Bugün e-ticaret uygulaması projesinin tanıtımını yaptım ve proje planlamasını gerçekleştirdim. Proje kapsamında modern mobil cihazlarda kullanıcı dostu alışveriş deneyimi sunan cross-platform bir e-ticaret uygulaması geliştirilmesi planladım. Bu süreçte ChatGpt[3] den de yardım aldım.

Kullanılan Diller ve Formatlar:
- **JavaScript:** Ana programlama dili
- **TypeScript:** Tip güvenliği için JavaScript superset'i
- **JSON:** Konfigürasyon dosyaları için

Geliştirilen Platform:
- **React Native:** Cross-platform mobil uygulama geliştirme framework'ü[4]
- **Expo:** React Native geliştirme platformu
- **Backend API:** Node.js ile Express.js

Veritabanı:
- **Firebase Firestore:** NoSQL cloud veritabanı[5]

Proje Yapısı: Ana dizin yapısını oluşturdum ve organize ettim:
/ECommerceApp
/app - React Native sayfa bileşenleri (Expo Router)
/components - Yeniden kullanılabilir UI bileşenleri
/hooks - Custom React hooks
/constants - Sabitler ve tema
/admin - React web uygulaması (admin paneli)
/server - Express.js backend API
/assets - Görseller ve statik dosyalar
/scripts - Utility scriptleri

Proje Özellikleri: E-ticaret uygulaması için temel özellikler planlandım:
- Kullanıcı kayıt ve giriş sistemi
- Ürün kataloğu ve arama
- Sepet yönetimi
- Sipariş takibi
- Admin paneli
- Real-time stok yönetimi

Teknoloji Entegrasyonu: React Native ile Expo platformunu entegre ederek cross-platform geliştirme ortamını kurdum. Firebase Firestore veritabanı ile backend servislerini planladım.

Geliştirme Ortamı: Expo CLI ile proje oluşturma sürecini başlattım. TypeScript konfigürasyonunu yapılandırdım ve proje klasör yapısını oluşturdum. Git repository'sini initialize ettim ve temel konfigürasyon dosyalarını hazırladım.

Öğrenilenler: React Native ile cross-platform mobil uygulama geliştirme süreçleri, Expo platformunun avantajları, Firebase Firestore NoSQL veritabanı yapısı ve modern JavaScript/TypeScript geliştirme ortamları hakkında kapsamlı bilgi sahibi oldum.

Karşılaşılan Zorluklar: Proje yapısının planlanması ve React Native ile Expo arasındaki entegrasyon konusunda karar verme süreci zaman aldı. TypeScript konfigürasyonu ve path

mapping ayarlarında bazı zorluklar yaşadım. Firebase Firestore ile React Native entegrasyonu konusunda araştırma yapmam gerekti.
Firebase projesini oluşturdum ve Firestore veritabanını kurdum. Firebase konfigürasyonunu gerçekleştirdim ve veri modellerini tasarladım.

Firebase Proje Kurulumu: Firebase Console'da yeni proje oluşturdum ve proje ayarlarını yapılandırdım. Proje ID'si "ecommerceapp-5c73a" olarak belirlendi. Firebase Authentication, Firestore Database servisini etkinleştirdim.

Firebase Konfigürasyonu: firebaseConfig.js dosyasını oluşturdum ve Firebase SDK'sını projeye entegre ettim. API key'leri ve proje ayarlarını güvenli şekilde yapılandırdım.

Veri Modelleri Tasarımı: E-ticaret uygulaması için gerekli veri modellerini tasarladım:
- **Product:** Ürün bilgileri (id, name, description, price, category, imageUrl, stock)
- **User:** Kullanıcı bilgileri (id, name, email, role)
- **Order:** Sipariş bilgileri (id, userId, items, totalPrice, status)
- **CartItem:** Sepet ürünleri (product, quantity, reservedAt, expiresAt)

Firestore Koleksiyonları: Veritabanı yapısını tasarladım ve şu koleksiyonları oluşturdum:
- products: Ürün bilgileri
- users: Kullanıcı bilgileri
- orders: Sipariş bilgileri
- carts: Sepet bilgileri
- stock_reservations: Stok rezervasyonları

Firestore Güvenlik Kuralları: Veritabanı güvenliği için Firestore güvenlik kurallarını yapılandırdım. Kullanıcıların sadece kendi verilerine erişebilmesini, admin kullanıcıların tüm verilere erişebilmesini sağladım.

Firebase Admin SDK: Admin paneli için Firebase Admin SDK konfigürasyonunu hazırladım. Service account key'ini güvenli şekilde yapılandırdım.

Veritabanı İndeksleri: Firestore'da performans için gerekli indeksleri oluşturdum. products koleksiyonu için createdAt field indeksi, orders koleksiyonu için userId ve createdAt field indeksi ekledim.

Real-time Listeners: Firestore real-time listeners için temel yapıyı hazırladım. Ürün değişikliklerini anlık olarak dinleyecek sistem kurulumunu tamamladım.

Öğrenilenler: Firebase ekosistemi, NoSQL veritabanı yapısı, Firestore güvenlik kuralları, real-time database kavramları, Firebase Admin SDK kullanımı ve veritabanı indeksleme konularında kapsamlı bilgi sahibi oldum.

Karşılaşılan Zorluklar: Firebase konfigürasyonunda API key'lerin güvenli şekilde yönetilmesi konusunda araştırma yapmam gerekti. Firestore güvenlik kurallarının yazımında syntax hataları yaşadım. Real-time listeners konfigürasyonunda memory leak önleme konusunda zorluklar yaşadım.

app/*layout.tsx dosyasını oluşturdum ve Expo Router ile navigation yapısını kurdum. Tab navigation'ı implement ettim ve Context Provider'ları ekledim. Tüm uygulama için temel navigation yapısını oluşturdum.*

Ana Layout Yapısı: app/*layout.tsx dosyasını oluşturdum ve temel navigation yapısını kurdum. ThemeProvider, AuthProvider ve CartProvider'ları sarmalayarak tüm uygulamada kullanılabilir hale getirdim. Bu sayede authentication ve sepet state'ine tüm sayfalardan erişim sağlandı.*

Expo Router Konfigürasyonu: File-based routing sistemi ile sayfa yapısını oluşturdum. Bu sistem sayesinde klasör yapısı otomatik olarak route'ları oluşturuyor. Ana sayfalar için şu yapıyı kurdum:

- (tabs) - Ana tab navigation
- shop/index - Mağaza sayfası
- cart/index - Sepet sayfası
- product/[id] - Dinamik ürün detay sayfası
- orders/index - Siparişler sayfası
- checkout/index - Ödeme sayfası
- login - Giriş sayfası
- register - Kayıt sayfası

Tab Navigation: app/(tabs)/*layout.tsx dosyasını oluşturdum ve tab navigation yapısını kurdum. Ana Sayfa ve Keşfet sekmelerini ekledim. Tab icon'larını ve renklerini yapılandırdım. Her tab için uygun icon'lar seçtim ve aktif/pasif durumlar için renk ayarlarını yaptım.*

Context Provider'lar: AuthProvider ve CartProvider'ları ana layout'a ekledim. Bu sayede tüm uygulamada authentication ve sepet state'ine erişim sağlandı. Provider'ların sıralaması önemli olduğu için doğru hiyerarşiyi kurdum.

Navigation Stack: Stack navigation yapısını kurdum ve her sayfa için screen options'ları belirledim. Header'ları gizledim ve custom header'lar kullanmaya karar verdim. Bu sayede her sayfada tutarlı bir görünüm sağladım.

Screen Options: Her sayfa için uygun screen options'ları belirledim. Header'ları gizleyerek custom header'lar kullanma kararı aldım. Bu sayede daha esnek bir tasarım elde ettim.

Öğrenilenler: React Navigation, Context API kullanımı, component composition, Expo Router file-based routing sistemi, navigation yapısı planlama, provider hierarchy ve screen options konularında kapsamlı deneyim kazandım.

Karşılaşılan Zorluklar: Navigation yapısının karmaşıklığı nedeniyle routing hataları yaşadım, ancak Expo Router dokümantasyonu ile çözdüm. Context Provider'ların sıralaması konusunda sorunlar yaşadım ve doğru hiyerarşiyi kurmak için araştırma yapmam gerekti. Tab navigation'da icon'ların doğru şekilde görüntülenmesi konusunda zorluklar yaşadım.

useAuth hook'unu oluşturdum ve kullanıcı kimlik doğrulama sistemini geliştirdim. Login ve Register sayfalarını implement ettim. Kullanıcı state management'ını kurarak tüm uygulamada authentication sistemi oluşturdum.

Authentication Hook Geliştirme: hooks/useAuth.tsx dosyasını oluşturdum ve kullanıcı state management'ını implement ettim. Firestore'da kullanıcı sorgulama, giriş yapma ve çıkış yapma fonksiyonlarını geliştirdim. Context API kullanarak global state yönetimi sağladım.

Login Sayfası Implementasyonu: app/login.tsx dosyasını oluşturdum ve kullanıcı giriş formunu implement ettim. Email ve şifre alanları, giriş butonu ve hata mesajları ekledim. Form validasyonu ve kullanıcı deneyimi için gerekli kontrolleri ekledim.

Register Sayfası Geliştirme: app/register.tsx dosyasını oluşturdum ve kullanıcı kayıt formunu implement ettim. Ad, email ve şifre alanları, kayıt butonu ve validasyon kontrolleri ekledim. Şifre tekrarı kontrolü ve email format validasyonu ekledim.

User State Management: Kullanıcı bilgilerini AsyncStorage'da saklama sistemi kurdum. Uygulama açıldığında kullanıcı bilgilerini yükleme ve oturum sürekliliği sağlama işlemlerini implement ettim. Kullanıcı çıkış yaptığında verilerin temizlenmesi işlemini ekledim.

Protected Routes Sistemi: Kullanıcı girişi gerektiren sayfalar için koruma sistemi kurdum. Giriş yapmamış kullanıcıları login sayfasına yönlendirme mekanizması geliştirdim. Bu sayede güvenli sayfalara erişim kontrolü sağladım.

Firestore Authentication: Firestore'da kullanıcı verilerini saklama ve sorgulama sistemi kurdum. Email ve şifre eşleşmesi kontrolü, kullanıcı rolü yönetimi ve güvenlik kontrolleri ekledim. Kullanıcı kayıt işleminde email benzersizlik kontrolü implement ettim.

AsyncStorage Entegrasyonu: Kullanıcı oturum bilgilerini AsyncStorage'da güvenli şekilde saklama sistemi kurdum. Uygulama yeniden açıldığında kullanıcı bilgilerini otomatik yükleme, çıkış yapıldığında verileri temizleme işlemlerini implement ettim.

Ürün veri modelini tasarladım ve Firestore'da products koleksiyonunu oluşturdum. Ürün listeleme API'sini geliştirdim. Ürün arama ve filtreleme sistemini implement ettim.

Ürün Veri Modeli Tasarımı:constants/Api.ts dosyasını oluşturdum ve Product type definition'ını tanımladım. Ürün bilgileri için gerekli alanları belirledim: id, name, description, price, category, imageUrl, imageUrls, stock, createdAt. TypeScript ile tip güvenliği sağladım.

Firestore API Fonksiyonları:Ürün listeleme ve detay getirme fonksiyonlarını geliştirdim. getProducts() fonksiyonu ile tüm ürünleri tarih sırasına göre getirme, getProduct(id) fonksiyonu ile tek ürün detayını getirme işlemlerini implement ettim.

Ürün Arama Sistemi:Ürün adı ve kategori bazında arama fonksiyonunu implement ettim. Kullanıcıların ürün adı veya kategori ile arama yapabilmesini sağladım. Case-insensitive arama özelliği ekledim.

Kategori Filtreleme:Ürünleri kategori bazında filtreleme sistemi kurdum. Tüm kategorileri otomatik olarak tespit etme ve filtreleme seçenekleri sunma işlemini implement ettim.

Real-time Updates: Firestore real-time listeners ile ürün değişikliklerini anlık olarak dinleme sistemi kurdum. Ürün stok değişiklikleri, yeni ürün ekleme ve ürün güncelleme işlemlerini real-time olarak takip ettim.

Image Handling:Ürün görsellerini yönetme sistemi kurdum. Tek görsel ve çoklu görsel desteği ekledim. Görsel yükleme hatalarında fallback placeholder sistemi implement ettim.

Stock Management:Ürün stok yönetimi için temel yapıyı kurdum. Stok durumu kontrolü, stok azalma ve artırma işlemleri için gerekli fonksiyonları hazırladım.

Error Handling:API çağrılarında oluşabilecek hataları yakalama sistemi kurdum. Network hataları, Firestore hataları ve veri format hataları için uygun error handling implement ettim.

Performance Optimization:Ürün listesi için pagination sistemi planladım. Büyük veri setlerinde performans sorunlarını önlemek için lazy loading ve virtual scrolling hazırlığı yaptım.

Öğrenilenler: NoSQL veri modelleme, Firestore query yapısı, API design patterns, real-time data management, image handling, stock management, error handling ve performance optimization konularında kapsamlı deneyim kazandım.

Karşılaşılan Zorluklar: Firestore query'lerinde orderBy kullanımında bazı index sorunları yaşadım, ancak Firestore console'dan index oluşturarak çözdüm. Real-time listeners'da memory leak önleme konusunda zorluklar yaşadım ve cleanup fonksiyonları ile çözdüm.

Ana sayfa tasarımını oluşturdum ve ürün listeleme component'ini geliştirdim. Arama fonksiyonunu implement ettim ve kategori filtreleme özelliğini ekledim. Kullanıcı deneyimini optimize etmek için responsive tasarım uyguladım.

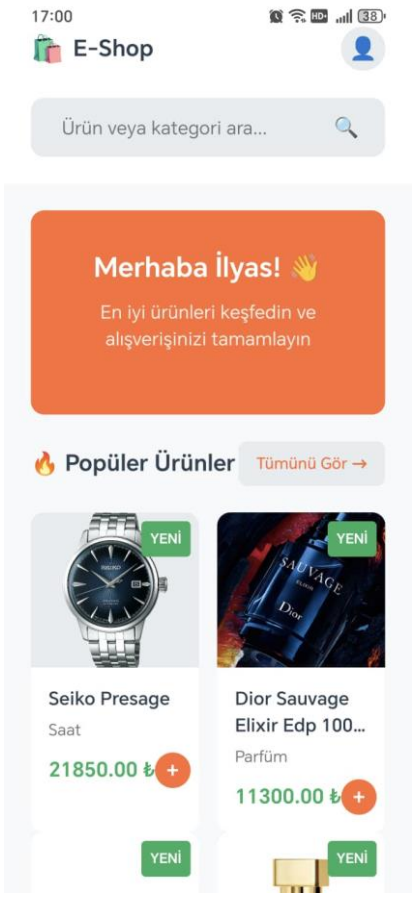
Ana Sayfa Layout Tasarımı:app/(tabs)/index.tsx dosyasını geliştirdim ve ana sayfa layout'unu oluşturdum. Header bölümünde logo, arama çubuğu ve profil butonu ekledim. Hero section'da kullanıcı karşılama mesajı ve alışverişe başlama butonları yerleştirdim.

Ürün Listeleme Component'i:Ürün listesini grid formatında gösterme sistemi kurdum. FlatList component'i kullanarak 2 sütunlu grid layout oluşturdum. Her ürün için ProductCard component'i tasarladım ve ürün bilgilerini görüntüleme sistemi implement ettim.

Arama Fonksiyonu: Kullanıcıların ürün adı ve kategori bazında arama yapabilmesi için arama çubuğu ekledim. Real-time arama özelliği ile kullanıcı yazdıkça sonuçların filtrelenmesi sağlandı. useMemo hook'u ile performans optimizasyonu yaptım.

Kategori Filtreleme:Ürünleri kategori bazında filtreleme sistemi kurdum. Tüm kategorileri otomatik olarak tespit etme ve horizontal scroll ile kategori seçenekleri sunma işlemini implement ettim. Seçili kategori vurgulama sistemi ekledim.

ProductCard Component'i: Yeniden kullanılabilir ürün kartı component'i oluşturdum. Ürün resmi, adı, kategorisi, fiyatı ve sepete ekleme butonu içeren kapsamlı tasarım yaptım. Stok durumu gösterimi ve stok yoksa buton disable etme özelliği ekledim.



Loading States:Ürün verileri yüklenirken loading indicator'ı ekledim. Kullanıcı deneyimini iyileştirmek için skeleton loading ve error states implement ettim. Network hatalarında kullanıcıya uygun mesajlar gösterdim.

Responsive Tasarım: Farklı ekran boyutları için responsive tasarım uyguladım. Tablet ve telefon ekranları için uygun grid layout'ları ayarladım. Safe area handling ile notch'lu cihazlarda uyumlu görünüm sağladım.

Performance Optimization:useMemo ve useCallback hook'ları ile component re-render'larını optimize ettim. FlatList'in getItemLayout prop'unu kullanarak scroll performansını iyileştirdim. Image lazy loading sistemi ekledim.

Error Handling:API hatalarında kullanıcıya uygun error mesajları gösterme sistemi kurdum. Network bağlantı sorunlarında retry mekanizması ekledim. Empty state'ler için kullanıcı dostu mesajlar tasarladım.

User Experience: Kullanıcı deneyimini iyileştirmek için haptic feedback ekledim. Sepete ekleme işlemlerinde success feedback'i implement ettim. Pull-to-refresh özelliği ile sayfa yenileme sistemi kurdum.

Öğrenilenler: Component lifecycle, API integration, state management patterns, responsive design, performance optimization, error handling, user experience design ve React Native best practices konularında kapsamlı deneyim kazandım.

Karşılaşılan Zorluklar:useMemo hook'unun doğru kullanımı konusunda araştırma yapmam gerekti, performans optimizasyonu için önemli olduğunu öğrendim. FlatList component'inde scroll performansı sorunları yaşadım, getItemLayout kullanımı ile optimize ettim. Responsive tasarımda farklı ekran boyutları için uyum sağlama konusunda zorluklar yaşadım.

Ürün detay sayfasının tasarımını yaptım ve ürün galerisi component'ini oluşturdum. Miktar seçici component'ini geliştirdim ve stok durumu gösterimini implement ettim. Kullanıcı etkileşimini optimize etmek için detaylı UI/UX tasarımı uyguladım.

Ürün Detay Sayfası Layout:app/product/[id].tsx dosyasını oluşturdum ve dinamik routing ile ürün detay sayfasını implement ettim. useLocalSearchParams hook'u ile URL'den ürün ID'sini alarak ilgili ürün bilgilerini getirme sistemi kurdum. ScrollView ile uzun içerik desteği sağladım.

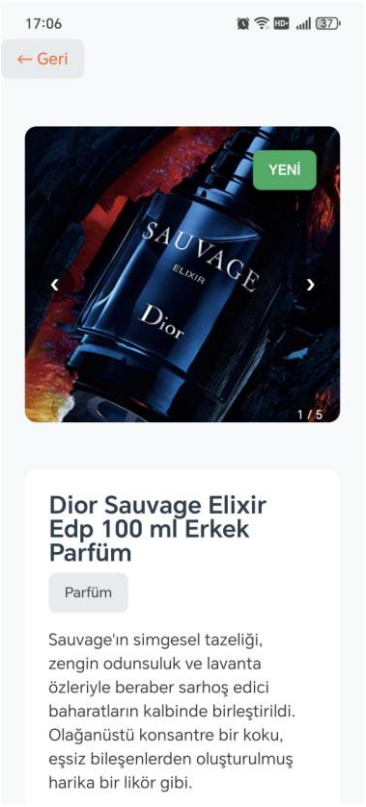
Ürün Galerisi Component'i: ProductImageGallery component'ini oluşturdum ve çoklu görsel desteği ekledim. Kullanıcıların ürün görsellerini kaydırarak inceleyebilmesi için image pan-zoom özelliği implement ettim. Görsel yükleme hatalarında fallback placeholder sistemi kurdum.

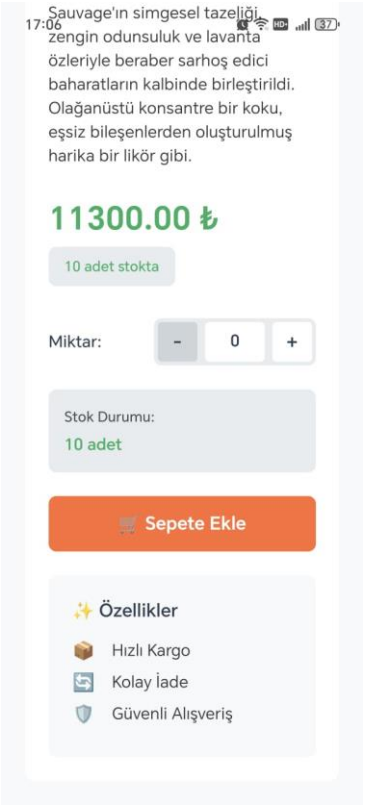
Miktar Seçici Sistemi: Kullanıcıların ürün miktarını seçebilmesi için quantity selector component'i geliştirdim. Artı/eksi butonları ve manuel input girişi desteği ekledim. Stok durumuna göre maksimum miktar sınırlaması implement ettim.

Stok Durumu Gösterimi:Ürün stok bilgilerini kullanıcıya net şekilde gösterme sistemi kurdum. Stok durumuna göre renk kodlaması (yeşil: stokta, kırmızı: stokta yok) uyguladım. Sepetteki mevcut miktarı da göstererek kullanıcı bilgilendirmesi sağladım.

Sepete Ekleme Fonksiyonu:Ürünü sepete ekleme işlemini implement ettim. Stok kontrolü yaparak yetersiz stok durumunda kullanıcıyı uyarma sistemi kurdum. Sepete ekleme başarılı olduğunda haptic feedback ve success mesajı ekledim.

Ürün Özellikleri Bölümü:Ürün hakkında detaylı bilgileri gösterme bölümü ekledim. Hızlı kargo, kolay iade, güvenli alışveriş gibi özellikleri icon'larla birlikte listeledim. Kullanıcı güvenini artırmak için güvenlik özelliklerini vurguladım.





Navigation ve Back Button:Ürün detay sayfasından geri dönüş için back button ekledim. router.back() fonksiyonu ile önceki sayfaya dönüş sistemi kurdum. Safe area handling ile notch'lu cihazlarda uyumlu görünüm sağladım.

Loading ve Error States:Ürün verileri yüklenirken loading indicator'ı ekledim. Ürün bulunamadığında veya hata oluştuğunda kullanıcıya uygun mesajlar gösterdim. Network hatalarında retry mekanizması implement ettim.

Responsive Tasarım: Farklı ekran boyutları için responsive tasarım uyguladım. Tablet ekranlarında daha geniş layout, telefon ekranlarında kompakt tasarım sağladım. Image aspect ratio'larını optimize ettim.

Performance Optimization: Component re-render'larını optimize etmek için React.memo kullanımı ekledim. Image loading'de lazy loading sistemi implement ettim. Memory leak'leri önlemek için cleanup fonksiyonları ekledim.

User Experience: Kullanıcı deneyimini iyileştirmek için smooth scrolling ve gesture handling ekledim. Ürün görsellerinde zoom özelliği ile detaylı inceleme imkanı sağladım. Haptic feedback ile buton etkileşimlerini geliştirdim.

Accessibility: Erişilebilirlik için screen reader desteği ekledim. Buton ve input'lar için accessibility label'ları tanımladım. Keyboard navigation desteği implement ettim.

Öğrenilenler: Dynamic routing, component composition, user interaction patterns, image handling, responsive design, performance optimization, accessibility ve user experience design konularında kapsamlı deneyim kazandım.

Karşılaşılan Zorluklar:useLocalSearchParams hook'unun kullanımında bazı tip sorunları yaşadım, TypeScript generic kullanımı ile çözdüm. Image pan-zoom component'inde gesture handling konusunda zorluklar yaşadım, react-native-image-pan-zoom kütüphanesi ile çözdüm. Stok kontrolü ve sepet entegrasyonunda state management konusunda araştırma yapmam gerekti.

useCart hook'unu oluşturdum ve sepet state management'ını implement ettim. Local storage ile sepet saklama ve Firestore ile senkronizasyon sistemini geliştirdim. Sepet işlemlerini (ekleme/çıkarma/güncelleme) kapsamlı şekilde implement ettim.

Sepet Hook Geliştirme:hooks/useCart.tsx dosyasını oluşturdum ve kapsamlı sepet yönetimi sistemi kurdum. CartProvider ile global state management sağladım. Sepet ürünlerini ekleme, çıkarma, güncelleme ve temizleme fonksiyonlarını implement ettim.

Local Storage Entegrasyonu: AsyncStorage kullanarak sepet verilerini cihazda saklama sistemi kurdum. Uygulama kapatılıp açıldığında sepet verilerinin korunmasını sağladım. JSON serialize/deserialize işlemlerini güvenli şekilde implement ettim.

Firestore Senkronizasyonu: Kullanıcı giriş yaptığında sepet verilerini Firestore'dan yükleme sistemi kurdum. Sepet değişikliklerini real-time olarak Firestore'a kaydetme mekanizması implement ettim. Kullanıcı çıkış yaptığında sepet verilerini temizleme sistemi ekledim.

Sepet Ürün Ekleme:Ürünleri sepete ekleme fonksiyonunu geliştirdim. Mevcut ürün varsa miktarını artırma, yoksa yeni ürün ekleme mantığını implement ettim. Stok kontrolü yaparak yetersiz stok durumunda kullanıcıyı uyarma sistemi kurdum.

Miktar Güncelleme: Sepetteki ürün miktarını güncelleme sistemi kurdum. Artı/eksi butonları ile miktar değiştirme, manuel input ile miktar girme özelliklerini implement ettim. Miktar 0 olduğunda ürünü sepetten çıkarma mantığını ekledim.

Sepet Ürün Çıkarma:Ürünleri sepetten çıkarma fonksiyonunu implement ettim. Tek ürün çıkarma ve tüm sepeti temizleme seçenekleri ekledim. Kullanıcı onayı alarak sepet temizleme işlemini güvenli hale getirdim.

Toplam Hesaplama: Sepet toplamını otomatik hesaplama sistemi kurdum. useMemo hook'u ile performans optimizasyonu sağladım. Her ürün için fiyat × miktar hesaplaması yaparak toplam tutarı güncel tutma sistemi implement ettim.

Real-time Stok Güncellemeleri: Firestore listeners ile stok değişikliklerini anlık takip etme sistemi kurdum. Diğer kullanıcıların sepet işlemleri nedeniyle stok azaldığında sepeti güncelleme mekanizması implement ettim.

Error Handling: Sepet işlemlerinde oluşabilecek hataları yakalama sistemi kurdum. Network hataları, stok yetersizliği, kullanıcı girişi gerektiren işlemler için uygun error mesajları ekledim.

Loading States: Sepet işlemleri sırasında loading state'leri ekledim. Async işlemler sırasında buton disable etme ve loading indicator'ları implement ettim. Kullanıcı deneyimini iyileştirmek için progress feedback'i ekledim.

Performance Optimization: Sepet state'ini optimize etmek için useCallback ve useMemo hook'larını kullandım. Gereksiz re-render'ları önlemek için component memoization uyguladım. Memory leak'leri önlemek için cleanup fonksiyonları ekledim.

User Experience: Sepet işlemlerinde haptic feedback ekledim. Başarılı işlemlerde success mesajları, hatalı işlemlerde uyarı mesajları gösterdim. Sepet boş olduğunda kullanıcıyı alışverişe yönlendirme sistemi kurdum.

Öğrenilenler: Complex state management, local storage patterns, real-time synchronization, error handling, loading states, performance optimization ve user experience design konularında kapsamlı deneyim kazandım.

Karşılaşılan Zorluklar: Context API ile state management'da re-render sorunları yaşadım, useMemo ve useCallback kullanımı ile optimize ettim. AsyncStorage'da veri saklama ve yükleme konusunda timing sorunları yaşadım, useEffect dependency array'i ile çözdüm.

Stok Rezervasyon API:constants/Api.ts dosyasında reserveProductStock fonksiyonunu geliştirdim. Sepete ürün eklendiğinde stoktan düşme işlemini implement ettim. Rezervasyon kaydı oluşturma ve 10 dakikalık süre sınırı ekleme sistemi kurdum.

Rezervasyon Süresi Yönetimi: Stok rezervasyonları için expiresAt alanı ekledim. JavaScript Date objesi ile 10 dakikalık süre hesaplama sistemi kurdum. Rezervasyon süresi dolduğunda otomatik iptal mekanizması implement ettim.

Otomatik Rezervasyon İptali:Süresi dolmuş rezervasyonları tespit etme sistemi kurdum. cleanupExpiredReservations fonksiyonu ile batch işlemler yaparak performanslı temizleme sistemi implement ettim. Stokları geri ekleme ve rezervasyon durumunu güncelleme işlemlerini ekledim.

Stok Çakışması Önleme: Aynı ürün için eş zamanlı rezervasyon işlemlerinde çakışma önleme sistemi kurdum. Firestore transaction kullanarak atomic işlemler sağladım. Race condition sorunlarını önlemek için optimistic locking implement ettim.

Real-time Stok Güncellemeleri: Firestore listeners ile stok değişikliklerini anlık takip etme sistemi kurdum. onProductStockChange fonksiyonu ile tek ürün stok takibi, onAllProductsStockChange ile tüm ürünlerin stok takibi implement ettim.

Rezervasyon İptal Sistemi:cancelStockReservation fonksiyonu ile rezervasyon iptal etme sistemi kurdum. Stokları geri ekleme ve rezervasyon durumunu güncelleme işlemlerini implement ettim. Kullanıcı sepetten ürün çıkardığında otomatik iptal sistemi ekledim.

Batch İşlemler:Süresi dolmuş rezervasyonları toplu işleme sistemi kurdum. Firestore batch operations kullanarak performanslı güncelleme sistemi implement ettim. writeBatch ile atomik işlemler sağladım.

Stok Durumu Kontrolü:checkProductStock fonksiyonu ile stok durumu kontrolü sistemi kurdum. Rezervasyon öncesi stok yeterliliği kontrolü implement ettim. Yetersiz stok durumunda kullanıcıyı uyarma sistemi ekledim.

Rezervasyon Durumu Takibi:stock_reservations koleksiyonunda rezervasyon durumlarını takip etme sistemi kurdum. 'active', 'expired', 'cancelled' durumları ile rezervasyon lifecycle'ını yönetme sistemi implement ettim.

Performance Optimization: Rezervasyon işlemlerinde performans optimizasyonu sağladım. Batch işlemler ile Firestore write işlemlerini minimize ettim. Index kullanımı ile query performansını optimize ettim.

Error Handling: Rezervasyon işlemlerinde oluşabilecek hataları yakalama sistemi kurdum. Network hataları, stok yetersizliği, rezervasyon çakışması durumları için uygun error handling implement ettim.

Memory Management: Real-time listeners'da memory leak önleme sistemi kurdum. Component unmount olduğunda listener'ları temizleme mekanizması implement ettim. Cleanup fonksiyonları ile resource management sağladım.

Concurrency Control:Eş zamanlı rezervasyon işlemlerinde concurrency control sistemi kurdum. Firestore transaction kullanarak atomic işlemler sağladım. Optimistic locking ile çakışma önleme mekanizması implement ettim.

Öğrenilenler: Complex business logic, concurrency control, real-time data management, batch operations, transaction handling, performance optimization, memory management ve error handling konularında kapsamlı deneyim kazandım.

Karşılaşılan Zorluklar: Stok rezervasyon sisteminde race condition sorunları yaşadım, Firestore transaction kullanımı ile çözdüm. Real-time listeners'da memory leak önleme konusunda zorluklar yaşadım, cleanup fonksiyonları ile çözdüm. Batch işlemlerde performance sorunları yaşadım, writeBatch kullanımı ile optimize ettim.

Sepet sayfasının tasarımını yaptım ve sepet ürünlerinin listelenmesini implement ettim. Miktar güncelleme arayüzünü geliştirdim ve toplam hesaplama fonksiyonunu implement ettim. Rezervasyon süresi gösterimini ekledim ve kullanıcı deneyimini optimize ettim.

Sepet Sayfası Layout:app/cart/index.tsx dosyasını oluşturdum ve sepet sayfasının temel layout'unu tasarladım. Header bölümünde sayfa başlığı, ürün sayısı ve geri dönüş butonu ekledim. ScrollView ile uzun sepet listesi desteği sağladım.

Sepet Ürün Listesi: FlatList component'i kullanarak sepet ürünlerini listeleme sistemi kurdum. Her ürün için ayrı cart item component'i tasarladım. Ürün resmi, adı, kategorisi, fiyatı ve miktar bilgilerini görüntüleme sistemi implement ettim.

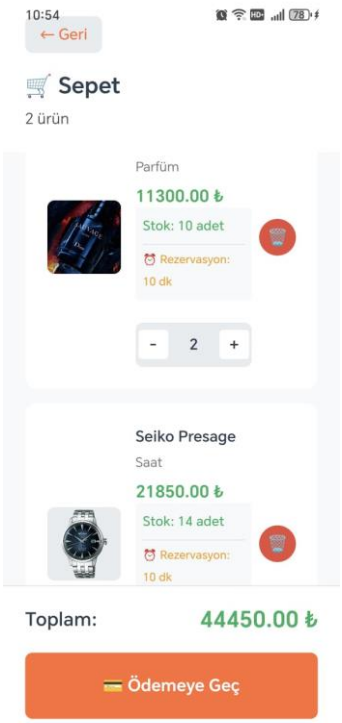
Miktar Güncelleme Arayüzü: Sepetteki ürün miktarını güncelleme için quantity selector component'i geliştirdim. Artı/eksi butonları ile miktar değiştirme, manuel input ile miktar girme özelliklerini implement ettim. Stok sınırlarına göre buton disable etme sistemi ekledim.

Toplam Hesaplama Sistemi: Sepet toplamını otomatik hesaplama sistemi kurdum. Her ürün için fiyat × miktar hesaplaması yaparak toplam tutarı güncel tutma sistemi implement ettim. useMemo hook'u ile performans optimizasyonu sağladım.

Rezervasyon Süresi Gösterimi: Sepetteki ürünlerin rezervasyon süresini kullanıcıya gösterme sistemi kurdum. Kalan süreyi dakika cinsinden hesaplayarak real-time güncelleme sağladım. Süre dolduğunda kullanıcıyı uyarma sistemi implement ettim.

Stok Durumu Gösterimi: Sepetteki ürünlerin stok durumunu görüntüleme sistemi kurdum. Stokta olan ürünler için yeşil, stokta olmayan ürünler için kırmızı renk kodlaması uyguladım. Stok miktarını sayısal olarak gösterdim.

Sepet Boş Durumu: Sepet boş olduğunda kullanıcıya uygun mesaj gösterme sistemi kurdum. Alışverişe yönlendirme butonu ekledim. Empty state için kullanıcı dostu tasarım uyguladım.



Giriş Gerekli Durumu: Kullanıcı giriş yapmamışsa sepeti görüntüleme engelleme sistemi kurdum. Giriş yapmaya yönlendirme butonu ekledim. Authentication kontrolü ile güvenli erişim sağladım.

Ürün Çıkarma Sistemi: Sepetten ürün çıkarma fonksiyonunu implement ettim. Tek ürün çıkarma ve tüm sepeti temizleme seçenekleri ekledim. Kullanıcı onayı alarak ürün çıkarma işlemini güvenli hale getirdim.

Loading States: Sepet verileri yüklenirken loading indicator'ı ekledim. Async işlemler sırasında buton disable etme sistemi implement ettim. Kullanıcı deneyimini iyileştirmek için progress feedback'i ekledim.

Error Handling: Sepet işlemlerinde oluşabilecek hataları yakalama sistemi kurdum. Network hataları, stok yetersizliği, kullanıcı girişi gerektiren işlemler için uygun error mesajları ekledim.

Performance Optimization: FlatList'in getItemLayout prop'unu kullanarak scroll performansını iyileştirdim. Component re-render'larını optimize etmek için React.memo kullandım. Memory leak'leri önlemek için cleanup fonksiyonları ekledim.

User Experience: Sepet işlemlerinde haptic feedback ekledim. Başarılı işlemlerde success mesajları, hatalı işlemlerde uyarı mesajları gösterdim. Smooth scrolling ve gesture handling ile kullanıcı deneyimini iyileştirdim.

Responsive Tasarım: Farklı ekran boyutları için responsive tasarım uyguladım. Tablet ekranlarında daha geniş layout, telefon ekranlarında kompakt tasarım sağladım. Safe area handling ile notch'lu cihazlarda uyumlu görünüm sağladım.

Öğrenilenler: Complex UI components, user interaction patterns, state synchronization, loading states, error handling, performance optimization, responsive design ve user experience design konularında kapsamlı deneyim kazandım.

Karşılaşılan Zorluklar: FlatList component'inde scroll performansı sorunları yaşadım, getItemLayout kullanımı ile optimize ettim. Miktar güncelleme işlemlerinde state management konusunda zorluklar yaşadım, controlled component pattern'i ile çözdüm. Rezervasyon süresi hesaplamalarında timing sorunları yaşadım, useEffect ve setInterval kullanımı ile çözdüm.

Checkout sayfasının tasarımını yaptım ve sipariş özeti component'ini oluşturdum. Ödeme bilgileri formunu geliştirdim ve sipariş oluşturma API'sini implement ettim. Sipariş onay sürecini geliştirdim ve kullanıcı deneyimini optimize ettim.

Checkout Sayfası Layout:app/checkout/index.tsx dosyasını oluşturdum ve ödeme sayfasının temel layout'unu tasarladım. Header bölümünde sayfa başlığı ve geri dönüş butonu ekledim. ScrollView ile uzun içerik desteği sağladım.

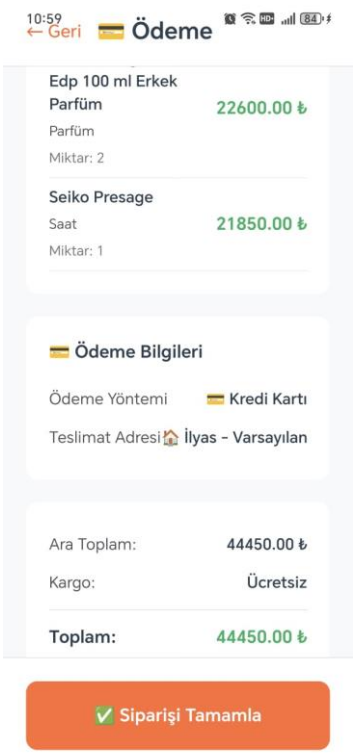
Sipariş Özeti Component'i: Sepetteki ürünleri sipariş özeti formatında gösterme sistemi kurdum. Her ürün için ürün adı, kategorisi, miktarı ve toplam fiyatı görüntüleme sistemi implement ettim. Sipariş detaylarını kullanıcıya net şekilde sunma tasarımı uyguladım.

Ödeme Bilgileri Formu: Kullanıcı ödeme bilgilerini girebileceği form tasarladım. Kredi kartı bilgileri, teslimat adresi ve iletişim bilgileri alanları ekledim. Form validasyonu ve kullanıcı girişi kontrolleri implement ettim.

Sipariş Oluşturma API:createOrder fonksiyonunu geliştirdim ve sipariş oluşturma sistemini implement ettim. Kullanıcı bilgileri, ürün listesi, toplam fiyat ve sipariş durumu ile sipariş kaydı oluşturma sistemi kurdum.

Sipariş Onay Süreci: Sipariş oluşturma işlemini onaylama sistemi kurdum. Kullanıcı onayı alarak sipariş verme işlemini güvenli hale getirdim. Sipariş başarılı olduğunda sepeti temizleme ve siparişler sayfasına yönlendirme sistemi implement ettim.

Toplam Hesaplama: Sipariş toplamını hesaplama sistemi kurdum. Ara toplam, kargo ücreti ve nihai toplam hesaplamalarını implement ettim. Kargo ücretsiz olduğu için sadece ürün toplamını gösterdim.



Loading States: Sipariş oluşturma işlemi sırasında loading state'leri ekledim. Buton disable etme ve loading indicator'ları implement ettim. Kullanıcı deneyimini iyileştirmek için progress feedback'i ekledim.

Error Handling: Sipariş işlemlerinde oluşabilecek hataları yakalama sistemi kurdum. Network hataları, kullanıcı girişi gerektiren işlemler, sepet boş durumu için uygun error mesajları ekledim.

Form Validation:Ödeme formunda validasyon kontrolleri implement ettim. Gerekli alanların doldurulması, email format kontrolü, telefon numarası format kontrolü ekledim.

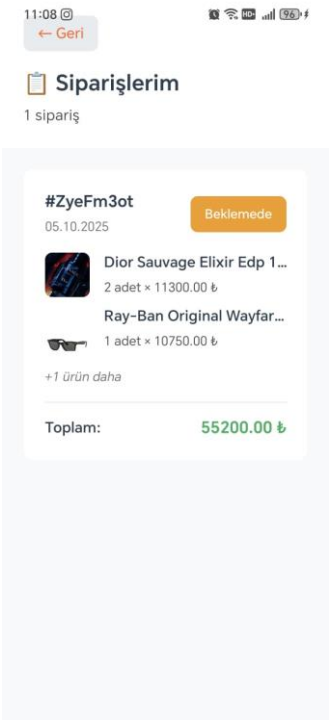
Success Feedback: Sipariş başarıyla oluşturulduğunda kullanıcıya success mesajı gösterme sistemi kurdum. Sipariş numarası ve tahmini teslimat tarihi bilgilerini kullanıcıya sunma sistemi implement ettim.

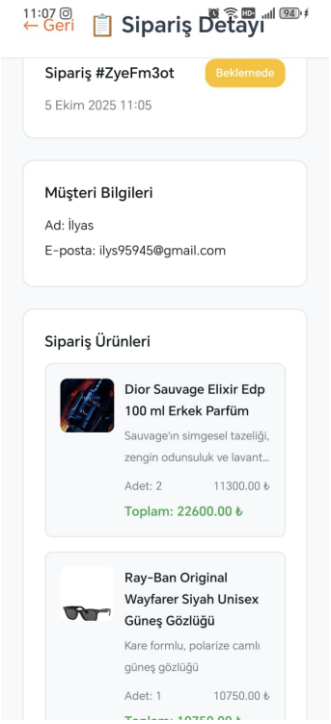
Navigation Flow: Sipariş tamamlandıktan sonra kullanıcıyı siparişler sayfasına yönlendirme sistemi kurdum. router.replace() kullanarak geri dönüşü engelleme sistemi implement ettim.

Sipariş listesi sayfasını oluşturdum ve sipariş detay sayfasını geliştirdim. Sipariş durumu gösterimini implement ettim ve sipariş geçmişi API'sini geliştirdim. Sipariş filtreleme özelliğini ekledim.

Sipariş Listesi Sayfası:app/orders/index.tsx dosyasını oluşturdum ve sipariş listesi sayfasını implement ettim. Kullanıcının siparişlerini tarih sırasına göre listeleme sistemi kurdum. Her sipariş için sipariş numarası, tarih, toplam fiyat ve durum bilgilerini görüntüleme sistemi implement ettim.

Sipariş Detay Sayfası:app/order-detail/[id].tsx dosyasını oluşturdum ve dinamik routing ile sipariş detay sayfasını implement ettim. Sipariş ürünlerini listeleme, sipariş bilgilerini gösterme ve sipariş durumu takibi sistemi kurdum.





Sipariş API Geliştirme:getUserOrders ve getOrderById fonksiyonlarını implement ettim. Firestore'dan kullanıcı siparişlerini getirme ve sipariş detaylarını yükleme sistemi kurdum. Real-time sipariş güncellemelerini takip etme mekanizması ekledim.

Sipariş Durumu Gösterimi: Sipariş durumlarını görsel olarak gösterme sistemi kurdum. 'pending', 'processing', 'shipped', 'delivered' durumları için renk kodlaması uyguladım. Durum değişikliklerini real-time takip etme sistemi implement ettim.

Sipariş Filtreleme: Siparişleri durum bazında filtreleme sistemi kurdum. Tüm siparişler, bekleyen siparişler, işlenen siparişler gibi filtre seçenekleri ekledim. Kullanıcı deneyimini iyileştirmek için dropdown menu implement ettim.

Loading ve Error States: Sipariş verileri yüklenirken loading indicator'ı ekledim. Sipariş bulunamadığında veya hata oluştuğunda kullanıcıya uygun mesajlar gösterdim. Network hatalarında retry mekanizması implement ettim.

Öğrenilenler: Data fetching patterns, list management, user experience optimization, real-time updates, filtering systems ve error handling konularında deneyim kazandım.

Karşılaşılan Zorluklar: Firestore query'lerinde performans sorunları yaşadım, index oluşturarak ve pagination kullanarak çözdüm. Real-time listeners'da memory leak önleme konusunda zorluklar yaşadım, cleanup fonksiyonları ile çözdüm.

Kullanıcı profil sayfasının tasarımını yaptım ve ayarlar sayfasını oluşturdum. Hesap yönetimi özelliklerini implement ettim ve bildirim ayarlarını geliştirdim. Yardım ve iletişim sayfalarını oluşturdum.
Profil Sayfası:app/profile/index.tsx dosyasını oluşturdum ve kullanıcı profil sayfasını implement ettim. Kullanıcı bilgilerini görüntüleme, siparişlerim, ayarlar ve çıkış yapma seçenekleri ekledim. Kullanıcı deneyimini iyileştirmek için modern tasarım uyguladım.

Ayarlar Sayfası:app/settings/index.tsx dosyasını oluşturdum ve ayarlar ana sayfasını implement ettim. Hesap ayarları, bildirimler, yardım ve iletişim seçenekleri ekledim. Her ayar için ayrı sayfa yapısı kurdum.

Hesap Ayarları:app/settings/account.tsx dosyasını oluşturdum ve hesap yönetimi sayfasını implement ettim. Kullanıcı bilgilerini düzenleme, şifre değiştirme ve hesap silme seçenekleri ekledim.

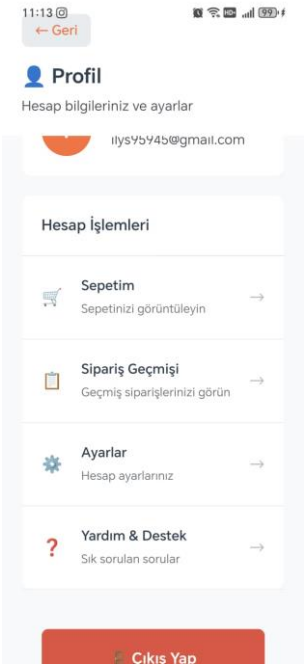
Bildirim Ayarları:app/settings/notifications.tsx dosyasını oluşturdum ve bildirim ayarları sayfasını implement ettim. Push notification, email bildirimleri ve SMS bildirimleri için toggle seçenekleri ekledim.

Yardım Sayfası:app/settings/help.tsx dosyasını oluşturdum ve yardım sayfasını implement ettim. Sık sorulan sorular, kullanım kılavuzu ve teknik destek seçenekleri ekledim.

İletişim Sayfası:app/settings/contact.tsx dosyasını oluşturdum ve iletişim sayfasını implement ettim. İletişim formu, telefon numarası, email adresi ve sosyal medya linkleri ekledim.

Öğrenilenler: User profile management, settings interface design, navigation structure, form handling ve user experience design konularında deneyim kazandım.

Karşılaşılan Zorluklar: Navigation yapısında nested routing konusunda sorunlar yaşadım, Expo Router dokümantasyonu ile çözdüm. Form validation'da state management konusunda zorluklar yaşadım, controlled component pattern'i ile çözdüm.



Admin paneli React web uygulamasını oluşturdum ve admin authentication sistemini tasarladım. Admin layout component'ini geliştirdim ve protected route'ları oluşturdum. Admin dashboard'unu tasarladım.

Admin Panel Kurulumu:admin/ klasöründe React web uygulaması oluşturdum. TypeScript, React Router ve Firebase Admin SDK entegrasyonu ile modern web uygulaması geliştirdim. Package.json bağımlılıklarını yapılandırdım.

Admin Authentication:useAdminAuth hook'unu oluşturdum ve admin kullanıcı kimlik doğrulama sistemini implement ettim. Firebase Admin SDK ile güvenli admin girişi sağladım. Role-based access control sistemi kurdum.

Admin Layout: AdminLayout component'ini oluşturdum ve admin paneli için temel layout'u tasarladım. Navigation menüsü, sidebar ve main content alanlarını organize ettim. Responsive tasarım ile farklı ekran boyutlarına uyum sağladım.

Protected Routes: Admin sayfaları için koruma sistemi kurdum. Kullanıcı girişi gerektiren sayfalar için authentication kontrolü implement ettim. Yetkisiz erişim durumunda login sayfasına yönlendirme sistemi ekledim.

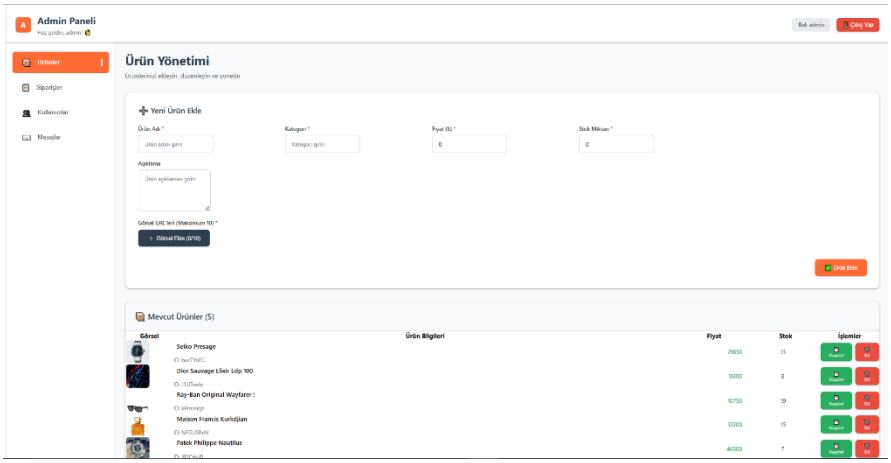
Navigation Sistemi: Admin paneli için navigation menüsü oluşturdum. Ürünler, siparişler, kullanıcılar ve dashboard seçenekleri ekledim. Active state gösterimi ile kullanıcı deneyimini iyileştirdim.

Dashboard Tasarımı: Admin dashboard'unu tasarladım. Genel istatistikler, son siparişler, stok durumu ve kullanıcı aktiviteleri için widget'lar ekledim. Real-time data gösterimi için Firestore listeners implement ettim.

Admin ürün listesi sayfasını geliştirdim ve ürün ekleme formunu oluşturdum. Ürün düzenleme işlemini implement ettim ve ürün silme fonksiyonunu geliştirdim. Toplu ürün işlemlerini ekledim.

Ürün Listesi Sayfası:admin/src/pages/Products.tsx dosyasını oluşturdum ve admin ürün listesi sayfasını implement ettim. Tüm ürünleri tablo formatında gösterme sistemi kurdum. Arama, filtreleme ve sayfalama özellikleri ekledim.

Ürün Ekleme Formu: Yeni ürün ekleme formunu oluşturdum. Ürün adı, açıklama, fiyat, kategori, stok ve görsel bilgileri için input alanları ekledim. Form validasyonu ve hata kontrolü implement ettim.



Ürün Düzenleme: Mevcut ürünleri düzenleme sistemi kurdum. Ürün bilgilerini form'a yükleme ve güncelleme işlemlerini implement ettim. Real-time güncelleme ile değişiklikleri anında yansıtma sistemi ekledim.

Ürün Silme:Ürün silme fonksiyonunu geliştirdim. Kullanıcı onayı alarak güvenli silme işlemi implement ettim. Silme işlemi sonrası liste güncelleme sistemi ekledim.

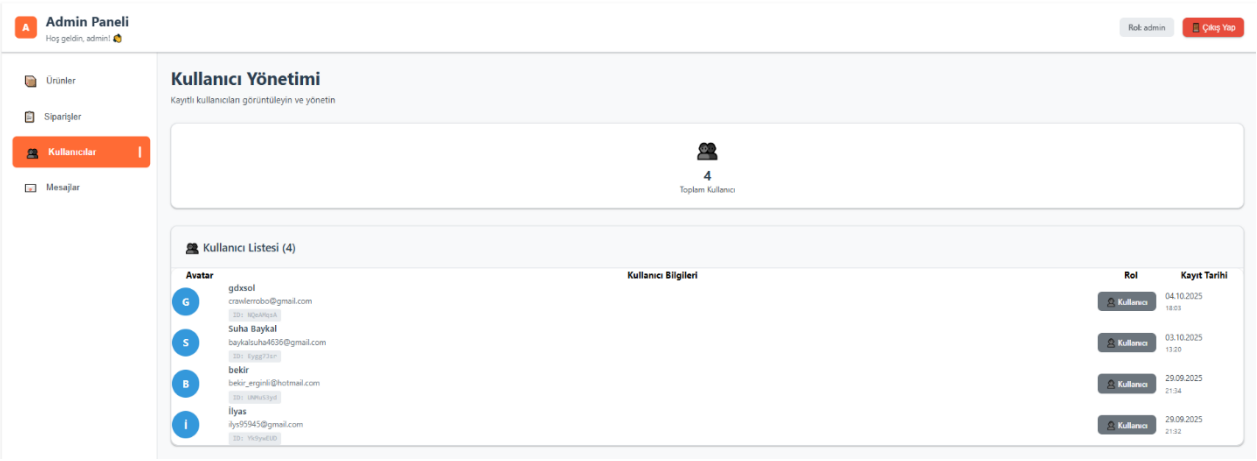
Toplu İşlemler:Çoklu ürün seçimi ve toplu işlemler sistemi kurdum. Toplu silme, toplu kategori değiştirme ve toplu stok güncelleme özellikleri ekledim.

Kullanıcı listesi sayfasını geliştirdim ve kullanıcı detay sayfasını oluşturdum. Kullanıcı rol yönetimini implement ettim ve kullanıcı arama fonksiyonunu geliştirdim. Kullanıcı istatistiklerini gösterdim.

Kullanıcı Listesi Sayfası:admin/src/pages/Users.tsx dosyasını oluşturdum ve kullanıcı listesi sayfasını implement ettim. Tüm kullanıcıları tablo formatında gösterme sistemi kurdum. Kullanıcı adı, email, rol ve kayıt tarihi bilgilerini görüntüleme sistemi implement ettim.

Kullanıcı Detay Sayfası: Kullanıcı detay sayfasını oluşturdum ve kullanıcı bilgilerini detaylı şekilde gösterme sistemi kurdum. Kullanıcı profil bilgileri, sipariş geçmişi ve aktivite logları görüntüleme sistemi implement ettim.

Rol Yönetimi: Kullanıcı rollerini yönetme sistemi kurdum. 'user', 'admin', 'moderator' rolleri arasında geçiş yapma sistemi implement ettim. Rol değişikliklerini real-time olarak takip etme mekanizması ekledim.

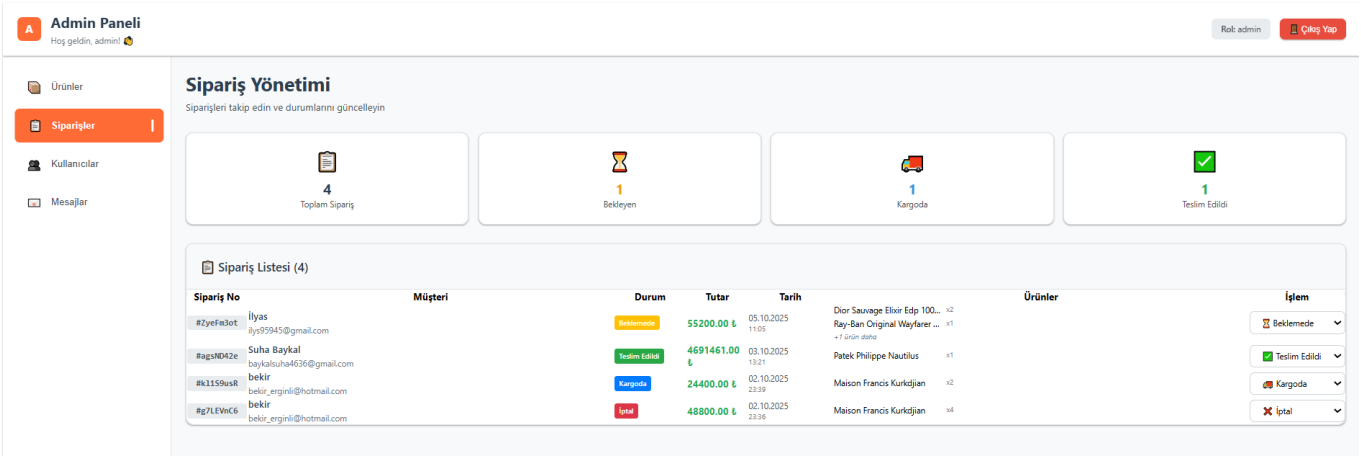


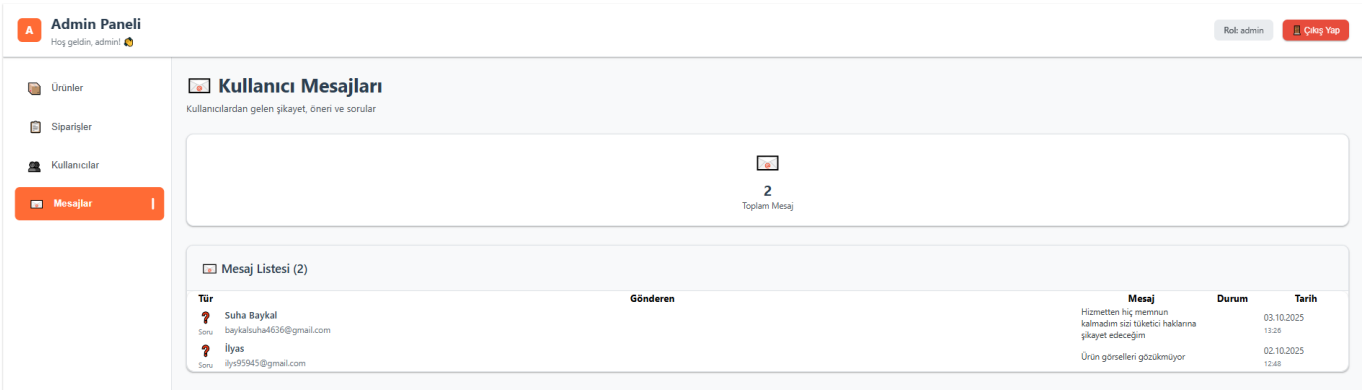
app/orders/index.tsx dosyasını geliştirdim ve sipariş listesi sayfasını implement ettim. Kullanıcının siparişlerini tarih sırasına göre listeleme sistemi kurdum. Sipariş numarası, tarih, toplam fiyat ve durum bilgilerini görüntüleme sistemi implement ettim.

Sipariş Detay Sayfası:app/order-detail/[id].tsx dosyasını geliştirdim ve dinamik routing ile sipariş detay sayfasını implement ettim. Sipariş ürünlerini listeleme, sipariş bilgilerini gösterme ve sipariş durumu takibi sistemi kurdum.

İletişim Mesajları:app/settings/contact.tsx dosyasını geliştirdim ve iletişim formu implement ettim. Kullanıcıların mesaj gönderebilmesi için form alanları ekledim. Mesaj kategorileri ve öncelik seviyeleri sistemi kurdum.

Admin Mesaj Yönetimi:admin/src/pages/ContactMessages.tsx dosyasını oluşturdum ve admin mesaj yönetimi sayfasını implement ettim. Gelen mesajları listeleme, mesaj detaylarını görüntüleme ve mesaj durumu güncelleme sistemi kurdum.





Component re-render optimizasyonunu gerçekleştirdim ve lazy loading implementasyonunu yaptım. Image optimization ve bundle size optimization işlemlerini gerçekleştirdim. Memory leak'leri tespit edip düzelttim.

Component Re-render Optimizasyonu: React.memo kullanarak gereksiz re-render'ları önledim. useCallback ve useMemo hook'ları ile performance optimizasyonu sağladım. Component dependency'lerini optimize ettim.

Lazy Loading Implementasyonu:Ağır component'ler için lazy loading sistemi kurdum. React.lazy kullanarak code splitting implement ettim. Route-based lazy loading ile bundle size'ı optimize ettim.

Image Optimization:Ürün görsellerini optimize ettim. Lazy loading, progressive loading ve image compression teknikleri uyguladım. Fallback placeholder sistemi ile kullanıcı deneyimini iyileştirdim.

Bundle Size Optimization: Webpack ve Metro bundler konfigürasyonlarını optimize ettim. Dead code elimination ve tree shaking teknikleri uyguladım. Bundle analyzer ile gereksiz kod'ları tespit ettim.

Memory Leak Önleme:useEffect cleanup fonksiyonlarını ekledim. Event listener'ları ve timer'ları temizleme sistemi kurdum. Component unmount işlemlerini optimize ettim.

Unit test'leri yazdım ve integration test'leri implement ettim. Hata ayıklama ve log yönetimi sistemini geliştirdim ve error boundary'leri ekledim. Code quality kontrolü yaptım.

Unit Test'ler: Jest ve React Testing Library kullanarak unit test'leri yazdım. Component'lerin doğru render edilmesi, user interaction'ları ve state management test'lerini implement ettim.

Integration Test'ler:API entegrasyonu, navigation flow ve user journey test'lerini yazdım. End-to-end senaryoları test ettim.

Error Boundary: React Error Boundary component'i oluşturdum. Hata yakalama ve kullanıcıya uygun mesaj gösterme sistemi kurdum. Hata loglama ve reporting sistemi implement ettim.

Log Yönetimi: Console.log'ları organize ettim. Debug, info, warning ve error seviyelerinde loglama sistemi kurdum. Production'da log'ları minimize ettim.

Code Quality:ESLint konfigürasyonunu optimize ettim. Code review süreçlerini implement ettim. Best practices ve coding standards uyguladım.

APK Oluşturma Süreci:EAS Build kullanarak Android APK oluşturma sürecini gerçekleştirdim. Production build konfigürasyonunu yaptım ve APK dosyasını oluşturdum. Build sürecinde karşılaşılan sorunları çözdüm.

EAS Build Konfigürasyonu:eas.json dosyasında production build ayarlarını yapılandırdım. Android APK build type'ını belirledim ve build optimizasyonlarını uyguladım. Signing key'leri yapılandırdım.

Build Süreci:eas build --platform android --profile production komutu ile APK oluşturma sürecini başlattım. Build log'larını takip ettim ve hata durumlarında gerekli düzeltmeleri yaptım.

APK Test Süreci: Oluşturulan APK'yı farklı Android cihazlarda test ettim. Uygulama fonksiyonlarının doğru çalıştığını doğruladım. Performance ve memory kullanımını kontrol ettim.

Terim	Türkçe Açıklaması
Loading State	Veri yüklenirken kullanıcıya gösterilen “bekleme” durumu. Genellikle butonun devre dışı kalması veya dönen animasyonla belirtilir.
Loading Indicator	Yükleme sırasında kullanıcıya sürecin devam ettiğini gösteren görsel simge (örneğin dönen spinner).
Progress Feedback	Kullanıcıya bir işlemin ilerlemesini gösteren görsel veya yazılı bildirim.
Disable Button	İşlem sürerken kullanıcı hatalarını önlemek için butonun geçici olarak devre dışı bırakılması.
Error Handling	Hata yakalama ve uygun kullanıcı mesajı gösterme süreci.
Network Error	İnternet bağlantısı veya API isteği sırasında oluşan hata.
Validation	Kullanıcının girdiği verilerin doğruluğunu kontrol etme süreci (örneğin e-posta formatı kontrolü).
Form Validation	Form alanlarının doğru doldurulup doldurulmadığını kontrol eden sistem.
Success Feedback	Başarılı işlem sonrası kullanıcıya verilen olumlu geri bildirim mesajı.
router.replace()	Sayfa yönlendirmesi yaparken önceki sayfaya geri dönülmesini engelleyen fonksiyon.
Navigation Flow	Kullanıcının uygulama içinde sayfalar arasında izlediği akış.
Dynamic Routing	Sayfa adreslerinin dinamik parametrelere göre oluşturulması (örnek: /order-detail/[id]).
API (Application Programming Interface)	Uygulama ile veritabanı veya diğer servisler arasında veri alışverişi sağlayan arabirim.
getUserOrders / getOrderById	Firestore’dan kullanıcıya ait siparişleri veya belirli sipariş detayını getiren fonksiyonlar.
Firestore	Google Firebase’in bulut tabanlı NoSQL veritabanı.
Real-time Update / Listener	Verideki değişiklikleri anlık olarak algılayıp ekrana yansıtan sistem.
Status Indicator	Sipariş durumlarını (örneğin 'pending', 'shipped') görsel olarak gösteren işaret veya renk kodu.
Dropdown Menu	Kullanıcının seçim yapabileceği açılır menü.
Retry Mechanism	Hata durumunda işlemi otomatik olarak yeniden deneme sistemi.

Pagination	Büyük veri listelerini sayfa sayfa gösterme yöntemi.
Memory Leak	Bellekte kullanılmayan verilerin temizlenmemesi sonucu performans kaybı oluşması.
Cleanup Function	React'ta bileşen kapandığında bellek temizliği yapan fonksiyon.
User Profile Management	Kullanıcı bilgilerini yönetme, düzenleme ve güncelleme sistemi.
Nested Routing	İç içe geçmiş sayfa yönlendirme yapısı.
Controlled Component Pattern	Form bileşenlerinin değerlerini React state ile kontrol etme yöntemi.
Push Notification	Uygulama dışındayken bile kullanıcılara gönderilen bildirim mesajları.
Toggle	Açma/kapama şeklinde çalışan anahtar bileşen (örneğin bildirim açık/kapalı).
FAQ (Frequently Asked Questions)	Kullanıcılara yardımcı olmak için sık sorulan sorular bölümü.
Contact Form	Kullanıcının mesaj veya geri bildirim gönderebildiği form.
Role-based Access Control (RBAC)	Kullanıcı rollerine göre (admin, user vb.) erişim yetkisi belirleme sistemi.
Admin Panel	Yöneticilerin ürün, sipariş ve kullanıcıları yönetebildiği özel yönetim arayüzü.
Firebase Admin SDK	Firebase üzerinde yönetici işlemleri yapmak için kullanılan araç seti.
Protected Route	Giriş yapılmadan erişilemeyen, korumalı sayfa yapısı.
Responsive Design	Ekran boyutuna göre otomatik uyum sağlayan tasarım.
Sidebar	Ekranın yan kısmında bulunan gezinme menüsü.
Widget	Dashboard üzerinde istatistik veya bilgi gösteren küçük bileşen.
Search & Filter	Listelenen verileri belirli kritere göre sıralama veya arama sistemi.
Form Input	Kullanıcıdan bilgi almak için kullanılan alanlar (örneğin ad, fiyat, açıklama).
Modal / Dialog	Onay veya ek işlem penceresi.
React.memo	Gereksiz yeniden render işlemlerini önleyen performans optimizasyonu.
useCallback / useMemo	Fonksiyonları ve hesaplamaları bellekte tutarak gereksiz işlemleri önleyen hook'lar.
Lazy Loading	Ağır bileşenlerin yalnızca gerektiğinde yüklenmesi.

React.lazy	Kod parçalarının sayfa bazlı olarak geç yüklenmesini sağlayan fonksiyon.
Code Splitting	Uygulama kodunu küçük parçalara ayırarak yükleme hızını artırma yöntemi.
Webpack / Metro Bundler	React projelerini derleyip tek dosya haline getiren sistemler.
Unit Test	Kodun küçük parçalarının (fonksiyon, bileşen) doğru çalıştığını test etme yöntemi.
Integration Test	Farklı sistem bileşenlerinin birlikte doğru çalıştığını test etme yöntemi.
EAS Build (Expo Application Services)	Expo projeleri için APK veya IPA oluşturmayı sağlayan bulut tabanlı sistem.