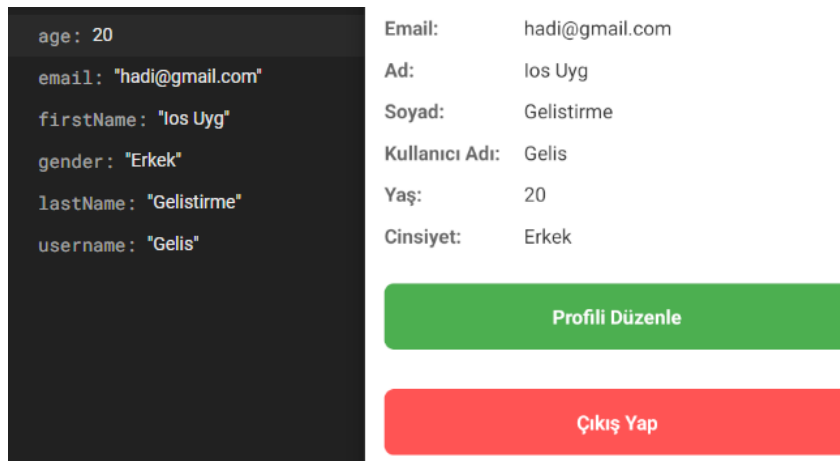
Pişirebilir miyim? ikonuna tıklanıldığı zaman eldeki malzemeler ile yapılabilen tariflerin listelendiği ekrana gidilir. Karşılaştır ikonuna basıldığı zaman tariflerin kıyaslandığı karşılaştırma ekranı açılır. Mutfaklar ikonuna ile tariflerin mutfaklarına göre sınıflandırıldığı mutfaklar ekranı açılır.

- Kullanıcı Profil Ekranı:



Şekil 2.3.1. Kullanıcı Profil Ekranı

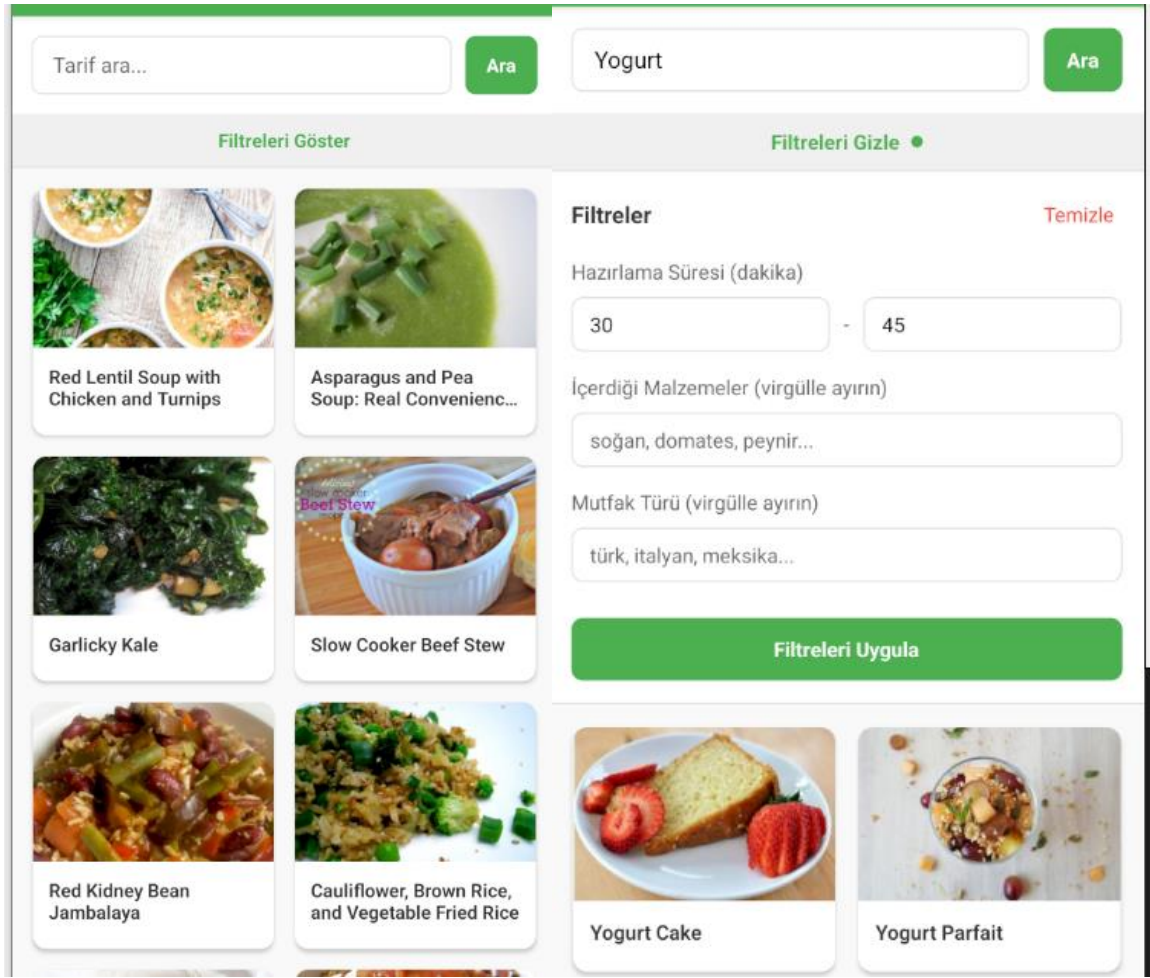
Şekil 2.3.1’de kullanıcı profil ekranı yapısı verilmiştir. Burada kullanıcı profil bilgileri Firestore Database’den çekilir ve bu bilgiler “Update Profile” butonu ile Şekil 2.3.2’deki gibi Firestore Database’de güncellenir.



Şekil 2.3.2 Kullanıcı Bilgilerinin FireStore Database’de güncellenmesi

- Tarifler Ekranı:

Şekil 2.4’te tarifler sayfasının başlangıç haliyle birlikte filtreler ve arama çubuğunun kullanılmış hali bulunmaktadır. Şeklin sol tarafındaki ekran, tarifler ekranının başlangıç halidir ve tarifleri filtrelemeden listeler. Sağ taraftaki ekranda ise filtrelerin uygulanması ve ”Yogurt” kelimesinin aratılması ile tarifler filtrelenerek listelenir.



Şekil 2.4. Tarifler Ekranı


- Tarif Detayı Ekranı:

Tarif Detay Ekranı, 3 yapıyı içerir. Bunlar sırası ile tarifin görseli ve bilgileri içeren “Tarif bilgileri” kısmı, kullanıcının favori tariflerine ekleme ve çıkarma yapmayı sağlayan “Favorilere Ekle” butonu ve kullanıcıların yorum yapmasını ve diğer kullanıcıların yorumlarının listelendiği “Yorumlar” kısmı.

Şekil 2.5.1’de Tarif Detay Ekranın “Tarif Bilgileri” kısmı verilmiştir. Burada tarif görseli, mutfak, tarif için gerekli süre, gerekli malzemeler ve adımlar spoonacular API’den API key yardımı ile çekilerek listelenir.

Şekil 2.5.2’de Tarif Detay Ekranın “Favori” ve “Yorum” kısımları verilmiştir. Favori butonu ile Firestore Database’deki kullanıcının favori tarifler listesine ekleme ve çıkarma yapılır. Hemen altında bulunan “Yorumla” kısmında ise yorum ve değerlendirme Firestore Database’ine eklenir ve Database’deki önceki yorumlar kullanıcı adı, yorumu ve puanı ile birlikte listelenir.

[←](#) Tarif Detayı



## Red Lentil Soup with Chicken and Turnips

Hazırlama Süresi: 55 dakika

### Malzemeler

- additional toppings: diced avocado, micro greens, chopped basil)
- 3 medium carrots, peeled and diced
- 3 celery stalks, diced
- 2 cups fully-cooked chicken breast, shredded (may be omitted for a vegetarian version)
- ½ cup flat leaf Italian parsley, chopped (plus extra for garnish)
- 6 cloves of garlic, finely minced
- 2 tablespoons olive oil
- 28 ounce-can plum tomatoes, drained and rinsed, chopped
- 2 cups dried red lentils, rinsed
- salt and black pepper, to taste
- 1 large turnip, peeled and diced

Şekil 2.5.1. Tarif Bilgileri Kısmı

### Yorumlar

Favorilere Ekle

Puanınız: ★ ★ ★ ★ ★

Yorum yazın...

Yorum Ekle

- Unknown User

★★★★★

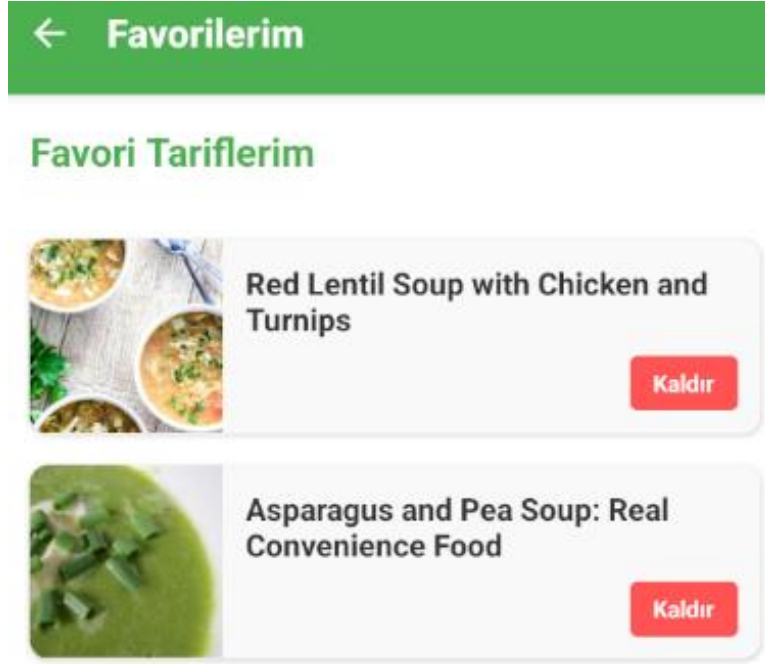
deneme
- 13123

★★★★★

AAAAAAAAAAAAAAAAAAAA

Şekil 2.5.2. Favori ve Yorum Kısımları

- Favoriler Ekranı:



Şekil 2.6 Favoriler Ekranı

Şekil 2.6’da favoriler ekranının yapısı verilmiştir. Bu ekranda kullanıcının favori tariflerinin idleri Firestore Database’den elde edilir ve bu idler ile eşleşen spoonacular API tarifleri listelenir.

- Karşılaştırma Ekranı:

Şekil 2.7.1’de karşılaştırma ekranının yapısı verilmiştir. Bu ekranda tarifler arasından ikisi seçilir ve “Karşılaştır” butonuna basılarak o iki tarif karşılaştırılır. İki tarifte, besin değerleri karşılaştırılır ve iki tarifin bilgilerine erişilebilecek tarif kartları altta görüntülerinir ve böylece istenilen tarifin adımları, malzemeleri, mutfağı, gerekli süresi ve yorumlarına kolayca erişilebilir.

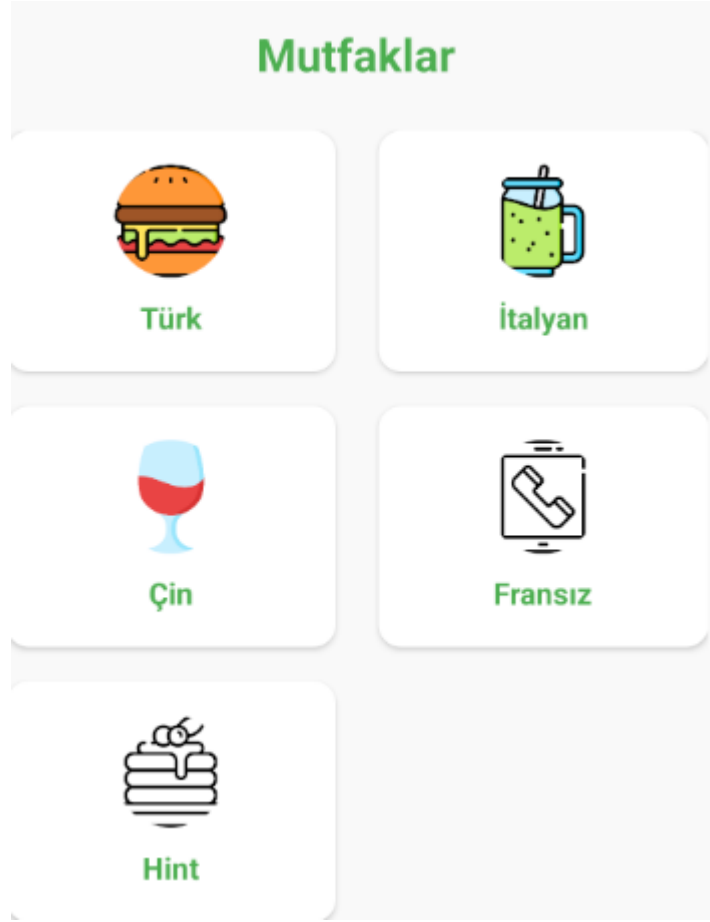


Şekil 2.7.1. Karşılaştırma Ekranı

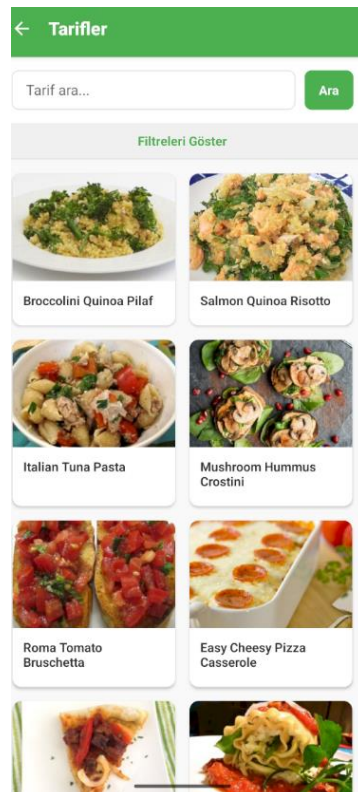


- Mutfaklar Ekranı:

Şekil 2.8.1’de mutfaklar ekranının yapısı verilmiştir. Bu ekranın amacı spoonacular API’den tarifleri mutfaklara göre filtreleyerek listelemek. Bu ekranda gösterilen mutfaklardan biri seçilerek o mutfağa ait tarifler listelenir. Şekil 2.8.2’de seçilen mutfağın tariflerinin listelenme yapısı verilmiştir. Şekilde İtalyan mutfağı seçilmiş ve İtalyan tarifler listelenmiştir. Burada filtrelemeler ve aramalar yapılabilir ama verilen tüm tarifler İtalyan mutfağına aittir.



Şekil 2.8.1. Mutfaklar Ekranı



Şekil 2.8.2. İtalyan mutfağına ait tarifler

- Bununla Ne Pişirebilirim? Ekranı:

Şekil 2.9’da “Bununla Ne Pişirebilirim?” ekranının yapısı verilmiştir. Bu ekranın amacı kullanıcının elinde bulunan malzemeler ile tüm malzemeleri karşılanan tariflerin listelenmesidir. Fakat spoonacular API’de böyle bir sorguyu sağlayan yapı olmadığı için bu özellik doğru çalışmamakta. Kendi oluşturacağım algoritma ise çok uzun bir sorguya sebep olacağı için hem API hakkımı bitirebileceği hemde performans açısından kullanıcıya kötü bir deneyim sunacağı için uygulamadaki amacı değiştirilmiştir. Uygulamadaki hali girilen malzemeleri içeren tarifleri sıralamaktadır.

← Bununla Ne Pişirebilirim?

## Bununla Ne Pişirebilirim?

Elinizdeki malzemeleri virgülle ayırarak girin

Ara

Vietnamese Pancakes with Vegetables, Herbs and a Fragrant Dipping Sauce (Bánh Xèo)

Poached Egg With Spinach and Tomato

Pad Se Ew Tofu With Vegetable Noodles

Open-Face Egg Sandwich with Bacon, Asparagus, and Pesto

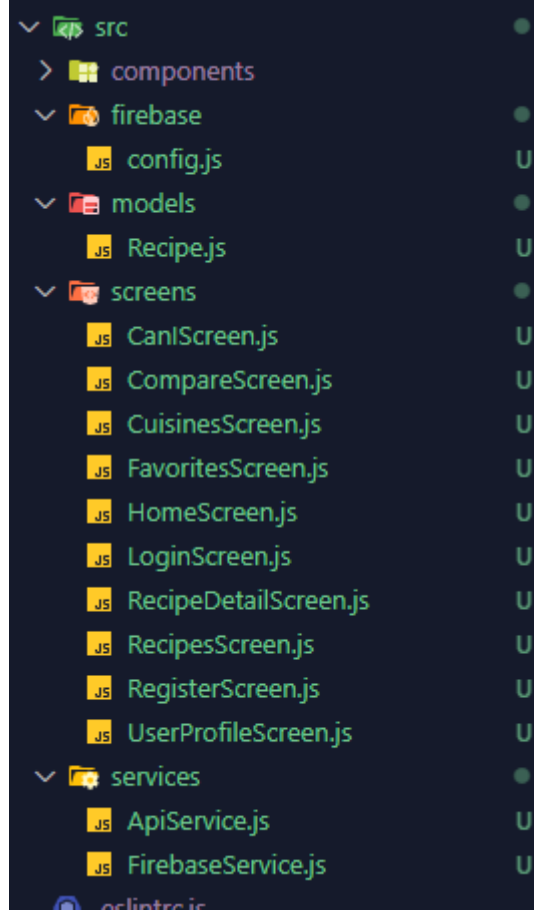
Asian Noodles

Kona Pork with Pineapple Slaw

Şekil 2.9. Bununla Ne Pişirebilirim? Ekranı

### 3. Projenin Kod Yapısı ve İşlevleri

Projenin kod yapısı ve işlevleri bu kısımda anlatılacaktır.



Şekil 3.1. Dosya Yapısı

Projenin dosyaları şekil 3.1’de gösterildiği formattadır. “models” klasörü modülleri oluşturmak için kullanılan dosyaları bulundurur. “screens” klasörü uygulama içinde değişen ekranların (profil ekranı, favoriler ekranı vs.) yönetimi ve oluşturulması için gerekli dosyaları bulundurur. “services” klasörü Firebase’den veri alma ve gönderme, spoonacular API’den api key ile tarifler ve bilgilerini almak gibi veri alış verişini sağlayan dosyaları içerir.

#### config.cs ve FirebaseService.js:

Burada Firebase uygulaması için platforma özgü yapılandırma seçeneklerini sağlayan bir sınıf oluşturur. Farklı platformlar için Firebase API anahtarları, uygulama kimlikleri, proje kimliği, depolama alanı bağlantıları gibi temel yapılandırma bilgilerini içerir.

```
const firebaseConfig = {
  apiKey: "AIzaSyAHhDTPT5Y6l1yie3ZSoQVvAqvIgBG_DCQ",
  appId: "1:566281682384:android:e5dbed137220da4fb60dd5",
  messagingSenderId: "566281682384",
  projectId: "cumagecedeneme",
  storageBucket: "cumagecedeneme.firebaseio.com",
  authDomain: "cumagecedeneme.firebaseio.com",
  databaseURL: "https://cumagecedeneme.firebaseio.com"
};
```

Şekil 3.2. Android Yapılandırması

Burada Firebase Authentication ve Firestore Database kullanarak kullanıcı yönetimi ve yorum işlemleri sağlayan bir hizmet sınıfı olan `FirestoreService`'i tanımlanır. `registerUser`, yeni bir kullanıcı kaydeder ve kullanıcı bilgilerini Firestore'daki `users` koleksiyonunda saklar. `addComment`, belirli bir tarif için yorum ve derecelendirme ekler. `getComments`, bir tarif için tüm kullanıcı yorumlarını ve derecelendirmelerini alır ve kullanıcı kimliklerinden kullanıcı adlarını bulmak için `getUsernameById` yöntemini kullanır. `addFavorite` ve `removeFavorite` ise tariflerin Firestore database'deki favoriler database'ine ekleme ve çıkarma yapmayı sağlar. `loginUser`, bir kullanıcının Firebase Authentication ile oturum açmasını sağlar, `logoutUser` oturumu sonlandırır. Bu sınıf, kullanıcı ve yorum yönetimini kolaylaştırmak için API sağlar.

```
async addComment(recipeId, userId, rating, comment) {
  try {
    const docRef = firestore().collection('ratings').doc(recipeId);
    const snapshot = await docRef.get();

    if (snapshot.exists) {
      await docRef.update({
        users: firestore.FieldValue.arrayUnion(userId),
        rate: firestore.FieldValue.arrayUnion(rating),
        comments: firestore.FieldValue.arrayUnion(comment),
      });
    } else {
      await docRef.set({
        users: [userId],
        rate: [rating],
        comments: [comment],
      });
    }
  }
}

async getUsernameById(userId) {
  try {
    const docRef = firestore().collection('users').doc(userId);
    const snapshot = await docRef.get();

    if (snapshot.exists) {
      const data = snapshot.data();
      const username = data.username || 'Unknown User';
      return username;
    } else {
      return null;
    }
  } catch (error) {
    console.error('Error getting username: ', error);
    return null;
  }
}

async getComments(recipeId) {
  try {
    const docRef = firestore().collection('ratings').doc(recipeId);
    const snapshot = await docRef.get();

    if (snapshot.exists) {
      const data = snapshot.data();
      const users = data.users || [];
      const comments = data.comments || [];
      const ratings = data.rate || [];

      const result = [];

      for (let i = 0; i < users.length; i++) {
        const userId = users[i];
        const username = await this.getUsernameById(userId);

        result.push({
          username: username || 'Unknown User',
          comment: i < comments.length ? comments[i] : null,
          rating: i < ratings.length ? ratings[i] : null,
        });
      }

      return result;
    } else {
      return [];
    }
  } catch (error) {
    console.error('Error getting comments: ', error);
    return [];
  }
}
```

Şekil 3.3. `addComment` ve `getComments` Methodları

Şekil 3.3'de yorum ve değerlendirmelerin gönderilmesi ve getirilmesi için gerekli methodların kodları verilmiştir.

```
async loginUser(email, password) {
  try {
    await auth().signInWithEmailAndPassword(email, password);
    return "success";
  } catch (error) {
    return error.message;
  }
}

async logoutUser() {
  try {
    await auth().signOut();
  } catch (error) {
    console.error("Error signing out: ", error);
  }
}

getCurrentUser() {
  return auth().currentUser;
}

async registerUser(email, password, name, surname, username, age, gender) {
  try {
    const userCredential = await auth().createUserWithEmailAndPassword(email, password);
    await firestore().collection('users').doc(userCredential.user.uid).set({
      email: email,
      firstName: name,
      lastName: surname,
      username: username,
      age: age,
      gender: gender,
    });
    return "success";
  } catch (error) {
    return error.message;
  }
}
```

Şekil 3.4. `loginUser` `logoutUser` ve Diğer User Methodları

Şekil 3.4'de kullanıcı kaydı, girişi ve çıkışını sağlayan methodların kodları verilmiştir.

## Recipe.js:

Burada bir yemek tarifi modelini temsil eden bir JS sınıfıdır ve tarifi kimliği, başlığı, görseli, hazırlanma süresi, mutfak türleri, malzemeler, talimatlar ve besin değerleri gibi özelliklerini içerir. "fromJson", bir JSON nesnesini alıp bir tarif nesnesine dönüştürür.

```

class Recipe {
  constructor(id, title, image, readyInMinutes, cuisines, ingredients, instructions, nutrition) {
    this.id = id;
    this.title = title;
    this.image = image;
    this.readyInMinutes = readyInMinutes;
    this.cuisines = cuisines;
    this.ingredients = ingredients;
    this.instructions = instructions;
    this.nutrition = nutrition;
  }

  static fromJson(json) {
    return new Recipe(
      json.id || 0,
      json.title || "Unknown",
      json.image || "",
      json.readyInMinutes,
      json.cuisines || [],
      (json.extendedIngredients || []).
        .map(ingredient => ingredient.original?.toString() || ""),
      json.instructions || "No instructions available.",
      json.nutrition || {}
    );
  }
}

```

Şekil 3.5. Recipe.js Kod Yapısı

### ApiService.js:

Burada yemek tarifi verilerini almak için bir ApiService tanımlanır ve Spoonacular API'sini kullanılır. fetchRecipes methodu, arama sorgusu ve isteğe bağlı filtreler alarak tariflerin bir listesini getirir; API çağrısında, filtreler sorgu parametrelerine dönüştürülür. fetchRecipeDetails methodu ise belirli bir tarifi detaylarını, besin bilgileri dahil, tarif kimliğine göre API'den alır. Her iki yöntem de HTTP GET istekleri yapar, yanıtları JSON formatında çözümler. Daha sonrasında bu JSON formatı “recipe.dart” tarafından dönüştürülür.

```

class ApiService {
  constructor() {
    this.apiKey = '59667d0b94dd461eb02a0110a3b87d4a';
  }

  async fetchRecipes(query = '', filters = null) {
    let filterString = '';

    if (filters) {
      Object.entries(filters).forEach(([key, value]) => {
        filterString += `&${key}=${value}`;
      });
    }

    const url = `https://api.spoonacular.com/recipes/complexSearch?apiKey=${this.apiKey}&query=${query}${filterString}`;

    try {
      const response = await axios.get(url);
      return response.data.results;
    } catch (error) {
      console.error('API Error:', error);
      throw new Error('Failed to load recipes');
    }
  }

  async fetchRecipeDetails(recipeId) {
    const url = `https://api.spoonacular.com/recipes/${recipeId}/information?includeNutrition=true&apiKey=${this.apiKey}`;

    try {
      const response = await axios.get(url);
      return response.data;
    } catch (error) {
      console.error('API Error:', error);
      throw new Error('Failed to load recipe details');
    }
  }
}

```

Şekil 3.6. ApiService.js kod yapısı

### CanIScreen.js:

Burada yapılmak istenilen girilen eldeki malzemeler ile malzemeleri tamamen karşılanan tariflerin listelenmesinin sağlanması ancak spoonacular API'de bunu destekleyen bir yapı olmadığı ve kendi oluşturabileceğim sorgu çok maliyetli ve performans açısından zararlı olduğu için buradaki özellik eldeki malzemeleri içeren tarifleri listeleyecek şekilde değiştirildi.

### CompareScreen.js:

Burada, kullanıcıların iki tarif seçip karşılaştırmalarına olanak tanınır. CompareScreen, tarifleri çeker ve her bir tarifi üzerine tıklayarak seçilmelerini sağlar. Kullanıcı iki tarif seçtikten sonra, "Karşılaştır" butonuna tıklayarak bu tariflerin detaylı karşılaştırmasını yapabileceği bir ekrana yönlendirilir. Bu karşılaştırma işlemi sırasında her iki tarifi özellikleri detaylı bir şekilde karşılaştırılır ve ekranın alt kısmında besin değerleri de karşılaştırılır.

Şekil 3.7'de iki tarifi özelliklerinin karşılaştırılması verilmiştir.

```
// Kalori, yağ, karbonhidrat için düşük olan daha iyidir
if (nutrient === 'Calories' || nutrient === 'Fat' || nutrient === 'Carbohydrates') {
  return {
    better: value1 < value2 ? 1 : value1 > value2 ? 2 : 0,
    difference: Math.abs(value1 - value2)
  };
}
// Protein için yüksek olan daha iyidir
else if (nutrient === 'Protein') {
  return {
    better: value1 > value2 ? 1 : value1 < value2 ? 2 : 0,
    difference: Math.abs(value1 - value2)
  };
}

return { better: null, difference: 0 };
}
```

Şekil 3.7. Özelliklerin Karşılaştırılması

### CuisinesScreen.js:

Burada kullanıcılara çeşitli mutfakları listeleyen bir ekran sunulur. CuisineListScreen, bir dizi mutfak türü ile her biri bir mutfak ismini içeren bir liste oluşturur. Her bir mutfak ismine tıklandığında, o mutfak türüne ait tariflerin gösterildiği ekrana yönlendirilir.

```
const CUISINES = [
  { name: 'Türk', image: 'https://cdn-icons-png.flaticon.com/512/3075/3075977.png', api: 'Turkish' },
  { name: 'İtalyan', image: 'https://cdn-icons-png.flaticon.com/512/3075/3075972.png', api: 'Italian' },
  { name: 'Çin', image: 'https://cdn-icons-png.flaticon.com/512/3075/3075975.png', api: 'Chinese' },
  { name: 'Fransız', image: 'https://cdn-icons-png.flaticon.com/512/3075/3075973.png', api: 'French' },
  { name: 'Hint', image: 'https://cdn-icons-png.flaticon.com/512/3075/3075974.png', api: 'Indian' },
];

const CuisinesScreen = ({ navigation }) => {
  return (
    <View style={styles.container}>
      <Text style={styles.title}>Mutfaklar</Text>
      <ScrollView contentContainerStyle={styles.cuisineList}>
        {CUISINES.map((cuisine, idx) => {
          <TouchableOpacity
            key={idx}
            style={styles.cuisineItem}
            onPress={() => navigation.navigate('Recipes', { cuisine: cuisine.api })}
          >
            <Image source={{ uri: cuisine.image }} style={styles.cuisineImage} />
            <Text style={styles.cuisineText}>{cuisine.name}</Text>
          </TouchableOpacity>
        })}
      </ScrollView>
    </View>
  );
};
```

Şekil 3.8. CuisinesScreen.js Kod Yapısı

### FavoritesScreen.js:

Burada kullanıcıların favori tariflerini Firebase üzerinden çekip görüntüleyen bir ekran sağlar. FavoritesScreen ekranı, kullanıcının Firebase Authentication ile giriş yapmış kullanıcıya göre, o kullanıcının favori tariflerini Firestore'dan alır. fetchFavorites fonksiyonu, kullanıcının favori tariflerini içeren usersfavorites koleksiyonundan ilgili veriyi çeker, tariflerin detaylarını API aracılığıyla alır ve bu tarifleri liste haline getirir.

Şekil 3.9'de Firestore'dan favorilerin alınması gösterilmiştir.

```
const fetchFavorites = async () => {
  setLoading(true);
  const currentUser = FirebaseService.getCurrentUser();

  if (!currentUser) {
    setLoading(false);
    return;
  }

  try {
    const doc = await firestore()
      .collection('usersfavorites')
      .doc(currentUser.uid)
      .get();

    if (doc.exists) {
      const favoritesData = doc.data().favorites || [];
      setFavorites(favoritesData);
      // Tarif başlıklarını çek
      const recipes = await Promise.all(
        favoritesData.map(async (id) => {
          try {
            const data = await ApiService.fetchRecipeDetails(id);
            return { id, title: data.title };
          } catch {
            return { id, title: 'Başlık bulunamadı' };
          }
        })
      );
      setFavoriteRecipes(recipes);
    }
  }
}
```

Şekil 3.9. Kullanıcının Favorilerinin Firestore'dan Alınması

### HomeScreen.js:

Burada HomeScreen ekranı, kullanıcıya altı farklı ikon sunar: "Tarifler", "Bununla Pişirebilir miyim?", "Karşılaştır", "Favorilerim", "Profilim" ve "Mutfaklar". Bu ikonlar, sırasıyla tarifler, malzemelerle yemek bulma, tarif karşılaştırma, favorileri listeleme, profili görüntüleme ve mutfak türleri sayfalarına yönlendirir.

### LoginScreen.js:

Burada kullanıcıların giriş yapabileceği bir login ekranı sunulur. LoginScreen ekranı, kullanıcıdan e-posta ve şifre bilgilerini alır. Kullanıcı bilgilerini doğrulamak için FirebaseService'den loginUser fonksiyonu çağrılır. Eğer giriş başarılı olursa, kullanıcı ana sayfaya yönlendirilir, aksi takdirde hata mesajı ile kullanıcıya bildirilir. Ayrıca, eğer kullanıcı kaydolmamışsa, kayıt sayfasına yönlendiren bir "Hesabınız yok mu? Kaydolun" butonu bulunur. Bu ekran, kullanıcı arayüzüne giriş yapmak için gerekli tüm alanları ve doğrulama işlemlerini içerir.



```
import FirebaseService from '../services/FirebaseService';

const LoginScreen = ({ navigation }) => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleLogin = async () => {
    if (!email || !password) {
      Alert.alert('Hata', 'Email ve şifre gereklidir');
      return;
    }

    const result = await FirebaseService.loginUser(email.trim(), password.trim());

    if (result === 'success') {
      navigation.replace('Home');
    } else {
      Alert.alert('Giriş Hatası', result);
    }
  };
};
```

Şekil 3.10. Firebase'deki Veriler Kontrol Ederek Giriş Sağlayan Yapı

### RegisterScreen.js:

Burada kullanıcıların yeni bir hesap oluşturabileceği bir kayıt ekranı sunulur. RegisterScreen ekranı, kullanıcıdan e-posta, şifre, ad, soyad, kullanıcı adı, yaş ve cinsiyet gibi bilgileri alır. Kullanıcı, bu bilgileri girdikten sonra FirebaseService aracılığıyla kayıt işlemi yapılır. registerUser fonksiyonu çağrılır ve kayıt işlemi başarılı olursa kullanıcı login ekranına yönlendirilir. Eğer işlem başarısız olursa, kullanıcıya bir hata mesajı gösterilir.

```
const RegisterScreen = ({ navigation }) => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [name, setName] = useState('');
  const [surname, setSurname] = useState('');
  const [username, setUsername] = useState('');
  const [age, setAge] = useState('');
  const [gender, setGender] = useState('');

  const handleRegister = async () => {
    if (!email || !password || !name || !surname || !username || !age || !gender) {
      Alert.alert('Hata', 'Tüm alanlar doldurulmalıdır');
      return;
    }

    const result = await FirebaseService.registerUser(
      email.trim(),
      password.trim(),
      name.trim(),
      surname.trim(),
      username.trim(),
      parseInt(age),
      gender.trim()
    );

    if (result === 'success') {
      Alert.alert('Başarılı', 'Kayıt başarılı, giriş yapabilirsiniz', [
        {
          text: 'Tamam',
          onPress: () => navigation.navigate('Login')
        }
      ]);
    } else {
      Alert.alert('Kayıt Hatası', result);
    }
  };
};
```

Şekil 3.11. Kayıt bilgilerinin alınması ve Firebase'e gönderilmesi



### UserProfileScreen.js:

Burada kullanıcıların profil bilgilerini görüntüleyip düzenleyebileceği bir ekran sağlanır. UserProfileScreen ekranı, kullanıcıya ad, soyad, kullanıcı adı, yaş ve cinsiyet gibi bilgileri görüntüler ve düzenlemelerine olanak tanır. Bu bilgiler Firebase Firestore'dan alınır ve kullanıcı Firebase Authentication ile giriş yaptıysa, veriler ilgili kullanıcıya ait olan Firestore dokümanından yüklenir. Kullanıcı, bilgilerini güncelledikten sonra "Profilimi Güncelle" butonuna tıklayarak verilerini güncelleyebilir. updateUserProfile fonksiyonu, Firestore veritabanındaki mevcut kullanıcı verilerini günceller.

```
const fetchUserData = async () => {
  setLoading(true);
  const currentUser = FirebaseAuthService.getCurrentUser();

  if (!currentUser) {
    navigation.navigate('Login');
    return;
  }

  try {
    const userDoc = await firestore()
      .collection('users')
      .doc(currentUser.uid)
      .get();

    if (userDoc.exists) {
      const data = userDoc.data();
      setUserData(data);

      // Form verilerini doldur
      setFirstName(data.firstName || '');
      setLastName(data.lastName || '');
      setUsername(data.username || '');
      setAge(data.age ? data.age.toString() : '');
      setGender(data.gender || '');
    }
  } catch (error) {
    console.error('Error fetching user data:', error);
    Alert.alert('Hata', 'Kullanıcı bilgileri yüklenirken bir sorun oluştu');
  } finally {
    setLoading(false);
  }
};
```

Şekil 3.12. Kullanıcı Bilgilerinin Firestore'dan Alınması

```
// Profil bilgilerini güncelle
setUserData({
  ...userData,
  firstName: firstName.trim(),
  lastName: lastName.trim(),
  username: username.trim(),
  age: age ? parseInt(age) : null,
  gender: gender.trim(),
});

setEditing(false);
Alert.alert('Başarılı', 'Profil bilgileriniz güncellendi');
} catch (error) {
  console.error('Error updating profile:', error);
  Alert.alert('Hata', 'Profil güncellenirken bir sorun oluştu');
}
```

Şekil 3.13 Kullanıcı Bilgilerinin Firestore'da güncellenmesi

### RecipesScreen.js:

Burada kullanıcıların tarifleri arayıp filtreleyebileceği bir ekran oluşturulur. RecipesScreen ekranı, kullanıcının tarifleri aramak ve çeşitli filtreler uygulamak için metin giriş alanları sunar. Kullanıcı, filtreleri girdikten sonra "Filtreleri Uygula" butonuna tıklayarak filtreleri uygular. Filtreler, API üzerinden tarifleri çekerken kullanılır. ApiService sınıfı, tarife dair verileri alırken, fetchRecipes fonksiyonu arama sorgusu ve filtre parametreleri ile tarifleri getirir. Kullanıcı, listeye tıklayarak tarifin detaylarına geçiş yapabilir. Veri çekme işlemi sırasında bir yüklenme göstergesi görüntülenir.

```
// Uygulanan filtreler
if (filtersApplied) {
  // Mutfak filtreleri
  if (selectedCuisines.trim()) {
    filters.cuisine = selectedCuisines.split(',').map(c => c.trim()).join(',');
  }

  // Malzeme filtreleri
  if (includedIngredients.trim()) {
    filters.includeIngredients = includedIngredients.split(',').map(i => i.trim()).join(',');
  }

  // Süre filtreleri
  if (minTime.trim()) {
    filters.minReadyTime = parseInt(minTime.trim());
  }

  if (maxTime.trim()) {
    filters.maxReadyTime = parseInt(maxTime.trim());
  }
}
```

Şekil 3.14. Filtrelerin Kontrolü

```
const recipeData = await ApiService.fetchRecipes(query, Object.keys(filters).length ? filters : null);
setRecipes(recipeData);
} catch (error) {
  setErrorMessage('Tarifler yüklenirken bir hata oluştu');
  console.error(error);
} finally {
  setLoading(false);
}
};
```

Şekil 3.15. Filtreler ile Birlikte Tariflerin Listelenmesi

### RecipeDetailScreen.js:

Burada bir tarife ait detayları gösteren bir ekran oluşturulur ve kullanıcıların tarifleri favorilerine ekleyip yorum yapmalarını sağlar. RecipeDetailScreen ekranı, tarifin detaylarını API'den çeker ve kullanıcıya tarifin başlığı, görüntüsü, malzemeleri, talimatları gibi bilgileri sunar. Kullanıcı, bir yorumu yazabilir ve puanlama yapabilir. Ayrıca, "Favorilere Ekle" butonu ile, kullanıcının tarifi favorilerine eklemesini veya çıkarmasını sağlar. Firebase üzerinden kullanıcının favorilerine dair veriler alınarak, favori durumu kontrol edilir ve butona tıklanarak favorilere eklenir veya çıkarılır. Yorumlar ve değerlendirmeler, Firestore database'den çekilir ve ekranda listelenir. Bu ekran, kullanıcı etkileşimini sağlayarak, tarifler hakkında yorum yapılmasını ve favori eklenmesini mümkün kılar.