

ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



BLM4538 – IOS İLE MOBİL UYGULAMA GELİŞTİRME II

Araba Satış Uygulaması

Yasin Çelik – 21290412

Github Linki- <https://github.com/YasinCelik01/YasinCelik01-BLM4538>

Video Linki Youtube - https://youtu.be/zA2yF_yKAKg

Araba Satış Uygulaması

Projenin Amacı

Bu proje, bir araba satış mobil uygulamasını temsil etmektedir. Yapmış olduğum bu uygulama kullanıcı oluşturma, kullanıcı girişi yapma, araba satış ilanı oluşturma ve favorilere ekleme gibi birçok özellik bulunmaktadır. Kullanıcılarla ve arabalarla ilgili veriler firebase veri tabanında tutulmaktadır. Proje REACT NATIVE üzerinden geliştirilmiştir tüm backend ve frontend kısımları ise REACT NATIVE üzerinden oluşturulmuştur. Projenin genel özellikleri şu şekildedir;

- Kullanıcı kaydı ve giriş sistemi.
- Kullanıcı profili düzenleme.
- JWT ile güvenli kimlik doğrulama.
- Kullanıcı rollerine göre farklı erişim seviyeleri (admin, satıcı).
- Satıcıların araba ilanı oluşturma.
- İlan silme işlemi.
- İlan düzenleme işlemi.
- Detaylı araç bilgisi sayfası (model, fiyat, yıl vb.).
- API üzerinden tüm veritabanı işlemlerinin (CRUD) yapılması.
- Firebase kullanılarak veritabanı yönetimi.
- Resim yükleme ve yönetme (araç fotoğrafı).
- İlanları favorilere ekleyebilme.

Projenin Sayfa Yapısı

Anasayfa

Kullanıcıların giriş yaptıktan sonra karşılaştıkları ekrandır. Uygulamadaki araba ilanları bu sayfada gösterilmektedir. Sayfanın en altında bulunan bar kısmında çeşitli butonlar bulunmaktadır kullanıcı hangi işlemi yapmak istiyorsa onu seçebilir. Sayfanın üst kısmında ise filtreler butonu yer almaktadır, kullanıcılar ilanı filtrelere göre listeleyebilirler.



Anasayfa için API kodu aşağıdadır.

```
getApprovedCars: async () => {
  try {
    const q = query(collection(db, 'cars'), where('status',
'==', 'approved'));
    const querySnapshot = await getDocs(q);
    const cars = [];
    querySnapshot.forEach((doc) => {
      cars.push({ id: doc.id, ...doc.data() });
    });
    return { data: cars, error: null };
  } catch (error) {
    return { data: null, error: error.message };
  }
},
```

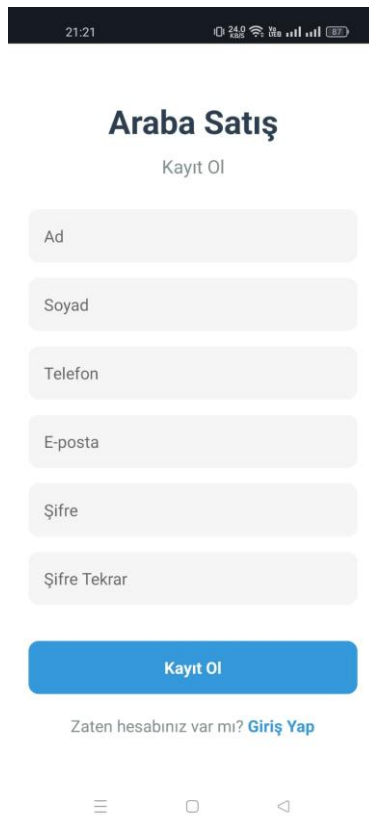
Anasayfa için Api ye istek atan React kodu aşağıdadır.

```
const loadCars = async () => {
  try {
    setLoading(true);
    const { data, error } = await carApi.getApprovedCars();
    if (error) {
      console.error('Araçları getirme hatası:', error);
      return;
    }
    setCars(data || []);
  } catch (error) {
```

```
    console.error('Araçları getirme hatası:', error);
  } finally {
    setLoading(false);
    setRefreshing(false);
  }
};
```

Register Sayfası

Kullanıcı kayıt olmak istediğinde bu sayfaya gelir ve gerekli bilgileri girerek kaydolma işlemini gerçekleştirir.

A mobile app registration screen titled "Araba Satış" (Car Sale) with a subtitle "Kayıt Ol" (Register). The form contains input fields for "Ad" (Name), "Soyad" (Surname), "Telefon" (Phone), "E-posta" (Email), "Şifre" (Password), and "Şifre Tekrar" (Repeat Password). Below the fields is a blue "Kayıt Ol" button. At the bottom, there is a link "Zaten hesabınız var mı? Giriş Yap" (Already have an account? Login). The screen has a status bar at the top showing the time 21:21 and various icons.

Register sayfası için API kodu aşağıdadır.

```
async register(email, password, userData) {
  try {
    const userCredential = await
createUserWithEmailAndPassword(auth, email, password);
    const user = userCredential.user;

    // Firestore'a kullanıcı bilgilerini kaydet
    await setDoc(doc(db, 'users', user.uid), {
      ...userData,
      email: user.email,
      createdAt: new Date()
```

```

    });

    return {
      success: true,
      user: {
        id: user.uid,
        email: user.email,
        ...userData
      }
    };
  } catch (error) {
    let errorMessage = 'Kayıt olurken bir hata oluştu';

    if (error.code === 'auth/email-already-in-use') {
      errorMessage = 'Bu e-posta adresi zaten kullanımda';
    } else if (error.code === 'auth/invalid-email') {
      errorMessage = 'Geçersiz e-posta adresi';
    } else if (error.code === 'auth/weak-password') {
      errorMessage = 'Şifre çok zayıf';
    }

    return { success: false, error: errorMessage };
  }
},

```

Register sayfası için Api ye istek atan React kodu aşağıdadır.

```

const handleRegister = async () => {
  setRegisterError('');
  if (!validateForm()) return;

  setLoading(true);
  try {
    const { success, error } = await authApi.register(email,
password, {
  firstName,
  lastName,
  phone
});

    if (success) {
      setRegisterError('');
      Alert.alert(

```

```
        'Başarılı',
        'Kayıt işlemi başarıyla tamamlandı',
        [
            {
                text: 'Tamam',
                onPress: () => {
                    setEmail('');
                    setPassword('');
                    setConfirmPassword('');
                    setFirstName('');
                    setLastName('');
                    setPhone('');
                    setErrors({});
                    navigation.navigate('Login');
                }
            }
        ]
    );
} else {
    setRegisterError(error || 'Bilinmeyen bir hata oluştu');
}
} catch (error) {
    setRegisterError(error?.message || 'Kayıt işlemi sırasında bir hata oluştu');
} finally {
    setLoading(false);
}
};

const validateForm = () => {
    if (!firstName || !lastName || !phone || !email || !password || !confirmPassword) {
        Alert.alert('Hata', 'Lütfen tüm alanları doldurun');
        return false;
    }

    if (password !== confirmPassword) {
        Alert.alert('Hata', 'Şifreler eşleşmiyor');
        return false;
    }

    return true;
};
```

Login Sayfası

Kullanıcı giriş yapmak istediğinde bu sayfaya gelir ve gerekli bilgileri girerek login işlemini gerçekleştirir.

21:21

Giriş Yap

E-posta

Şifre

Giriş Yap

Hesabınız yok mu? [Kayıt Ol](#)

Admin hesabı oluşturmak için [tıklayın](#)

≡ □ ◀

Login sayfası için API kodu aşağıdadır.

```
async login(email, password) {
  try {
    const userCredential = await
signInWithEmailAndPassword(auth, email, password);
    const userDoc = await getDoc(doc(db, 'users',
userCredential.user.uid));

    if (!userDoc.exists()) {
      return { success: false, error: 'Kullanıcı bilgileri
bulunamadı' };
    }

    return {
      success: true,
      user: {
        id: userCredential.user.uid,
        ...userDoc.data()
      }
    };
  } catch (error) {
    let errorMessage = 'Giriş yapılırken bir hata oluştu';

    if (error.code === 'auth/invalid-email') {
      errorMessage = 'Geçersiz e-posta adresi';
    }
  }
}
```

```
    } else if (error.code === 'auth/user-disabled') {
      errorMessage = 'Bu hesap devre dışı bırakılmış';
    } else if (error.code === 'auth/user-not-found' ||
error.code === 'auth/wrong-password') {
      errorMessage = 'E-posta veya şifre hatalı';
    }

    return { success: false, error: errorMessage };
  }
},
```

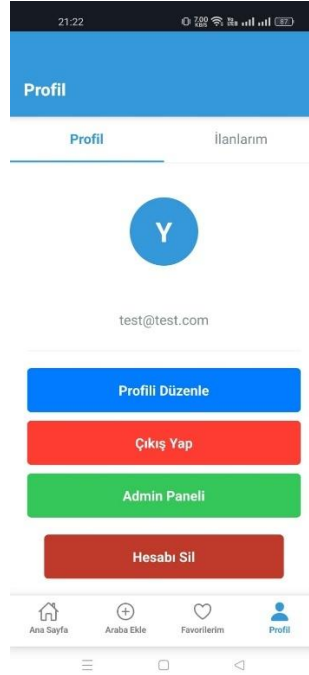
Login sayfası için Api ye istek atan React kodu aşağıdadır.

```
try {
  const { success, user, error } = await authApi.login(email,
password);

  if (success) {
    navigation.replace('Main');
  } else {
    Alert.alert('Hata', error);
  }
} catch (error) {
  Alert.alert('Hata', 'Giriş yapılırken bir hata oluştu');
} finally {
  setLoading(false);
}
};
```


Profil Sayfası

Profil sayfası kullanıcı giriş yaptıktan sonra erişilen bir sayfadır. Eğer kullanıcı admin yetkisine sahipse sadece admin kullanıcılarına özel admin panel butonu vardır.



Profil sayfasının API kodu aşağıdadır.

```
getUserProfile: async (userId) => {
  try {
    const userDoc = await getDoc(doc(db, 'users', userId));
    if (userDoc.exists()) {
      return { data: userDoc.data(), error: null };
    }
    return { data: null, error: 'Kullanıcı bulunamadı' };
  } catch (error) {
    return { data: null, error: error.message };
  }
},
```

Profil sayfası için Api ye istek atan React kodu aşağıdadır.

```
const { data: profileData, error: profileError } = await
userApi.getUserProfile(currentUser.uid);
if (profileError) throw profileError;
setUserProfile(profileData);
```

Hesabımı Sil Butonu

Hesabımı sil butonuna basıldığında aşağıdaki gibi bir uyarı çıkar ve kullanıcının yanlışlıkla hesabını silmesine engel olunur. Eğer silme işlemini gerçekleştirmek istiyorsa çıkan uyarıda HESABI SİL butonuna basması yeterlidir.



Hesap silme işlemi için API kodu aşağıdadır.

```
deleteUserProfile: async (userId) => {
  try {
    await deleteDoc(doc(db, 'users', userId));
    return { success: true, error: null };
  } catch (error) {
    return { success: false, error: error.message };
  }
},
```

Hesap Silme işlemi için Api ye istek atan React kodu aşağıdadır.

```
const handleDeleteAccount = async () => {
  try {
    const { success: deleteProfileSuccess, error:
deleteProfileError } =
    await userApi.deleteUserProfile(user.uid);

    if (deleteProfileError) {
      Alert.alert('Hata', 'Profil verileri silinirken bir hata
oluşturtu');
    }
  }
}
```

```
        return;
    }

    const { success: deleteAuthSuccess, error: deleteAuthError
} =
    await authApi.deleteAccount();

    if (deleteAuthError) {
        Alert.alert('Hata', 'Hesap silinirken bir hata oluştu');
        return;
    }

    navigation.replace('Login');
} catch (error) {
    Alert.alert('Hata', 'Hesap silinirken bir hata oluştu');
}
};
```

İlan Ver Sayfası

İlan ver sayfası araçla ilgili bilgilerin girilip admin onayına gönderildiği sayfadır.

21:22

Araba Ekle

Marka

Model

Yıl

Kilometre

Fiyat (TL)

Yakıt Türü: Benzin

Vites: Manuel

Açıklama

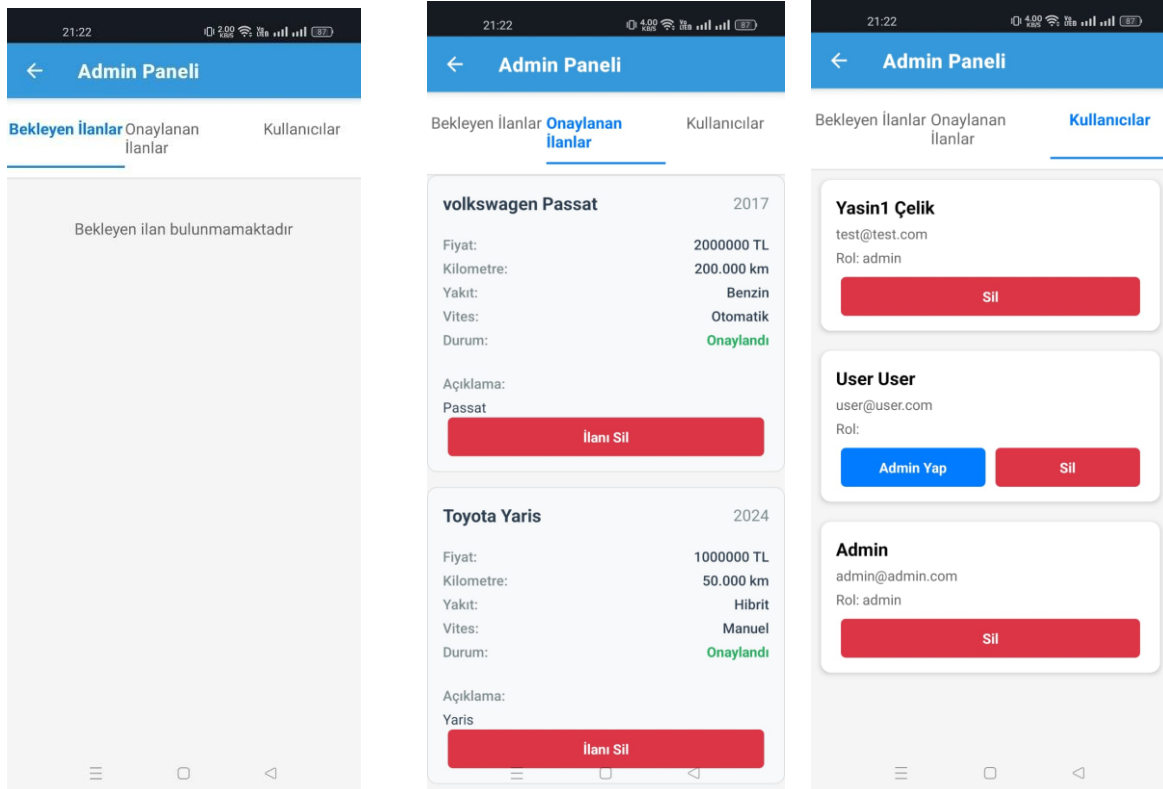
Ana Sayfa Araba Ekle Favorilerim Profil

İlan ver sayfasının API kodları aşağıdadır.

```
createCar: async (carData) => {
  try {
    const carRef = doc(db, 'cars', carData.id);
    await setDoc(carRef, {
      ...carData,
      status: 'pending', // İlan durumu: pending (beklemede),
      approved (onaylandı), rejected (reddedildi)
      createdAt: new Date().toISOString(),
      updatedAt: new Date().toISOString()
    });
    return { success: true, error: null };
  } catch (error) {
    return { success: false, error: error.message };
  }
},
```

Admin Panel Sayfası

Bu sayfa ilana çıkacak araçların admin onayından geçmesini sağlar. Admin onay vermezse araç satışa çıkamaz. Onaylanan ilanlar admin panelden gözükebilir ve ilanlar silinebilir. Ayrıca kullanıcı hesapları admin panelden silinebilir veya admin yetkisi verilebilir.



Bu sayfa için kullanılan API kodları aşağıdadır.

```
getAllCars: async () => {
  try {
    const carsSnapshot = await getDocs(collection(db, 'cars'));
    const cars = [];
    carsSnapshot.forEach((doc) => {
      cars.push({ id: doc.id, ...doc.data() });
    });
    return cars;
  } catch (error) {
    console.error('Araçları getirme hatası:', error);
    throw error;
  }
},
```

```
updateCarStatus: async (carId, status) => {
  try {
    await updateDoc(doc(db, 'cars', carId), {
      status,
      updatedAt: new Date().toISOString()
    });
    return { success: true, error: null };
  } catch (error) {
    return { success: false, error: error.message };
  }
},
```

```
deleteCar: async (carId) => {
  try {
    await deleteDoc(doc(db, 'cars', carId));
    return { success: true };
  } catch (error) {
    console.error('Araç silme hatası:', error);
    throw error;
  }
},
```

```
getPendingCars: async () => {
  try {
    const q = query(collection(db, 'cars'), where('status',
'==', 'pending'));
    const querySnapshot = await getDocs(q);
    const cars = [];
    querySnapshot.forEach((doc) => {
```

```

        cars.push({ id: doc.id, ...doc.data() });
    });
    return { data: cars, error: null };
} catch (error) {
    return { data: null, error: error.message };
}
},

```

```

approveCar: async (carId) => {
    try {
        await updateDoc(doc(db, 'cars', carId), {
            status: 'approved',
            updatedAt: new Date().toISOString()
        });
        return { success: true, error: null };
    } catch (error) {
        return { success: false, error: error.message };
    }
},

```

```

getUserAds: async (userId) => {
    try {
        const carsRef = collection(db, 'cars');
        const q = query(carsRef, where('sellerId', '==', userId));
        const querySnapshot = await getDocs(q);

        const ads = [];
        querySnapshot.forEach((doc) => {
            ads.push({ id: doc.id, ...doc.data() });
        });

        return { data: ads, error: null };
    } catch (error) {
        console.error('Kullanıcı ilanları getirme hatası:', error);
        return { data: null, error: error.message };
    }
},

```

```

getApprovedCars: async () => {
    try {
        const q = query(collection(db, 'cars'), where('status',
'==', 'approved'));
        const querySnapshot = await getDocs(q);
        const cars = [];
        querySnapshot.forEach((doc) => {

```

```

        cars.push({ id: doc.id, ...doc.data() });
    });
    return { data: cars, error: null };
} catch (error) {
    return { data: null, error: error.message };
}
},

```

```

deleteUserByAdmin: async (userId) => {
    try {
        await deleteDoc(doc(db, 'users', userId));
        return { success: true, error: null };
    } catch (error) {
        return { success: false, error: error.message };
    }
},

```

Admin Panel sayfası için Kullanılacak Apilere istek atan React kodu aşağıdadır.

```

const AdminScreen = () => {
    const [activeTab, setActiveTab] = useState('pending'); //
    pending, approved, users
    const [ads, setAds] = useState([]);
    const [users, setUsers] = useState([]);
    const [loading, setLoading] = useState(true);
    const [refreshing, setRefreshing] = useState(false);
    const [pendingCars, setPendingCars] = useState([]);
    const navigation = useNavigation();

    const loadData = async () => {
        try {
            setLoading(true);
            if (activeTab === 'users') {
                const usersData = await userApi.getAllUsers();
                setUsers(usersData);
            } else if (activeTab === 'pending') {
                const { data, error } = await carApi.getPendingCars();
                if (error) {
                    console.error('Bekleyen ilanlar yükleme hatası:',
error);
                    setAds([]);
                    return;
                }
                setAds(data || []);
            } else {

```

```
    // Onaylanan ilanlar için
    const { data, error } = await carApi.getApprovedCars();
    if (error) {
        console.error('Onaylanan ilanlar yükleme hatası:',
error);
        setAds([]);
        return;
    }
    setAds(data || []);
}
} catch (error) {
    console.error('Veri yükleme hatası:', error);
    setAds([]);
} finally {
    setLoading(false);
    setRefreshing(false);
}
};

useEffect(() => {
    loadData();
    loadPendingCars();
}, [activeTab]);

const loadPendingCars = async () => {
    try {
        setLoading(true);
        const { data, error } = await carApi.getPendingCars();
        if (error) {
            console.error('Bekleyen ilanlar yükleme hatası:', error);
            setPendingCars([]);
            return;
        }
        setPendingCars(data || []);
    } catch (error) {
        console.error('Bekleyen ilanlar yükleme hatası:', error);
        setPendingCars([]);
    } finally {
        setLoading(false);
    }
};

const handleApprove = async (adId) => {
    try {
```



```

        await carApi.updateCarStatus(adId, 'approved');
        Alert.alert('Başarılı', 'İlan onaylandı');
        loadData();
        loadPendingCars();
    } catch (error) {
        console.error('İlan onaylama hatası:', error);
        Alert.alert('Hata', 'İlan onaylanırken bir hata oluştu');
    }
};

const handleReject = async (adId) => {
    try {
        await carApi.updateCarStatus(adId, 'rejected');
        Alert.alert('Başarılı', 'İlan reddedildi');
        loadData();
        loadPendingCars();
    } catch (error) {
        console.error('İlan reddetme hatası:', error);
        Alert.alert('Hata', 'İlan reddedilirken bir hata oluştu');
    }
};

const handleUpdateUserRole = async (userId, newRole) => {
    try {
        await userApi.updateUserRole(userId, newRole);
        Alert.alert('Başarılı', 'Kullanıcı rolü güncellendi');
        loadData();
    } catch (error) {
        Alert.alert('Hata', 'Kullanıcı rolü güncellenirken bir hata oluştu');
    }
};

const handleDeleteUser = async (userId) => {
    Alert.alert(
        'Kullanıcıyı Sil',
        'Bu kullanıcıyı silmek istediğinizden emin misiniz?',
        [
            {
                text: 'İptal',
                style: 'cancel'
            },
            {
                text: 'Sil',

```

```

        style: 'destructive',
        onPress: async () => {
            try {
                const { success, error } = await
userApi.deleteUser(userId);
                if (success) {
                    Alert.alert('Başarılı', 'Kullanıcı başarıyla
silindi');
                    if (auth.currentUser && auth.currentUser.uid ===
userId) {
                        await signOut(auth);
                        navigation.replace('Login');
                    } else {
                        loadData();
                    }
                } else {
                    Alert.alert('Hata', error || 'Kullanıcı
silinirken bir hata oluştu');
                }
            } catch (error) {
                Alert.alert('Hata', 'Kullanıcı silinirken bir hata
oluştı');
            }
        }
    ]
);
};

const handleDeleteCar = async (carId) => {
    Alert.alert(
        'İlanı Sil',
        'Bu ilanı silmek istediğinizden emin misiniz?',
        [
            {
                text: 'İptal',
                style: 'cancel'
            },
            {
                text: 'Sil',
                style: 'destructive',
                onPress: async () => {
                    try {

```

```
const { success, error } = await
carApi.deleteCar(carId);
if (success) {
  Alert.alert('Başarılı', 'İlan başarıyla
silindi');
  navigation.navigate('Home', { refresh: true });
  loadData();
} else {
  Alert.alert('Hata', error || 'İlan silinirken bir
hata oluştu');
}
} catch (error) {
  Alert.alert('Hata', 'İlan silinirken bir hata
oluştı');
}
}
}
]
);
};
```

Araçlarım Sayfası

İlanlarım sayfasında kullanıcılar ilandaki araçlarını ve ilana çıkmak için bekleyen araçlarını görebilirler, araç ilanlarını silebilirler ve admin onay durumunu görebilirler.



İlanlarım sayfasının API kodu aşağıdadır.

```
getUserAds: async (userId) => {
  try {
    const carsRef = collection(db, 'cars');
    const q = query(carsRef, where('sellerId', '==', userId));
    const querySnapshot = await getDocs(q);

    const ads = [];
    querySnapshot.forEach((doc) => {
      ads.push({ id: doc.id, ...doc.data() });
    });

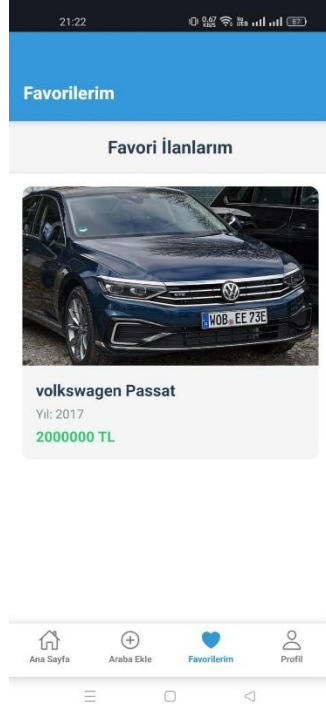
    return { data: ads, error: null };
  } catch (error) {
    console.error('Kullanıcı ilanları getirme hatası:', error);
    return { data: null, error: error.message };
  }
},
```

Araçlarım sayfası için Api ye istek atan React kodu aşağıdadır.

```
const loadMyAds = async () => {
  setAdsLoading(true);
  try {
    const { data, error } = await carApi.getUserAds(user.uid);
    if (error) throw error;
    setMyAds(data);
  } catch (error) {
    Alert.alert('Hata', 'İlanlar yüklenirken bir hata oluştu');
  } finally {
    setAdsLoading(false);
  }
};
```

Favorilerim Sayfası

Favorilerim sayfasına kullanıcının favorilere eklediği araçlar listelenmektedir.



Favorilerim sayfasına ait API kodu aşağıdadır.

```
addFavorite: async (userId, carId) => {
  try {
    // Aynı favori var mı kontrol et
    const q = query(collection(db, 'favorites'),
where('userId', '==', userId), where('carId', '==', carId));
    const querySnapshot = await getDocs(q);
    if (!querySnapshot.empty) {
      return { success: false, error: 'Bu ilan zaten
favorilerde.' };
    }
    await addDoc(collection(db, 'favorites'), {
      userId,
      carId,
      createdAt: new Date().toISOString(),
    });
    return { success: true, error: null };
  } catch (error) {
    return { success: false, error: error.message };
  }
},

// Favoriden çıkar
```

```

removeFavorite: async (userId, carId) => {
  try {
    const q = query(collection(db, 'favorites'),
where('userId', '==', userId), where('carId', '==', carId));
    const querySnapshot = await getDocs(q);
    if (querySnapshot.empty) {
      return { success: false, error: 'Favori bulunamadı.' };
    }
    // Tüm eşleşen favorileri sil
    const deletePromises = querySnapshot.docs.map(docSnap =>
deleteDoc(doc(db, 'favorites', docSnap.id)));
    await Promise.all(deletePromises);
    return { success: true, error: null };
  } catch (error) {
    return { success: false, error: error.message };
  }
},

// Kullanıcının favori arabalarını getir (sadece carId listesi döner)
getUserFavoriteCarIds: async (userId) => {
  try {
    const q = query(collection(db, 'favorites'),
where('userId', '==', userId));
    const querySnapshot = await getDocs(q);
    const carIds = querySnapshot.docs.map(docSnap =>
docSnap.data().carId);
    return { data: carIds, error: null };
  } catch (error) {
    return { data: null, error: error.message };
  }
},
};

```

Favoriler sayfası için Api ye istek atan React kodu aşağıdadır.

```

const { data: carIds, error } = await
favoriteApi.getUserFavoriteCarIds(userId);
if (error) throw new Error(error);
if (!carIds || carIds.length === 0) {
  setFavoriteCars([]);
  setLoading(false);
  return;
}

```

```
// Favori arabaların detaylarını çek
const carDetailsPromises = carIds.map(carId =>
carApi.getCarById(carId));
const carDetailsResults = await
Promise.all(carDetailsPromises);
const cars = carDetailsResults
.filter(res => res.data)
.map(res => res.data);
setFavoriteCars(cars);
} catch (error) {
Alert.alert('Hata', 'Favoriler yüklenirken bir hata
oluştı.');
```

Filtreleme ekranı

Anasayfada bulunan ilanları uygun özelliklere göre filtrelemeyi sağlar.

21:22 0.70 KB/s 4G 87

Filtrele

Marka

Model

Min Yıl

Max Yıl

Min Fiyat

Max Fiyat

Min Km

Max Km

Yakıt Türü

Vites

Kapat Temizle

Ana Sayfa Araba Ekle Favorilerim Profil

Filtreleme sayfası için React kodu aşağıdadır.

```
const HomeScreen = ({ navigation }) => {
  const [loading, setLoading] = useState(true);
  const [cars, setCars] = useState([]);
  const [refreshing, setRefreshing] = useState(false);
  const [favoriteCarIds, setFavoriteCarIds] = useState([]);
  const [filterModalVisible, setFilterModalVisible] =
useState(false);
  const [filterBrand, setFilterBrand] = useState('');
  const [filterModel, setFilterModel] = useState('');
  const [filterMinYear, setFilterMinYear] = useState('');
  const [filterMaxYear, setFilterMaxYear] = useState('');
  const [filterMinPrice, setFilterMinPrice] = useState('');
  const [filterMaxPrice, setFilterMaxPrice] = useState('');
  const [filterMinKm, setFilterMinKm] = useState('');
  const [filterMaxKm, setFilterMaxKm] = useState('');
  const [filterFuelType, setFilterFuelType] = useState('');
  const [filterTransmission, setFilterTransmission] =
useState('');
```

```
// Filtrelenmiş arabalar
  const filteredCars = cars.filter(car => {
    if (filterBrand && car.brand.toLowerCase() !==
filterBrand.toLowerCase()) return false;
    if (filterModel && car.model.toLowerCase() !==
filterModel.toLowerCase()) return false;
    if (filterMinYear && Number(car.year) <
Number(filterMinYear)) return false;
    if (filterMaxYear && Number(car.year) >
Number(filterMaxYear)) return false;
    if (filterMinPrice && Number(car.price) <
Number(filterMinPrice)) return false;
    if (filterMaxPrice && Number(car.price) >
Number(filterMaxPrice)) return false;
    if (filterMinKm && Number(car.km || car.mileage) <
Number(filterMinKm)) return false;
    if (filterMaxKm && Number(car.km || car.mileage) >
Number(filterMaxKm)) return false;
    if (filterFuelType && car.fuelType !== filterFuelType) return
false;
    if (filterTransmission && car.transmission !==
filterTransmission) return false;
    return true;
  });
```