🏗️ BÖLÜM 1: Firebase Nedir? (Mutfaktaki Görünmez Aşçı)

Bir mobil uygulama geliştirdiğinizi hayal edin. Bu uygulamanın iki ana kısmı vardır:

1. **Frontend (Ön Yüz):** Kullanıcının gördüğü butonlar, renkler, yazılar. Tıpkı bir restoranın şık masaları ve menüsü gibidir.
2. **Backend (Arka Yüz / Veritabanı):** Kullanıcı şifrelerinin saklandığı, verilerin tutulduğu yer. Tıpkı restoranın mutfağı ve deposu gibidir.

Eğer hem şık bir restoran açıp hem de mutfakta yemek pişirmeye, bulaşık yıkamaya, depoyu beklemeye çalışırsanız çok yorulursunuz. İşte **Firebase** tam bu noktada devreye girer!

Firebase, Google tarafından sunulan bir **BaaS (Backend as a Service - Hizmet olarak Arka Uç)** platformudur.
Analoji: **Firebase sizin için dünyanın en iyi aşçılarını, güvenlik görevlilerini ve devasa bir depoyu kiralayan bir hizmettir.** Siz sadece restoranın ön tarafını (React Native ile) güzelleştirirsiniz, mutfaktaki tüm zor işleri (sunucu kurma, veritabanı güvenliği) Firebase halleder.

Firebase'in Bize Sunduğu Temel Hizmetler:

- 🔐 **Authentication (Kimlik Doğrulama):** Restoranın kapısındaki güvenlik görevlisidir. "Bu kişi kayıtlı mı? Şifresi doğru mu?" diye kontrol eder.
- 🗄️ **Cloud Firestore (Veritabanı):** Restoranın devasa, süper hızlı dosya dolabıdır. Kullanıcıların notlarını, mesajlarını, isimlerini anında kaydeder ve bulur.
- 📦 **Storage (Depolama):** Resim, video ve ses dosyalarını tuttuğunuz devasa depodur.

💳 BÖLÜM 2: Kayıt Süreci ve Ücretlendirme (Cebimizden Para Çıkacak Mı?)

Firebase'in harika bir ücretsiz planı vardır. Buna **"Spark Plan"** denir.

- Kredi kartı girmeniz **gerekmez.**
- Öğrenci projeleri, dönem ödevleri ve kendi kendinize yapacağınız denemeler için sınırları o kadar geniştir ki, bu sınırları aşmanız neredeyse imkansızdır.
- Günlük 50.000 veri okuma, 20.000 veri yazma hakkınız vardır. Sınıftaki herkes uygulamanıza girse bile bu limiti dolduramazsınız!

Nasıl Kayıt Olunur?
Sadece bir **Google (Gmail)** hesabınızın olması yeterlidir. firebase.google.com adresine gidip sağ üstten "Go to console" (Konsola git) butonuna tıklamanız yeterlidir.

🎛️ BÖLÜM 3: Firebase Konsol Arayüzü ve Yeni Proje Oluşturma

Firebase Konsolu, bizim kontrol panelimizdir. Şimdi sıfırdan bir proje oluşturalım:

1. Firebase Konsoluna girin ve ekranın ortasındaki kocaman **"Add Project" (Proje Ekle)** butonuna tıklayın.
2. **Proje Adı:** Projenize bir isim verin (Örn: OkulNotUygulamam). Devam et butonuna basın.
3. **Google Analytics:** Karşınıza Analytics (Analiz aracı) ekranı gelecek. Bu araç uygulamanıza kaç kişi girmiş, nerelerden girmiş onu gösterir. *Ders projelerimizde süreci hızlandırmak için şimdilik bu tiki kapatabilirsiniz (Disable).* Eğer açık bırakırsanız bir sonraki adımda Google hesabınızı seçmeniz gerekir.
4. **Create Project (Projeyi Oluştur):** Butona basın ve o büyülü yükleme çubuğunun dolmasını, Firebase'in sizin için o devasa sunucuları hazırlamasını izleyin!
5. Çubuk dolduğunda **Continue (Devam Et)** diyerek kontrol panelinize ulaşın.

Sol Menü (Build / Geliştirme Menüsü): Ekranın sol tarafında "Build" menüsünü göreceksiniz. Burası bizim alet çantamızdır. Authentication ve Firestore Database araçlarını buradan seçeceğiz.

⚙️ BÖLÜM 4: Hizmetleri Aktifleştirme ve Ayarlar

Sadece proje açmak yetmez, kullanacağımız mutfak aletlerinin fişini prize takmalıyız!

Adım 1: Authentication (Güvenlik Görevlisini İşe Alma)

1. Sol menüden **Build -> Authentication** seçeneğine tıklayın.
2. **"Get Started" (Başla)** butonuna basın.
3. Karşınıza "Sign-in method" (Giriş Yöntemi) sekmesi gelecek. Buradan **Email/Password (E-posta/Şifre)** seçeneğine tıklayın.
4. İlk butonu (Enable / Aktifleştir) açık konuma getirin ve **Save (Kaydet)** deyin. Artık uygulamanızda kullanıcılar e-posta ve şifreyle kayıt olabilir!

Adım 2: Cloud Firestore (Dosya Dolabını Kurma)

1. Yine sol menüden **Build -> Firestore Database** seçeneğine tıklayın.
2. **"Create Database" (Veritabanı Oluştur)** butonuna basın.
3. Size konum soracak, Avrupa'da olduğumuz için eur3 (europe-west) gibi bir konum seçebilirsiniz. İleri deyin.
4. **ÇOK ÖNEMLİ:** Güvenlik kuralları sorulacak. Burada **"Start in test mode" (Test modunda başla)** seçeneğini işaretleyin.
(Eğitmen Notu: Öğrencilere test modunun veritabanını 30 gün boyunca herkese açık ve geliştirilebilir hale getirdiğini, gerçek bir uygulama yayınlarken bu kapıları şifrelememiz gerektiğini mutlaka belirtin.)
5. **Enable (Aktifleştir)** diyerek veritabanınızı hazır hale getirin.

🔗 BÖLÜM 5: React Native (Expo) ile Firebase'i Birleştirme

Şimdi restoranımız (React Native) ile mutfağımızı (Firebase) gizli bir tünelle birbirine bağlayacağız!

1. Firebase projenizin ana ekranına (Project Overview) dönün.
2. Ekranın ortasında iOS, Android ve Web ikonları (</>) göreceksiniz. Biz React Native (JavaScript) kullandığımız için **Web (</>) ikonuna** tıklayın.
3. Uygulamanıza bir takma ad (Nickname) verin ve **Register App** deyin.
4. Karşınıza **firebaseConfig** adında bir kod bloğu çıkacak. Bu sizin mutfağınızın gizli adresi ve anahtarlarıdır. Bu sayfayı kapatmayın, birazdan kopyalayacağız!

Expo Projesi Oluşturma ve Kurulum:

Bilgisayarınızın terminalini (Komut İstemi/CMD) açın ve şu komutları sırasıyla yazın:

```
# Yeni bir Expo projesi oluşturuyoruz
npx create-expo-app NotUygulamam

# Proje klasörünün içine giriyoruz
cd NotUygulamam

# Sihirli kuryemiz Firebase paketini projemize indiriyoruz
npm install firebase
```

💻 BÖLÜM 6 ve 7: Kodlama Zamanı (Senaryo ve Açıklamalı Kodlar)

Senaryomuz: Kullanıcılar e-posta ile uygulamamıza kayıt olacak/giriş yapacak. Başarıyla giriş yaptıktan sonra karşılarına bir not defteri çıkacak. Buraya not ekleyebilecek (Create), notlarını görebilecek (Read), güncelleyebilecek (Update) ve silebilecekler (Delete). Buna yazılım dünyasında **CRUD** operasyonları denir.

Dosya 1: firebaseConfig.js (Bağlantı Ayarları)

Projenizin ana dizininde firebaseConfig.js adında yeni bir dosya oluşturun ve içine şu kodları yazın:

```
// DERS NOTU: Bu dosya bizim Firebase ile olan köprümüzdür.
// 'firebase/app' içinden initializeApp fonksiyonunu çağırıyoruz. Bu, sistemi başlatır.
import { initializeApp } from "firebase/app";

// DERS NOTU: Kimlik doğrulama ve veritabanı araçlarımızı projemize dahil ediyoruz.
import { getAuth } from "firebase/auth";
import { getFirestore } from "firebase/firestore";

// AÇIKLAMA: Bu ayarlar sizin Firebase konsolundan (Bölüm 5'te) aldığınız GİZLİ ANAHTARLARDIR.
// Kendi Firebase konsolunuzdaki kodları buraya yapıştırmalısınız!
const firebaseConfig = {
  apiKey: "AIzaSyB-BURAYA_KENDI_BILGILERINIZ_GELECEK",
```

```
  authDomain: "proje-adiniz.firebaseapp.com",
  projectId: "proje-adiniz",
  storageBucket: "proje-adiniz.appspot.com",
  messagingSenderId: "123456789",
  appId: "1:123456789:web:abcdef"
};

// DERS NOTU: app değişkeni ile Firebase'i anahtarlarımızla uyandırıyoruz (çalıştırıyoruz).
const app = initializeApp(firebaseConfig);

// DERS NOTU: auth değişkeni bizim güvenlik görevlimiz, db değişkeni ise dosya dolabımızdır.
// Bunları diğer sayfalarda kullanabilmek için 'export' diyerek dışa aktarıyoruz.
export const auth = getAuth(app);
export const db = getFirestore(app);
```

Dosya 2: App.js (Uygulamanın Ana Merkezi)

App.js dosyanızın içini tamamen silin ve aşağıdakileri yapıştırın:

```
// DERS NOTU: React'ın temel kancalarını (Hook) çağırıyoruz. State (durum) ve Effect (yan etki).
import React, { useState, useEffect } from 'react';
import { StyleSheet, View, Text, ActivityIndicator } from 'react-native';

// DERS NOTU: Güvenlik görevlimiz auth'u ve Firebase'in kullanıcı durumunu izleyen fonksiyonunu
çağırıyoruz.
import { auth } from './firebaseConfig';
import { onAuthStateChanged } from 'firebase/auth';

// DERS NOTU: Birazdan oluşturacağımız iki farklı ekranı (Komponenti) içeri aktarıyoruz.
import AuthScreen from './AuthScreen';
import NotesScreen from './NotesScreen';

export default function App() {
  // DERS NOTU: 'user' adında bir kutu (state) yapıyoruz. İçinde giriş yapan kullanıcı bilgisi tutulacak.
Başlangıçta boş (null).
  const [user, setUser] = useState(null);

  // DERS NOTU: Uygulama ilk açıldığında arka planda kimin giriş yaptığını kontrol eden süreölçer/bekletici.
  const [loading, setLoading] = useState(true);

  // AÇIKLAMA: useEffect, uygulama ekranda ilk göründüğünde SADECE BİR KERE çalışan özel bir
fonksiyondur.
  useEffect(() => {
    // onAuthStateChanged: Güvenlik görevlimiz kapıda durur. Biri giriş yaparsa veya çıkış yaparsa bize
haber verir.
    const unsubscribe = onAuthStateChanged(auth, (currentUser) => {
      setUser(currentUser); // Gelen kişiyi 'user' kutumuza koyuyoruz.
```

```
      setLoading(false);    // Yükleme ekranını kapatıyoruz.
    });

    return unsubscribe; // Uygulama kapanırsa dinlemeyi bırak.
  }, []);

  // Eğer sistem hala kullanıcının kim olduğunu kontrol ediyorsa, ekranda dönen bir yükleme çemberi
(ActivityIndicator) gösteriyoruz.
  if (loading) {
    return (
      <View style={styles.center}>
        <ActivityIndicator size="large" color="#0000ff" />
        <Text>Giriş durumu kontrol ediliyor...</Text>
      </View>
    );
  }

  // AÇIKLAMA: Şartlı Render (Koşullu Gösterim)
  // Eğer 'user' kutusu doluysa (biri giriş yaptıysa) NotesScreen (Notlar) ekranını göster.
  // Eğer 'user' kutusu boşsa AuthScreen (Giriş/Kayıt) ekranını göster.
  return (
    <View style={styles.container}>
      {user ? <NotesScreen user={user} /> : <AuthScreen />}
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1, // Ekranın tamamını kapla
    backgroundColor: '#f5f5f5', // Arka planı açık gri yap
  },
  center: {
    flex: 1,
    justifyContent: 'center', // İçindekileri dikeyde ortala
    alignItems: 'center',     // İçindekileri yatayda ortala
  }
});
```

Dosya 3: AuthScreen.js (Giriş ve Kayıt Ekranı)

Yeni bir dosya oluşturun, adını AuthScreen.js koyun:

```
import React, { useState } from 'react';
import { StyleSheet, View, Text, TextInput, TouchableOpacity, Alert } from 'react-native';

// DERS NOTU: Firebase'den üye olma (createUser) ve giriş yapma (signIn) komutlarını çağırıyoruz.
import { auth } from './firebaseConfig';
import { createUserWithEmailAndPassword, signInWithEmailAndPassword } from 'firebase/auth';

export default function AuthScreen() {
  // AÇIKLAMA: Kullanıcının klavyeden yazdıklarını tutacağımız state'ler (kutucuklar).
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  // Bu değişken true ise Login (Giriş), false ise Register (Kayıt) ekranındayız demektir.
  const [isLogin, setIsLogin] = useState(true);

  // DERS NOTU: Kayıt ol butonuna basıldığında çalışacak fonksiyon.
  const handleRegister = async () => {
    try {
      // Firebase'e "Bu e-posta ve şifreyle yeni birini kaydet" diyoruz.
      // 'await' kelimesi, bu işlem bitene kadar kodun beklemesini sağlar.
      await createUserWithEmailAndPassword(auth, email, password);
      Alert.alert("Başarılı!", "Kayıt oldunuz, artık giriş yapabilirsiniz.");
    } catch (error) {
      // Eğer e-posta formatı yanlışsa veya şifre 6 karakterden kısaysa Firebase hata fırlatır, bunu ekranda
gösteririz.
      Alert.alert("Hata Oluştu", error.message);
    }
  };

  // DERS NOTU: Giriş yap butonuna basıldığında çalışacak fonksiyon.
  const handleLogin = async () => {
    try {
      // Firebase'e "Bu bilgilerle sisteme giriş yap" diyoruz.
      await signInWithEmailAndPassword(auth, email, password);
      // Başarılı olursa App.js'deki onAuthStateChanged durumu yakalayacak ve bizi Notlar ekranına atacak!
    } catch (error) {
      Alert.alert("Giriş Başarısız", "Bilgilerinizi kontrol edin.");
    }
  };

  return (
    <View style={styles.container}>
      {/* Eğer isLogin true ise başlık Giriş Yap, false ise Kayıt Ol yazsın. */}
      <Text style={styles.title}>{isLogin ? 'Giriş Yap' : 'Kayıt Ol'}</Text>
```

```
      <TextInput
        style={styles.input}
        placeholder="E-posta adresiniz"
        value={email}
        onChangeText={setEmail} // Klavye değiştikçe email state'ini güncelle
        autoCapitalize="none"   // İlk harfi otomatik büyütmeyi kapat
      />

      <TextInput
        style={styles.input}
        placeholder="Şifreniz"
        value={password}
        onChangeText={setPassword}
        secureTextEntry // Şifrenin nokta nokta (***) görünmesini sağlar
      />

      {/* Dinamik Buton: Modumuza göre Login veya Register fonksiyonunu çalıştırır. */}
      <TouchableOpacity
        style={styles.button}
        onPress={isLogin ? handleLogin : handleRegister}
      >
        <Text style={styles.buttonText}>{isLogin ? 'Sisteme Gir' : 'Hesap Oluştur'}</Text>
      </TouchableOpacity>

      {/* Ekranlar arası geçiş yapmamızı sağlayan ufak bir buton */}
      <TouchableOpacity onPress={() => setIsLogin(!isLogin)}>
        <Text style={styles.switchText}>
          {isLogin ? "Hesabın yok mu? Kayıt Ol" : "Zaten üye misin? Giriş Yap"}
        </Text>
      </TouchableOpacity>
    </View>
  );
}

// Tasarım (CSS benzeri) ayarları
const styles = StyleSheet.create({
  container: { flex: 1, justifyContent: 'center', padding: 20 },
  title: { fontSize: 32, fontWeight: 'bold', marginBottom: 20, textAlign: 'center', color: '#333' },
  input: { borderWidth: 1, borderColor: '#ccc', padding: 15, borderRadius: 10, marginBottom: 15,
backgroundColor: '#fff' },
  button: { backgroundColor: '#FF5722', padding: 15, borderRadius: 10, alignItems: 'center' },
  buttonText: { color: 'white', fontWeight: 'bold', fontSize: 18 },
  switchText: { marginTop: 20, textAlign: 'center', color: '#007BFF', fontSize: 16 }
});
```

Dosya 4: NotesScreen.js (Veritabanı ile CRUD Operasyonları)

Son olarak veritabanı ekranımız. Yeni dosya oluşturun NotesScreen.js:

```
import React, { useState, useEffect } from 'react';
import { StyleSheet, View, Text, TextInput, TouchableOpacity, FlatList, Alert } from 'react-native';

// DERS NOTU: Güvenlik görevlimiz (çıkış yapmak için) ve Dosya dolabımız (veri eklemek için)
import { auth, db } from './firebaseConfig';
import { signOut } from 'firebase/auth';

// DERS NOTU: Firestore (Veritabanı) işlemleri için gerekli fonksiyonları çağırıyoruz.
// collection (Klasör), addDoc (Belge Ekle), deleteDoc (Belge Sil), updateDoc (Güncelle), onSnapshot (Canlı Takip)
import { collection, addDoc, deleteDoc, updateDoc, doc, onSnapshot, query, where } from 'firebase/firestore';

export default function NotesScreen({ user }) {
  const [notes, setNotes] = useState([]); // Tüm notlarımızın listesi
  const [newNote, setNewNote] = useState(''); // Yeni eklenecek notun yazısı
  const [editingId, setEditingId] = useState(null); // Eğer bir notu güncelliyorsak, o notun ID'si burada durur.

  // AÇIKLAMA: READ (Veri Okuma) İşlemi - Canlı Dinleme
  useEffect(() => {
    // 1. Nereye bakacağımızı söylüyoruz: 'notes' adında bir klasör yarat veya bul.
    // Sadece sisteme giren kişiye (user.uid) ait olan notları filtreleyerek (where) getiriyoruz.
    const q = query(collection(db, 'notes'), where("userId", "==", user.uid));

    // 2. onSnapshot: Mükemmel bir özelliktir! Veritabanında bir değişiklik olursa ANINDA telefonu günceller (Real-time).
    const unsubscribe = onSnapshot(q, (querySnapshot) => {
      const notesArray = [];
      // Klasördeki her bir dosyayı (doc) tek tek dolaşıp diziye ekliyoruz.
      querySnapshot.forEach((doc) => {
        notesArray.push({ id: doc.id, ...doc.data() });
      });
      setNotes(notesArray); // State'i güncelliyoruz, ekran baştan çiziliyor.
    });

    return unsubscribe;
  }, []);

  // AÇIKLAMA: CREATE (Veri Ekleme) İşlemi
  const handleAddNote = async () => {
    if (newNote.trim() === '') return; // Eğer kutu boşsa hiçbir şey yapma

    try {
      // 'notes' klasörünün içine yeni bir belge fırlatıyoruz.
      await addDoc(collection(db, 'notes'), {
```

```
      text: newNote,       // Notun içeriği
      userId: user.uid,    // Notun kime ait olduğu (Güvenlik için çok önemli!)
      createdAt: new Date()// Hangi tarih/saatte eklendiği
    });
    setNewNote(''); // İşlem bitince metin kutusunu temizle
  } catch (error) {
    Alert.alert("Hata", "Not eklenemedi.");
  }
};

// AÇIKLAMA: DELETE (Veri Silme) İşlemi
const handleDelete = async (id) => {
  try {
    // Silinecek belgenin tam adresini (doc) veriyoruz ve deleteDoc ile yok ediyoruz.
    await deleteDoc(doc(db, 'notes', id));
  } catch (error) {
    Alert.alert("Hata", "Not silinemedi.");
  }
};

// AÇIKLAMA: UPDATE (Veri Güncelleme) İşlemi
const handleUpdate = async () => {
  try {
    // Güncellenecek belgenin adresini bul ve sadece 'text' kısmını yeni yazıyla değiştir.
    await updateDoc(doc(db, 'notes', editingId), {
      text: newNote
    });
    setNewNote(''); // Kutuyu temizle
    setEditingId(null); // Düzenleme modundan çık
  } catch (error) {
    Alert.alert("Hata", "Not güncellenemedi.");
  }
};

// AÇIKLAMA: Düzenle butonuna basınca olacaklar
const startEditing = (note) => {
  setNewNote(note.text); // Tıklanan notun yazısını yukarıdaki girdi kutusuna taşı
  setEditingId(note.id); // Sistemi "Güncelleme" moduna geçir
};

// Çıkış Yapma Fonksiyonu
const handleLogout = () => {
  signOut(auth); // Güvenlik görevlisine "Beni sistemden at" diyoruz.
};

return (
  <View style={styles.container}>
    <View style={styles.header}>
      <Text style={styles.welcome}>Merhaba, {user.email}</Text>
```

```
        <TouchableOpacity style={styles.logoutButton} onPress={handleLogout}>
          <Text style={styles.logoutText}>Çıkış Yap</Text>
        </TouchableOpacity>
      </View>

      <View style={styles.inputContainer}>
        <TextInput
          style={styles.input}
          placeholder="Aklına ne geldi?"
          value={newNote}
          onChangeText={setNewNote}
        />
        {/* editingId doluysa GÜNCELLE butonu, boşsa EKLE butonu gösteriyoruz. */}
        <TouchableOpacity
          style={styles.addButton}
          onPress={editingId ? handleUpdate : handleAddNote}
        >
          <Text style={styles.addButtonText}>{editingId ? "Güncelle" : "Ekle"}</Text>
        </TouchableOpacity>
      </View>

      {/* FlatList: Veritabanından gelen notları alt alta performanslı bir şekilde dizer. */}
      <FlatList
        data={notes}
        keyExtractor={(item) => item.id}
        renderItem={({ item }) => (
          <View style={styles.noteCard}>
            <Text style={styles.noteText}>{item.text}</Text>

            <View style={styles.actionButtons}>
              <TouchableOpacity style={styles.editBtn} onPress={() => startEditing(item)}>
                <Text style={styles.btnText}>Düzenle</Text>
              </TouchableOpacity>

              <TouchableOpacity style={styles.deleteBtn} onPress={() => handleDelete(item.id)}>
                <Text style={styles.btnText}>Sil</Text>
              </TouchableOpacity>
            </View>
          </View>
        )}
      />
    </View>
  );
}

// Tasarım (CSS benzeri) ayarları
const styles = StyleSheet.create({
  container: { flex: 1, padding: 20, paddingTop: 50 },
  header: { flexDirection: 'row', justifyContent: 'space-between', alignItems: 'center', marginBottom: 20 },
```

```
  welcome: { fontSize: 16, fontWeight: '600' },
  logoutButton: { backgroundColor: '#d9534f', padding: 8, borderRadius: 5 },
  logoutText: { color: 'white', fontWeight: 'bold' },
  inputContainer: { flexDirection: 'row', marginBottom: 20 },
  input: { flex: 1, borderWidth: 1, borderColor: '#ccc', padding: 10, borderRadius: 5, backgroundColor: '#fff',
marginRight: 10 },
  addButton: { backgroundColor: '#4CAF50', justifyContent: 'center', paddingHorizontal: 20, borderRadius: 5
},
  addButtonText: { color: 'white', fontWeight: 'bold' },
  noteCard: { backgroundColor: '#fff', padding: 15, borderRadius: 8, marginBottom: 10, flexDirection: 'row',
justifyContent: 'space-between', alignItems: 'center', elevation: 2 },
  noteText: { flex: 1, fontSize: 16 },
  actionButtons: { flexDirection: 'row' },
  editBtn: { backgroundColor: '#f0ad4e', padding: 8, borderRadius: 5, marginRight: 5 },
  deleteBtn: { backgroundColor: '#d9534f', padding: 8, borderRadius: 5 },
  btnText: { color: 'white', fontSize: 12, fontWeight: 'bold' }
});
```

Kapanış

- Firebase'in kapısındaki güvenliği **(Auth)** kurdunuz.
- Arka planda devasa bir dosya dolabı **(Firestore)** açtınız.
- React Native ile kullanıcının eline harika bir menü **(Uygulama Arayüzü)** verdiniz.