# **Proje Konusunun Önemi ve Özgün Değeri**

Hayvancılık sektörü, tarım ve gıda sanayisinin temel taşlarından biri olup hem ülke ekonomisine katkı sağlamakta hem de kırsal kesimde milyonlarca insanın geçim kaynağı olmaktadır. Ancak, sektörün karşılaştığı en büyük sorunlardan biri piyasa verilerinin şeffaf olmaması, ticaretin büyük ölçüde yerel pazarlara bağımlı olması, bilgi akışının sınırlı kalması ve tüm sistemin geleneksel yöntemler üzerinden işlemesidir. Geleneksel hayvan ticaretinde fiyat dalgalanmalarının takip edilmesi, güvenilir alıcı ve satıcı bulunması, lojistik süreçlerin yönetilmesi ve hastalık risklerinin izlenmesi gibi konular büyük ölçüde bireysel çabalara ve sınırlı ağlara dayanmaktadır. Bu durum, sektörde verimsizliği artırmakta, üreticilerin zarar görmesine neden olmakta ve ticaretin güvenilirliğini azaltmaktadır.

Son yıllarda farklı sektörlerde dijitalleşme süreçleri hız kazanırken, hayvancılık sektöründe bu dönüşüm henüz tam anlamıyla sağlanamamıştır. Ayrıca, hayvan fiyatlarındaki dalgalanmaların anlık olarak takip edilememesi, hastalık duyurularının üreticilere geç ulaşması, lojistik süreçlerin düzensiz olması ve satıcıların güvenilirlik sorunu yaşaması gibi faktörler, sektörde verimsizliğe yol açmaktadır. Örneğin, et ve süt fiyatlarındaki değişimler çiftçiler için hayati öneme sahipken, bu fiyatlara dair sağlıklı bir veri kaynağına erişim oldukça kısıtlıdır. Benzer şekilde, bölgesel hayvan hastalıklarının yayılmasıyla ilgili duyuruların merkezi bir sistem üzerinden yapılmaması, üreticilerin hayvan sağlığını korumasını zorlaştırmakta ve salgın hastalıkların yayılmasına neden olmaktadır.

Bu proje, hayvan ticaretinin dijitalleşmesini sağlayarak hem üreticilerin hem de alıcıların daha güvenilir, şeffaf ve verimli bir ticaret ortamına sahip olmasını hedeflemektedir. Mevcut platformlardan farklı olarak, gerçek zamanlı fiyat verileri sunan, kullanıcı güvenilirliğini artıran ve ticaret sürecini kolaylaştıran bir sistem oluşturulacaktır. Ayrıca, bölgesel hastalık verilerinin harita tabanlı gösterilmesi, veteriner hizmetlerine erişimin kolaylaştırılması ve nakliye süreçlerinin optimize edilmesi, projeyi benzerlerinden ayıran önemli unsurlar olacaktır.

Bu proje sayesinde, üreticiler piyasa koşullarını daha iyi analiz edebilecek, hayvanlarını daha geniş bir kitleye ulaştırabilecek, lojistik süreçleri daha etkin yönetebilecek ve hastalık duyurularını daha hızlı alarak gerekli önlemleri alabilecektir. Dijitalleşmeyle birlikte, hayvancılık sektöründe daha güvenilir, hızlı ve şeffaf bir ekosistem oluşturulması hedeflenmektedir.

# **Projenin Amacı ve Hedefi**

Bu projenin temel amacı, hayvan ticaretini dijitalleştirerek üreticiler, tüccarlar ve alıcılar arasındaki iletişimi kolaylaştırmak, sağlıklı ve güvenilir bir pazar ortamı oluşturmaktır. Hayvan alım-satım süreçlerini şeffaf hale getirmek, fiyat dalgalanmalarını takip edilebilir kılmak ve sektördeki bilgi eksikliğini gidermek, projenin temel odak noktalarıdır. Ayrıca, bölgesel hastalık duyuruları, veteriner hizmetleri ve lojistik süreçlerin entegrasyonu gibi destekleyici modüller ile sektördeki önemli sorunlara çözüm sunmak hedeflenmektedir.

Bu proje ile;

1. Hayvan alım-satım işlemlerini dijital platform üzerinden güvenli ve hızlı bir şekilde gerçekleştirebilecek,
2. Güncel problemler hakkında bilgi alış-verişinde bulunabilecek,
3. Et ve süt borsası fiyatlarını anlık olarak takip edebilecek,
4. Bölgesel hayvan hastalıkları hakkında güncel bilgilere harita üzerinden ulaşabilecek,
5. Veteriner hizmetleri ile hızlı bir şekilde iletişime geçebilecek,
6. Nakliye sürecini platform üzerinden organize edebilecek ve uygun taşıyıcıları bulabileceklerdir.

**Projenin Hedefleri**

* **Güvenilir Ticaret:** Kullanıcı değerlendirme ve yorum sistemi ile güvenilir satıcı ve alıcı profili oluşturmak.
* Piyasa Verilerinin Yaygınlaştırılması: Günlük et ve süt borsası fiyatlarını platformda otomatik olarak güncellemek ve çiftçilerin piyasa koşullarına uygun satış yapmasına yardımcı olmak.

Hastalık Takibi: Bölgesel hastalık duyurularını harita tabanlı göstererek, üreticilerin hızlı önlem almasını sağlamak.

Lojistik Süreçlerinin Entegrasyonu: Nakliye hizmetlerini sistemle entegre ederek, üreticilerin hızlı ve uygun maliyetli taşıma çözümlerine ulaşmasını sağlamak.

Gerçek Zamanlı İletişim: Alıcı ve satıcı arasında anlık mesajlaşma sistemi ile güvenli iletişim ortamı oluşturmak.

Fikir Alışverişi Yapabildiğiniz Kitleler: Bölge halkı veya katıldığınız gruplar ile istişare yaparak sorunlarınıza çözüm bulabilirsiniz.

Bu hedefler doğrultusunda geliştirilecek olan proje, hayvancılık sektöründe güvenilir, yenilikçi ve verimli bir dijital ekosistem oluşturmayı amaçlamaktadır.

**Proje İş- Zaman Çizelgesi**

|  |  |  |  |
| --- | --- | --- | --- |
| İP.  No | İş paketlerinin Adı ve Hedefleri | Zaman Aralığı (Haftalık) | Yapılacak İşlemler |
| 1 | Proje Gereksinimlerinin Belirlenmesi ve Kapsamın Tanımlanması | 1.hafta | Kullanıcı ihtiyaçları analiz edilerek, proje kapsamı ve hedefleri belirlenmiştir. İş-zaman çizelgesi oluşturulmuş ve teknoloji seçimleri yapılmıştır. |
| 2 | Kullanıcı Giriş/Kayıt Sistemi Geliştirme | 2. ve 3. hafta | Kullanıcıların güvenli bir şekilde hesap oluşturabilmesi, giriş yapabilmesi ve şifre sıfırlayabilmesi sağlanmıştır. Veri tabanına kullanıcı bilgileri güvenli şekilde kaydedilmiş ve doğrulanmıştır. |
| 3 | Al-Sat Platformunun Temel Yapısının Geliştirilmesi | 4. ve 5. hafta | Kullanıcıların ilan ekleyebileceği, fotoğraf yükleyebileceği, fiyat belirleyebileceği bir modül tamamlanmıştır. İlanların kategori bazlı filtrelenmesi ve listelenmesi işlevleri başarıyla geliştirilmiştir. |
| 4 | Al-Sat Sisteminde Kategori ve Filtreleme Özelliklerinin Eklenmesi | 6. hafta | İlanların hayvan türü, fiyat aralığı ve konum gibi kategorilerle filtrelenebilmesi sağlanmıştır. Kullanıcılar istedikleri özelliklere göre ilan arayabilmektedir. |
| 5 | Genel Sohbet Sistemi Geliştirme | 7. ve 8. hafta | Tüm kullanıcıların bir sohbet odasında mesaj gönderebilmesi ve geçmiş mesajlara erişebilmesi sağlanmıştır. Gerçek zamanlı mesajlaşma Firebase Firestore veya WebSockets üzerinden gerçekleştirilmiştir. |
| 6 | Özel Sohbet Modülü (Alıcı-Satıcı İletişimi) | 9. ve 10. hafta | Alıcı ve satıcıların birebir sohbet edebileceği bir sistem geliştirilmiştir. Sohbet geçmişi güvenli şekilde saklanmış, kullanıcıların son görülme durumları gibi özellikler eklenmiştir. |
| 7 | Et ve Süt Borsası Takip Modülünün Geliştirilmesi | 11. hafta | Et ve süt fiyatlarının günlük olarak web scraping veya manuel girişle çekilip kullanıcıya gösterilmesi sağlanmıştır. Fiyat değişimleri grafiklerle görselleştirilmiştir. |
| 8 | İl ve İlçe Bazlı Hayvan Piyasası Takibi, Hastalık Haritası ve Duyurular | 12. hafta | Bölgesel hayvan fiyatlarının gösterilmesi, farklı illerdeki fiyatların karşılaştırılması ve kullanıcıların bu bilgilere kolayca erişmesi sağlanmıştır. Google Maps API ile harita üzerinde bölgesel hayvan hastalıklarının işaretlenmesi sağlanmıştır. Kullanıcılar hastalık bildiriminde bulunabilmekte ve bu bilgiler admin onayı sonrası haritaya işlenmektedir. |
| 9 | Bildirim Sistemi Geliştirme ve Test aşamaları | 13. hafta | |  | | --- | |  |  |  | | --- | | Kullanıcılara yeni ilan, mesaj ve hastalık duyuruları gibi konularda push bildirim gönderimi sağlanmıştır. Favori ilanlarda fiyat değişikliği olduğunda bildirim gönderilmektedir. |   Uygulamanın tüm modülleri test edilmiştir. Performans sorunları giderilmiş, kullanıcı deneyimini artırmaya yönelik optimizasyonlar yapılmıştır. |
| 10 | Proje Raporunun Hazırlanması | 14. hafta | Yapılan proje uygun şekilde raporlanmıştır. |

**1. İş Paketi Sonuçları (Hafta 1): Teknoloji Araştırması**

Gerekli teknoloji araştırmaları tamamlanmıştır. Yapılan analizler sonucunda:

* Veri tabanı olarak Google Firebase kullanılmasına,
* Programlama dili olarak Dart tercih edilmesine karar verilmiştir.
* Kişisel veriler, Firebase Database üzerinde saklanmaktadır.
* Kimlik doğrulama işlemleri Firebase Authentication ile sağlanmaktadır.
* Görsel veriler, Firebase Storage üzerinde tutulmaktadır.

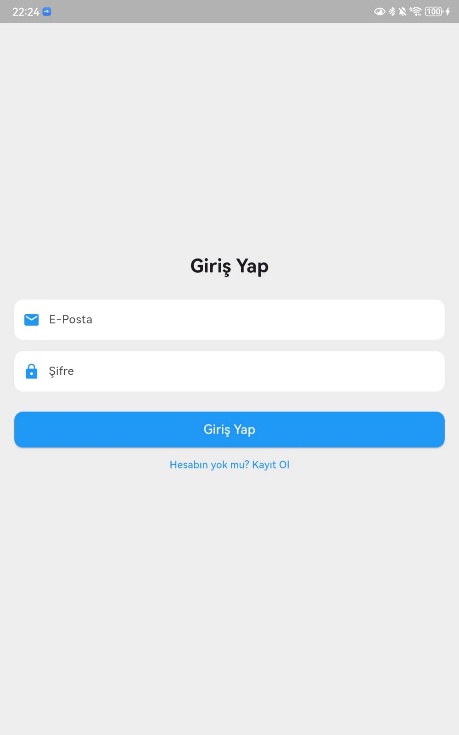
Şu ana kadar yapılan araştırmalar bu doğrultuda ilerlemektedir.

**2. İş Paketi Sonuçları (Hafta 2-3): Kullanıcı Giriş & Kayıt İşlemleri**

Giriş ve kayıt işlemleri için ekran tasarımları tamamlanmıştır.

* Kullanıcılar kayıt işlemi sırasında bilgilerini girdikten sonra e-posta doğrulaması yapmak zorundadır.
* E-posta doğrulama işlemi için Firebase Authentication kullanılmaktadır.
* Firebase'in e-posta doğrulama hizmeti ücretli olup, bu özelliği aktif edebilmek için Google Firebase'in Blaze Planı'na geçiş yapılmıştır.

Projeye dair anlık görseller ilgili dokümantasyon içerisinde paylaşılmıştır.



Şekil 1. Giriş ekranı

Giriş ekranında bulunan widgetlar, Text ile ana başlığın gösterilmesini sağlamaktadır. E-posta ve şifre girişleri için TextField widgetları kullanılmıştır ve bu widgetların içinde Flutter’ın sunduğu Icon bileşenleri yer almaktadır. Kullanıcı, giriş yapabilmek için gerekli bilgileri girdikten sonra ElevatedButton ile giriş işlemi başlatılır. Bu butonun altında bulunan "Hesabın yok mu? Kayıt Ol" yazısı TextButton kullanılarak oluşturulmuştur ve bu butona tıklanması durumunda kullanıcı kayıt olma ekranına yönlendirilir. Kullanıcının girdiği bilgiler Firebase Authentication'a gönderilir ve burada e-posta ile şifrenin Firestore’da var olup olmadığı kontrol edilir. Eğer e-posta doğrulanmamışsa kullanıcıya doğrulama yapması gerektiğine dair bir uyarı gösterilir. Bknz. “Şekil 4”. Kullanıcı bilgileri doğruysa ve e-posta doğrulanmışsa ana ekrana yönlendirilir.

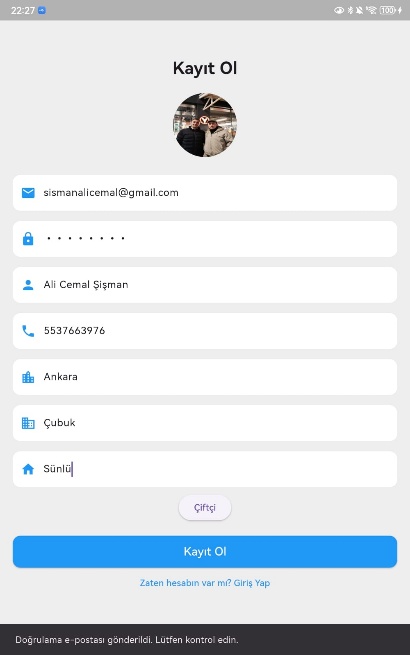
metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 2. Kayıt ekranı

Kayıt ol ekranında Kullanıcı, e-posta, şifre, ad soyad, telefon, şehir, ilçe ve mahalle gibi bilgileri TextField bileşenleri aracılığıyla girebilir. Kullanıcıya ayrıca meslek seçme opsiyonu sunulmuş olup, bu işlem AlertDialog üzerinden gerçekleştirilir. Eğer kullanıcı "Diğer" seçeneğini tercih ederse manuel olarak meslek bilgisini girebileceği bir alan açılır. Kullanıcıya profil fotoğrafı ekleme imkânı sunulmuş olup, GestureDetector kullanılarak galeri üzerinden resim seçmesine olanak tanınmıştır. Seçilen resim Firebase Storage’a yüklenmeden önce Image paketini kullanarak sıkıştırılır ve optimize edilir.

Kullanıcı kayıt işlemini tamamladığında, Firebase Authentication kullanılarak yeni bir hesap oluşturulur. Kullanıcı bilgileri Firestore’a kaydedilir ve profil resmi Firebase Storage’a yüklenir. Ayrıca, kullanıcının e-posta adresine doğrulama bağlantısı gönderilir (bknz şekil 3 ve şekil 5)ve doğrulama işlemi tamamlanmadan giriş yapmasına izin verilmez(bknz şekil 4). Kayıt işlemi başarılı olduğunda, kullanıcı giriş ekranına yönlendirilir ve sisteme giriş yapabilmesi için e-posta doğrulamasını tamamlaması beklenir.



Şekil 3. Bilgilerin girildiği

Kayıt ekranı

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 4. Kayıt yapıldığı halde e-posta

Onayı yapılmaz ise kullanıcıya bildirim

Veriliyor.

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 5. Kullanıcıya kayıt için gelen e-posta

**3. İş Paketi Sonuçları (Hafta 4-5): Al-Sat Platformunun Temel Yapısının Geliştirilmesi**

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 6. Ana Sayfa Ekranı

Anasayfa ekranında, kullanıcı giriş yaptıktan sonra Firebase Authentication üzerinden giriş yapan kullanıcının adı Firestore’dan çekilerek ekranda "Hoş Geldiniz, [Kullanıcı Adı]" şeklinde bir mesaj gösterilir. Kullanıcı dostu bir arayüz sağlamak için AppBar widgetı kullanılmıştır. Sağ üst köşeye konumlandırılmış bir IconButton bileşeni ile kullanıcı, daha önce oluşturduğu ilanları görüntüleyebileceği MyListingsScreen adlı ekrana yönlendirilebilir.

Ana ekranda, üç farklı kategori sunulmuştur: "Nakliye", "Hayvan Al-Sat" ve "Tarım Ürünleri". Bu kategoriler, ListView içerisinde ListTile widgetları kullanılarak oluşturulmuştur. Her bir kategoriye, ilgili konuyu temsil eden bir Icon atanmış ve sağ tarafına bir arrow\_forward\_ios ikonu eklenerek navigasyon akışı sağlanmıştır. Kullanıcı, herhangi bir kategoriye tıkladığında ilgili ilan listesini görüntüleyebileceği ListingScreen adlı sayfaya yönlendirilir.

Ekranın sağ alt köşesinde, FloatingActionButton bileşeni kullanılarak bir ekleme butonu oluşturulmuştur. Kullanıcı bu butona tıkladığında, showModalBottomSheet ile açılan bir alt menü sayesinde ilan ekleme türünü seçmesi istenir. Bu menüde "Nakliye İlanı", "Hayvan Al-Sat İlanı" ve "Tarım Ürünleri İlanı" olmak üzere üç seçenek bulunmaktadır. Her bir seçenek ListTile bileşeniyle oluşturulmuş ve yanlarına ilgili kategoriye uygun ikonlar eklenmiştir.

Kullanıcı, ilan türünü seçtikten sonra, seçtiği türe göre ilgili ilan ekleme ekranına yönlendirilir. Örneğin, "Hayvan Al-Sat İlanı" seçildiğinde, kullanıcı AddListingScreen adlı ekrana yönlendirilerek ilan bilgilerini girebileceği bir form ile karşılaşır (bknz şekil 7). Form tamamlandığında, ilan Firebase Firestore veritabanına kaydedilir ve kullanıcı ilanlarını yönetebileceği bir ekrana yönlendirilir. Projede şu an sadece “Hayvan al-sat” kısmı aktif olarak çalışmaktadır.

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 7. FloatingActionButton ile hangi

Kategoride ilan vereceğimizi seçiyoruz.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 8. İlan ekleme ekranı

İlan ekleme ekranında Kullanıcılar ilan başlığını, açıklamasını ve fiyat bilgisini girebilmek için TextField widgetlarını kullanır. Bu alanlar, kullanıcının metin girişini kolaylaştırmak amacıyla uygun input türleriyle yapılandırılmıştır. Örneğin, fiyat alanı yalnızca sayısal giriş kabul edecek şekilde tasarlanmıştır.

Kategori seçimi için DropdownButtonFormField bileşeni kullanılmıştır. Bu bileşen sayesinde kullanıcı, mevcut kategoriler arasından seçim yapabilir. Seçilen kategori, ilgili ilan verileriyle birlikte Firestore'a kaydedilir.

Fotoğraf ekleme işlemi için, kullanıcıların en fazla 3 adet fotoğraf seçmesine izin verilmiştir. Kullanıcı, ImagePicker kütüphanesi aracılığıyla galeriden fotoğraf seçebilir. Seçilen fotoğraflar bir liste halinde GridView bileşeni kullanılarak ekranda gösterilir. Kullanıcı, üzerine tıklayarak fotoğrafları ön izleyebilir ve kaldırmak istediği görselleri silebilir.

İlan ekleme işlemi tamamlandığında, Firebase Storage kullanılarak fotoğraflar yüklenir ve Firestore veritabanında ilan bilgileri saklanır. Kullanıcı, ilan ekleme sürecinde eksik bilgi girdiğinde veya gerekli alanları doldurmadığında, ScaffoldMessenger ile bir uyarı mesajı görüntülenir. İşlem başarıyla tamamlandığında ise kullanıcıya "İlan başarıyla eklendi!" mesajı gösterilir ve kullanıcı bir önceki ekrana yönlendirilir.

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 9. İlanlarım Sayfası

İlanlarım ekranında, kullanıcıların daha önce oluşturduğu ilanları listeleyerek yönetebileceği bir arayüz bulunmaktadır. Bu ekran, Firebase Authentication ile oturum açmış kullanıcının kimliğini belirleyerek Firestore veritabanında kayıtlı olan ilanlarını filtreler. Kullanıcının eklediği ilanlar, Firestore’dan gerçek zamanlı olarak StreamBuilder ile çekilir ve ListView.builder kullanılarak ekranda listelenir.

Her ilan, Card ve ListTile widgetları kullanılarak tasarlanmıştır. İlanın başlığı, fiyat bilgisi ve ilan için eklenen ilk fotoğraf bir arada görüntülenir. Kullanıcıya, ilanı düzenleme veya silme işlemlerini gerçekleştirebilmesi için iki farklı buton sunulmuştur.

İlan düzenleme işlemi için, kullanıcının mevcut ilan bilgilerini düzenleyebileceği ayrı bir ekrana yönlendirme yapılır. Bu işlem, ilgili ilana ait Firestore verilerinin düzenlenmesini sağlar. Kullanıcı ilan başlığını, açıklamasını veya fiyat bilgisini değiştirdiğinde, güncellenmiş veriler Firestore üzerinde anında güncellenir.

İlan silme işlemi için, kullanıcıya bir onay penceresi (AlertDialog) gösterilir. Kullanıcı silme işlemini onayladığında, ilgili ilan Firestore veritabanından kaldırılır. İşlem tamamlandığında, kullanıcıya "İlan başarıyla silindi." şeklinde bir geri bildirim mesajı gösterilir.

Kullanıcı bir ilan üzerine tıkladığında, ilan detaylarını görüntüleyebileceği ayrı bir ekrana yönlendirilir. Burada, ilan ile ilgili daha fazla bilgiye ulaşabilir ve ilan fotoğraflarını tam ekran olarak görüntüleyebilir. Kullanıcı deneyimini artırmak için, fotoğraflar arasında geçiş yapabileceği bir PageView bileşeni eklenmiştir.

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir. Şekil 10. Hayvan Al-Sat İlanları Ekranı

Hayvan ilanları ekranında, kullanıcılar mevcut ilanları görüntüleyebilir ve detaylarına ulaşabilir. Firestore veritabanında kayıtlı olan ilanlar, StreamBuilder kullanılarak gerçek zamanlı olarak çekilir ve ListView.builder ile listelenir. Her ilan, Card ve ListTile widgetları kullanılarak tasarlanmış olup, ilan başlığı, fiyat bilgisi, satıcı adı ve eklenmiş ilk fotoğraf bir arada görüntülenir.

İlan listesi, belirli bir kategoriye ait ilanları gösterecek şekilde filtrelenmiştir. Kullanıcı bir ilana tıkladığında, ListingDetailScreen adlı ekrana yönlendirilir (bknz Şekil 11) ve burada ilan detayları daha ayrıntılı şekilde gösterilir.

İlan detay ekranında, ilan başlığı, fiyat bilgisi, açıklama, satıcı adı, telefon numarası ve konum bilgileri kullanıcıya sunulur. Kullanıcı deneyimini iyileştirmek için, ilan fotoğraflarını tam ekran görüntüleme özelliği eklenmiştir. Sayfada bulunan fotoğraf, PageView ile kaydırılabilir hale getirilmiş ve kullanıcının görseller arasında rahatça geçiş yapmasına olanak sağlanmıştır.

Satıcı bilgileri, Firestore veritabanındaki kullanıcı bilgilerinden dinamik olarak çekilerek gösterilir. Kullanıcı, ilan sahibine ait telefon numarası ve konum bilgilerine ulaşarak iletişime geçebilir. Kullanıcı dostu bir arayüz sağlamak amacıyla, tüm bilgiler sade ve anlaşılır bir biçimde sunulmuştur.

metin, sığır, ekran görüntüsü, canlı hayvan içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 11. İlan Detayları Ekranı

metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 12. Uygulama Açıldığında Gelen

Animasyon Ekranı (main.dart) Splash

Screen

Splash ekranı, uygulama açıldığında kullanıcıya markayı tanıtmak ve yükleme sürecini yönetmek amacıyla tasarlanmıştır. Ekranın arka planı koyu mavi bir renk ile belirginleştirilmiş ve marka adı "Çiftlik Plus" vurgulanmıştır. Bu başlık, Text widgetı kullanılarak oluşturulmuş ve animasyonlu bir giriş efekti ile ekranda belirir.

Alt kısımda, uygulamanın sloganı olan "Hayvancılık ve Ticarette Dijital Dönüşüm!" metni, italic font stili ile kullanıcıya sunulmuştur. Animasyonlu bir geçiş efekti eklenerek kullanıcı deneyimi daha dinamik hale getirilmiştir.

Ekranın merkezinde bir yükleme göstergesi (CircularProgressIndicator) bulunmaktadır. Bu bileşen, uygulamanın yüklenme sürecini temsil eder ve belirli bir süre sonunda kullanıcıyı otomatik olarak giriş ekranına yönlendirir. Navigasyon işlemi, Future.delayed fonksiyonu ile zamanlanmış (3 saniye) ve Firebase Authentication kullanılarak kullanıcının oturum durumu kontrol edilmiştir.

Splash ekranı tasarımında, SingleTickerProviderStateMixin kullanılarak animasyonlar senkronize edilmiştir. Başlık ve slogan metinleri için SlideTransition ve FadeTransition widgetları kullanılmıştır. Bu yapı, giriş animasyonlarını akıcı hale getirerek uygulamanın profesyonel bir görünüm kazanmasını sağlamıştır.

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 13. Nakliye İlan Ekleme

Ekranı

Nakliye ilanı ekleme ekranında kullanıcıların girdiği bilgiler Firestore veritabanına kaydedilerek, nakliye ilanlarının oluşturulmasını sağladım.

Bu ekranda DropdownButtonFormField kullanarak kullanıcıların araç tipini seçmesini sağladım. Kullanıcı, önceden tanımlı seçenekler arasından bir araç tipi seçebiliyor. TextField bileşenleriyle kapasite, hizmet bölgesi, fiyatlandırma ve açıklama gibi bilgilerin girişini sağladım. ElevatedButton kullanarak ilanı yayınlama işlemini başlattım. Kullanıcı butona tıkladığında, girilen veriler Firestore’a ekleniyor.

Ayrıca, eksik alan kontrolü yaparak, eğer bir alan boş bırakılırsa kullanıcıya uyarı mesajı gösteriyorum. Böylece hatalı ilan girişlerini engelliyorum. Bu ekran, kullanıcıların nakliye ilanlarını hızlı ve kolay bir şekilde oluşturmasını sağlayacak şekilde tasarlandı.

metin, ekran görüntüsü, ekran, görüntüleme, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 14. Nakliye İlanlarını Görme

Ekranı

Nakliye hizmetleri ekranında, kullanıcıların mevcut nakliye ilanlarını görüntüleyebilmesi ve güzergâh bazlı arama yapabilmesi için nereden- nereye arama çubuklarını ekledim. Kullanıcılar konum belirterek ilgili ilanları listeleyebiliyorlar.

Araç türüne göre filtreleme özelliği de eklenerek Kamyon, Tır, Kamyonet, Dorse gibi seçeneklerden birini seçerek listeyi daraltabiliyorlar. Seçilen araç tipi vehicleType alanına göre Firestore veritabanında filtreleniyor ve yalnızca ilgili araçlarla eşleşen ilanlar ekrana yansıtılıyor.

İlanlar ListView.builder kullanılarak listelendi ve her ilan Card widget içerisinde araç sahibi, araç tipi, kapasite ve hizmet verilen bölge bilgileriyle sunuldu. Kullanıcılar bir ilana tıkladığında Nakliye Detay Sayfasına yönlendiriliyor ve burada ilan sahibinin adı, araç tipi, kapasitesi, hizmet bölgesi, fiyatlandırması ve iletişim bilgileri detaylı şekilde görüntüleniyor.Filtreleme işlemi için kullanıcının girdiği Nereden ve Nereye bilgileri serviceArea alanına göre veritabanında aranıyor. Kullanıcı herhangi bir güzergâh ya da araç tipi seçmeden de tüm ilanları görüntüleyebiliyor.

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 15. Tarım Ürünleri Ekleme

Ekranı

Tarım ilanı verme ekranında, kullanıcıların tarım ürünleriyle ilgili ilan oluşturmasını sağladım. Kullanıcılar, ilan oluştururken ürün adı, miktar, fiyat ve açıklama gibi temel bilgileri girebiliyor.

İlanın görselliğini artırmak için ImagePicker kütüphanesiyle kullanıcının cihazından bir fotoğraf seçmesine olanak tanıdım. Seçilen fotoğraf Firebase Storage’a yükleniyor ve ardından ilanla ilişkilendiriliyor. TextFormField bileşenleri ile kullanıcıdan zorunlu bilgileri alıyorum ve eksik alanlar için doğrulama yaparak formun hatalı gönderilmesini engelliyorum.

İlan yayınla butonuna basıldığında, tüm bilgiler FirebaseFirestore veritabanına kaydediliyor. Aynı zamanda, kullanıcının adı, telefon numarası ve konumu otomatik olarak ilana ekleyerek alıcıların satıcıyla iletişim kurmasını kolaylaştırıyorum.

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 16. Tarım Ürünlerini

Listeleme Ekranı

Tarım ilanları listeleme ekranında, kullanıcıların tarım ürünleriyle ilgili mevcut ilanları görüntüleyebilmesi için Firestore’dan tarim\_ilanlari koleksiyonundaki verileri çekerek gerçek zamanlı bir liste oluşturdum.

Her ilan, ürün adı, miktar, fiyat ve satıcı bilgilerini içerecek şekilde tasarlandı. Kullanıcılar ilanlara tıkladığında detay sayfasına yönlendiriliyor ve burada ilan hakkında daha fazla bilgi alabiliyorlar. Detay sayfasında ürünün fotoğrafı, miktarı, fiyatı, açıklaması ve satıcının iletişim bilgileri gibi tüm önemli detaylar yer alıyor.

İlanlar ListView.builder kullanılarak listeleniyor ve her ilan Card widget içinde ürün resmi ve açıklamalarıyla birlikte sunuluyor. Eğer ilan için bir ürün fotoğrafı yoksa, varsayılan bir ikon göstererek kullanıcıya anlaşılır bir arayüz sağladım. Ayrıca, satıcının adı, telefon numarası ve konumu gibi bilgiler ilan detaylarında yer alarak kullanıcıların satıcıyla doğrudan iletişime geçmesini kolaylaştırıyor. Böylece, tarım ürünü alıcıları ve satıcıları için daha pratik ve hızlı bir alışveriş süreci sunulmuş oldu.

Tarım ürünleri ve nakliye ekranları için detay sayfaları da bulunmaktadır. Bu sayfalar, hayvan ilanlarının detay sayfasına benzer olduğu için (Şekil 11) görsellerin eklenmesine gerek duyulmamıştır. Haber listeleme için web scraping yöntemini kullanmaya çalıştım ancak başarılı olamadım. Önümüzdeki süreçte, mevcut hataları gidermeye ve uygulamanın işlevselliğini artırmaya odaklanacağım.

**4. İş Paketi Sonuçları (Hafta 6) Al-Sat Sisteminde Kategori ve Filtreleme Özelliklerinin Eklenmesi**

Bu iş paketi kapsamında:

* Hayvan ilanları bölümünde yalnızca büyükbaş ve küçükbaş hayvanlara yönelik filtreleme yapılabilmektedir; bu sistem ilerleyen aşamalarda geliştirilecektir.
* Nakliye hizmetleri bölümünde, nereden nereye bilgisi ve dört farklı araç tipine göre filtreleme imkânı sunulmaktadır.
* Tarım ilanları bölümünde ise henüz bir filtreleme özelliği bulunmamaktadır; ancak ilerleyen süreçte konuma dayalı filtreleme özelliği eklenmesi planlanmaktadır.

**5. İş Paketi Sonuçları (Hafta 7-8): Genel Sohbet Sistemi Geliştirme**

metin, elektronik donanım, ofis ekipmanı, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 17. Genel Sohbet Kanalı

**Şekil 17. Kanal Sohbeti Ekranı**

Bu ekran, kullanıcıların bir kanala katıldıktan sonra kanal içerisindeki toplulukla mesajlaşmasını sağlayan sohbet arayüzünü temsil eder. Her bir mesaj, gönderen kişinin adıyla birlikte görünmekte olup, kullanıcının kendi mesajları sağda yeşil arka planla, diğer kullanıcıların mesajları ise solda gri arka planla sunulmaktadır.

Kullanıcılar ekranın alt kısmında bulunan metin kutusuna mesajlarını yazarak "gönder" ikonuna tıklayabilir ve mesajlarını kanala iletebilir. Mesajın yanında saat bilgisi de yer alır. Sağ üstteki üç nokta menüsü üzerinden "Kanal Ayarları" ekranına yönlendirme yapılabilmektedir.

**Kullanılan Teknolojiler:**

* Firebase Firestore: Kanal mesajları channels/{channelId}/messages koleksiyonu altında saklanmaktadır.
* Firebase Auth: Gönderilen mesajlarda kullanıcının kimliği FirebaseAuth.instance.currentUser üzerinden alınmaktadır.
* StreamBuilder: Gerçek zamanlı olarak yeni gelen mesajları anında görüntülemek için kullanılmaktadır.
* DateFormat (intl paketi): Mesajların saat bilgisi HH:mm formatında gösterilmektedir.
* Material UI Bileşenleri: TextField, ListTile, Container, IconButton, PopupMenuButton gibi Flutter bileşenleri ile ekranın kullanıcı dostu ve temiz görünmesi sağlanmıştır.

**Teknik İşleyiş:**

* Kullanıcı mesaj gönderdiğinde:
* Firestore'dan kullanıcının adı çekilir.
* senderId, senderName, text, sentAt alanlarıyla birlikte Firestore'a mesaj olarak kaydedilir.
* Mesajlar zaman sırasına göre artan şekilde listelenir (orderBy('sentAt')). Bu yapının sağlanabilmesi için firestore’da değişkenin indexlenmesi gerekmektedir.
* StreamBuilder kullanımı sayesinde yeni gelen mesajlar anlık olarak listeye eklenir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 18. Kanal Oluşturma Ekranı

**Şekil 18. Yeni Kanal Oluşturma Ekranı**

Bu ekran, kullanıcıların topluluk kanalları oluşturabilmesini sağlayan bir arayüz sunar. Kullanıcılar kanal adı ve isteğe bağlı olarak açıklama bilgilerini girerek sistemde yeni bir kanal oluşturabilirler.

Buton tıklanıldığında, kullanıcıdan alınan veriler doğrulanarak Firestore üzerinde bir kanal belgesi oluşturulur. Kanal oluşturulmasının ardından kullanıcı kanalın hem yöneticisi (admins) hem de üyesi (members) olarak kaydedilir.

**Kullanılan Teknolojiler:**

* **Firebase Firestore**: Kanal verileri channels koleksiyonuna kaydedilir. Her kanal için name, description, createdBy, admins, members, createdAt alanları tutulur.
* **Firebase Auth**: Kanalı oluşturan kullanıcının kimliği FirebaseAuth.instance.currentUser ile alınarak createdBy alanına yazılır.
* **Form & TextFormField**: Flutter’ın form altyapısı ile kullanıcıdan kanal adı ve açıklaması alınır. TextFormField bileşenleri ile doğrulama (validation) işlemleri yapılır.
* **Loading Durumu**: Kanal oluşturma işlemi sırasında CircularProgressIndicator gösterilerek kullanıcıya sürecin işlediği görsel olarak bildirilir.

**Teknik İşleyiş:**

1. Kullanıcı formu doldurur ve “Kanalı Oluştur” butonuna tıklar.
2. Form doğrulaması yapılır (FormState.validate()).
3. FirebaseFirestore.instance.collection('channels').add({...}) komutu ile kanal Firestore'a yazılır.
4. createdBy, admins ve members alanlarında kullanıcı kimliği (uid) atanır.
5. İşlem tamamlandığında kullanıcı bir önceki ekrana yönlendirilir.

Bu ekran sayesinde kullanıcılar kendi topluluklarını başlatabilir ve diğer kullanıcılarla bu kanal üzerinden etkileşime geçebilir.

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 19. Katıldığım Kanallar

Ekranı

**Şekil 20. Katıldığım Kanallar Ekranı**

Bu ekran, kullanıcıların halihazırda üyesi oldukları topluluk kanallarını listeleyen bir yapıya sahiptir. Kullanıcının UID’si FirebaseAuth üzerinden alınarak, channels koleksiyonunda members dizisinde yer aldığı kanallar sorgulanır. Böylece sadece ilgili kullanıcının katıldığı kanallar dinamik olarak gösterilir.

**Kullanılan Teknolojiler ve Mantık:**

* **Firebase Authentication**: Aktif oturum açmış kullanıcının kimliği alınarak sadece onunla ilişkili kanallar filtrelenir.
* **Firestore StreamBuilder**: Kullanıcıya ait kanal verileri where('members', arrayContains: currentUserId) filtresi ile alınır. StreamBuilder ile dinamik bağlantı sağlanarak kanal listesi anlık olarak güncellenir.
* **ListView.builder**: Kanallar liste formatında sunulur. Her kanal için name, description, members.length gibi bilgiler gösterilir.
* **Card + ListTile**: Her kanal bilgisi sade bir kart içerisinde sunulmuş ve kartın sağ kısmına yerleştirilen yön simgesi ile kullanıcı kanal sohbet ekranına yönlendirilmiştir.

**Teknik Özellikler:**

* **Kanal Adı** (name): Kalın font ile başlık olarak gösterilir.
* **Üye Sayısı** (members.length): Toplam kullanıcı sayısını gösterir.
* **Açıklama** (description): Varsa kısa açıklama, yoksa alan boş bırakılır.
* **Yönlendirme**: Her kanal tıklanabilir yapıdadır. Tıklanınca ChannelChatScreen(channelId) ekranına (**Şekil 17**) geçiş yapılır.

**Kullanıcı Deneyimi:**

Kullanıcı, üyesi olduğu tüm kanalları bu ekranda tek bakışta görebilir ve istediği zaman sohbet ekranına erişebilir. Böylece topluluklar arası geçiş hızlı ve kullanıcı dostu bir biçimde sağlanır. Ayrıca Firestore’un snapshots() fonksiyonu sayesinde liste her an günceldir; yeni kanallar anında görünür, çıkılan kanallar anında kaybolur.

metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir. metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 21. Kanal Üyesinin Ayarlar Şekil 22. Yöneticinin Ayarlar

Ekranı Ekranı

Bu ekran (Şekil 21), kullanıcının katıldığı bir kanalın ayarlarını göstermektedir. Ancak bu örnekte kullanıcı yönetici değildir, bu nedenle yalnızca “Üyeleri Gör” ve “Kanaldan Çık” gibi sınırlı işlemler yapabilir.

* “Üyeleri Gör” butonuna basıldığında kanal üyeleri ekranına (Şekil 23) geçilir.
* “Kanaldan Çık” işlemi, kullanıcının hem members hem de admins dizilerinden silinmesini sağlar.

Kod içerisinde yönetici kontrolü şu şekilde sağlanır:

final bool isAdmin = admins.contains(currentUserId);

Şekil 22 ve Şekil 23’ teki 2 ekran kanal yöneticisi için geliştirilmiştir.

1. **(Ayar Ekranı)** (Şekil 22):

* Kanal adı ve açıklaması düzenlenebilir.
* Güncelleme işlemi Firestore’daki dokümanı .update() ile yapar.
* Yönetici, kanalı silebilir. Kanal silinmeden önce AlertDialog ile onay alınır.

1. **Üyeler Ekranı** (Şekil 23):

* Kanal üyeleri listelenir. Her üyeye ait profil fotoğrafı, isim ve rol bilgisi gösterilir.
* Yönetici, diğer üyeleri kanaldan çıkarabilir veya yönetici yapabilir.
* Bu işlemler, ilgili Firestore dizilerine FieldValue.arrayUnion() veya arrayRemove() ile yapılır.

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 24. Kanal Arama ve

Katılma Ekranı

**Şekil 24. Yeni Kanalları Keşfet Ekranı**

Yeni Kanalları Keşfet ekranı, uygulamadaki kullanıcıların henüz katılmadıkları kanalları listeleyerek bu kanallara katılımı teşvik etmektedir. Firestore veritabanında channels koleksiyonundan alınan tüm belgeler (kanallar), StreamBuilder kullanılarak gerçek zamanlı olarak ekrana yansıtılmaktadır.

Kullanıcının kendi UID’si, kanalın members listesinde bulunmuyorsa (Bu katıldığı kanalların gösterilmemesini sağlıyor), ilgili kanal listede gösterilir. Arama kutusu yardımıyla, kanal ismine göre case-insensitive filtreleme yapılır. Filtreleme işlemi client-side'da gerçekleştirilir.

**Teknik Özellikler:**

* **Veri Kaynağı**: Firestore – channels koleksiyonu.
* **Gerçek Zamanlı Veri**: StreamBuilder ile tüm kanallar dinlenir.
* **Filtreleme**: members içinde kullanıcının UID'si olmayan kanallar filtrelenir. Ayrıca, name alanı toLowerCase() ile karşılaştırılarak arama yapılır.
* **Arayüz Elemanları**:
  + TextField: Kanal adıyla arama yapmayı sağlar.
  + ListView.builder: Filtrelenmiş kanalları listeler.
  + ElevatedButton: Katıl butonu ile kullanıcı kanala eklenir ve sohbet ekranına yönlendirilir.

**Katılma İşlemi:**

Katıl butonuna basıldığında:

1. Firestore’da ilgili kanal belgesinde members alanına FieldValue.arrayUnion([uid]) ile kullanıcı eklenir.
2. Eğer işlem başarılı olursa, ChannelChatScreen sayfasına yönlendirme yapılır.
3. Hata durumları try-catch bloğunda loglanır.

**Kullanıcı Deneyimi:**

Bu ekran sayesinde kullanıcılar, uygulamada aktif olan diğer kanalları keşfedebilir ve ilgi alanlarına göre sohbete katılabilir. Arama çubuğunun case-insensitive çalışması, kullanıcı deneyimini kolaylaştırmakta; gereksiz filtre hatalarını azaltmaktadır.

**6. İş Paketi Sonuçları (Hafta 9-10): Özel Sohbet Modülü (Alıcı-Satıcı İletişimi)**

metin, memeli, sığır, canlı hayvan içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 25. Güncellenmiş

Hayvan İlanı Ekranı

**Şekil 25. Güncellenmiş Hayvan İlanı Detay Ekranı**

İlan Detay ekranı, kullanıcıların seçtikleri ilana ait bilgileri detaylı biçimde görebildikleri ekran olup; ilan görseli, fiyat bilgisi, açıklama ve satıcıya ait iletişim detayları burada listelenmektedir. Bu ekran, ListingDetailScreen sınıfı içerisinde yapılandırılmıştır. (Bu ekranımız mevcuttu)

Sonradan entegre edilen “Mesaj Gönder” butonu sayesinde kullanıcı, ilan sahibine direkt olarak mesaj gönderme ekranına yönlendirilir. Ancak kullanıcı kendi ilanına bakıyorsa, bu butonun yerine “Bu ilan size ait.” uyarısı gösterilerek mesajlaşma aksiyonu engellenir. Bu kontrol, Firebase Authentication üzerinden alınan aktif kullanıcı UID’si ile yapılır.

Koşullu Buton Gösterimi: Eğer kullanıcı ilanın sahibi ise ElevatedButton yerine bilgilendirme yazısı gösterilir. Farklı bir kullanıcı ise butona tıklandığında ChatScreen ekranına yönlendirme yapılır (Şekil26).

**Kodun İşleyişi:**

* ElevatedButton ile “Mesaj Gönder” butonuna tıklanınca \_openOrCreateChat() fonksiyonu çağrılır.
* Fonksiyon, önce chats koleksiyonunda mevcut bir sohbet olup olmadığını kontrol eder.
* Eğer sohbet mevcut değilse, Firestore’da yeni bir belge oluşturulur.
* Kullanıcı ChatScreen sayfasına yönlendirilir. Mesajlaşma sağlanır.

metin, ekran görüntüsü, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, ekran görüntüsü, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 26. Mesajlaşma Ekranı Şekil 27. Mesajlaşma Ekranı

**Şekil 26 ve 27. Sohbet Ekranı**

Sohbet ekranı, iki kullanıcı arasında birebir mesajlaşma yapılmasını sağlayan ekran olup ChatScreen sınıfı içinde tanımlanmıştır. Firebase Firestore’un gerçek zamanlı verisi kullanılarak messages alt koleksiyonuna eklenen her mesaj, StreamBuilder ile ekrana anlık yansıtılır.

Kullanıcının gönderdiği mesajlar sağda ve yeşil tonlarında, gelen mesajlar ise solda gri renkle gösterilir. Ekranın alt kısmında bulunan TextField aracılığıyla mesaj yazılır ve IconButton ile gönderilir.

Mesaj gönderildikten sonra yalnızca veritabanına eklenmekle kalmaz, aynı zamanda karşı kullanıcıya push notification gönderilir. Bu bildirim altyapısı, Google Firebase Cloud Messaging (FCM) ve service\_account kimlik bilgileri kullanılarak çalışır.

**Teknik Özellikler:**

* Veri Tabanı: Firestore – chats/{chatId}/messages alt koleksiyonu
* Gerçek Zamanlı Veri: StreamBuilder ile anlık dinleme
* Bildirim Sistemi: sendPushNotification() fonksiyonu ile FCM API üzerinden bildirim gönderilir
* Mesaj Özeti: chats ana belgesinde son mesaj bilgisi güncellenir

**Kodun işleyişi:**

1. Kullanıcı mesajı yazıp gönderdiğinde, mesaj messages koleksiyonuna kaydedilir.
2. Ana chat belgesine lastMessage, lastMessageTime ve lastSenderId güncellenir.
3. sendPushNotification() fonksiyonu çalışarak karşı kullanıcıya mesaj bildirimi gönderilir.
4. Kullanıcı, gönderdiği ve aldığı tüm mesajları anlık olarak görebilir.

metin, ekran görüntüsü, mektup, harf, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 28. Mesajlarım Ekranı

**Şekil 28. Mesajlarım Ekranı**

Mesajlarım ekranı, kullanıcının daha önce başka kullanıcılarla gerçekleştirdiği sohbetleri toplu şekilde listeleyen ekran olarak yapılandırılmıştır. Her kart yapısı bir sohbete karşılık gelir ve bu kartlar kullanıcının gönderdiği veya aldığı son mesaj bilgilerini içerir.

Ekranda yer alan her sohbet kartı, kullanıcının mesajlaştığı kişinin adı, sohbetin ait olduğu ilan başlığı ve son mesaj içeriğini gösterir. Mesaj sahibi ve kullanıcı rolleri kontrol edilerek başlık dinamik olarak “Senin mesajın” ya da “İlanına gelen mesaj” şeklinde belirlenir. Kullanıcı, bu liste üzerinden önceki konuşmalarına hızlıca erişebilir ve kartlara tıklayarak ChatScreen sayfasına yönlendiriliyor (Şekil 26).

**Teknik Özellikler:**

* **Veri Kaynağı:** Firestore – chats, users, listings koleksiyonları
* **Gerçek Zamanlı Veri:** StreamBuilder ile kullanıcının dahil olduğu sohbetler (users içinde currentUserId) anlık olarak dinlenir.
* **Sıralama:** Sohbetler lastMessageTime alanına göre azalan sırayla listelenir.
* **Dinamik Veri Erişimi:** Her kart için, ilgili listingId ve karşı kullanıcı userId bilgileri kullanılarak ilanın başlığı ve kullanıcı adı FutureBuilder ile çekilir.
* **Navigasyon:** ListTile öğesine tıklanarak ilgili sohbet ekranına (ChatScreen) geçiş sağlanır.

**Kodun İşleyişi:**

1. StreamBuilder, chats koleksiyonundan kullanıcının dahil olduğu tüm sohbetleri dinler.
2. Her sohbet için, karşı kullanıcı ve ilgili ilana ait Firestore belgeleri Future.wait() ile paralel olarak getirilir.
3. Kullanıcının rolüne göre başlık belirlenir:
   * Eğer kullanıcı ilanın sahibi ise: İlanına gelen mesaj: [başlık]
   * Değilse: Senin mesajın: [başlık]
4. Liste elemanına tıklanıldığında ChatScreen(chatId: ...) ekranına yönlendirilir.
5. Ekran her an güncel kalır çünkü Firestore veri değişimlerinde yeniden tetiklenir.

**Eklenen Diğer Özellik ve Güncellemeller**

metin, ekran görüntüsü, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, ekran görüntüsü, işletim sistemi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 29. Yenilenmiş Anasayfa Şekil 30. Yan Menü (Drawer)

Ana ekran (HomeScreen), uygulamanın giriş noktası olup kullanıcının hem kendi verilerini görebildiği hem de farklı işlem ekranlarına kolaylıkla ulaşabildiği bir merkez olarak tasarlanmıştır. Bu ekran, Drawer (yan menü) ve kategori kartlarıyla donatılmış ana gövde olmak üzere iki bölümden oluşur.

Yeni tasarımda hem görsel bütünlük hem de kullanıcı deneyimi ön planda tutulmuştur. Kullanıcının adı, e-posta adresi ve profil fotoğrafı yan menüde üst kısımda UserAccountsDrawerHeader bileşeniyle gösterilmiştir.

**Teknik Özellikler:**

* **Drawer:** UserAccountsDrawerHeader + ListTile kombinasyonu ile oluşturulmuştur.
* **Veri Kaynağı:** Firebase Auth → Kullanıcının UID’si üzerinden users koleksiyonundan name, email, profile\_picture alanları çekilir.
* **Responsive Yapı:** Drawer genişliği dinamik olarak ekran genişliğinin %75’i olarak ayarlanır.
* **Kategori Listeleme:** ListView üzerinde Card içinde ListTile kullanılarak kategoriye özel yönlendirme yapılır.
* **FloatingActionButton:** showModalBottomSheet fonksiyonu ile alt ilan türü seçeneklerini getirir. (+ İşareti ilan ekleme ekranlarına yönlendirme amaçlıdır).

**Kodun İşleyişi:**

1. Kullanıcı Bilgileri Getirilir

Sayfa açıldığında, aktif kullanıcı Firebase Authentication üzerinden alınır ve Firestore’daki kullanıcı bilgileri (name, profile\_picture) çekilir. Bu bilgiler yan menüde gösterilir.

2. Yan Menü (Drawer) Oluşturulur

* Kullanıcının adı, maili ve fotoğrafı üstte gösterilir (UserAccountsDrawerHeader).
* Aşağıda, kullanıcının ilanlarına, kanallarına ve ayarlara gidebileceği ListTile butonları bulunur.
* “Çıkış Yap” seçeneği ile kullanıcı oturumdan çıkartılır.

3. + Butonu ile İlan Ekleme

Ekranın sağ altındaki FloatingActionButton tıklanınca ilan türü seçimi için bir alt menü (showModalBottomSheet) açılır. İlan türlerinden birini seçerek ilgili ekleme ekranına yönlendirilir.

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 31. Profil Güncelleme

Ekranı

**Şekil 31. Profil Bilgileri Ekranı**

Profil Bilgileri ekranı, kullanıcıların kişisel bilgilerini güncelleyebileceği bir arayüz olarak tasarlanmıştır. Bu ekran üzerinden ad-soyad, telefon, şehir, ilçe ve mahalle bilgileri düzenlenebilir. Ayrıca kullanıcı profil fotoğrafını da değiştirebilir.

**Teknik Özellikler:**

* **Veri Kaynağı:** Firestore – users koleksiyonu
* **Depolama:** Firebase Storage – Profil fotoğrafı için
* **Kimlik Doğrulama:** Firebase Auth – Aktif kullanıcıyı tanımlamak için
* **Görsel Seçimi:** image\_picker paketi kullanılarak galeri erişimi sağlanır

**Kodun İşleyişi:**

1. Kullanıcı Bilgilerinin Yüklenmesi: initState() içinde \_loadUserData() fonksiyonu çağrılır. Bu fonksiyon, Firestore'dan aktif kullanıcıya ait users/{uid} belgesini çeker ve form alanlarına otomatik olarak doldurur.
2. Fotoğraf Seçimi ve Yükleme:

* Kullanıcı, profil resmine tıkladığında ImagePicker kullanılarak galeri açılır.
* Yeni resim seçilirse \_pickedImage değişkenine atanır.
* Kaydetme işleminde fotoğraf Firebase Storage'a yüklenir ve bağlantısı Firestore'a yazılır.

1. Bilgilerin Güncellenmesi:

* Form alanlarının doğruluğu validate() ile kontrol edilir.
* Kaydet butonuna basıldığında \_saveProfile() çağrılır.
* Gerekirse fotoğraf yüklenir, ardından tüm bilgiler users/{uid} belgesi içinde güncellenir.

1. Geri Bildirim ve Yönlendirme: Güncelleme tamamlandıktan sonra kullanıcıya SnackBar ile bilgi verilir ve bir önceki sayfaya (Navigator.pop) dönülür.

metin, harita, ekran görüntüsü, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 32. Hastalık Bildirme Ekranı

**Şekil 32. Hastalık Bildirme Ekranı**

Hastalık Bildir ekranı, kullanıcıların yaşadıkları bölgede karşılaştıkları hayvan hastalıklarını bildirebilmeleri amacıyla tasarlanmıştır. Bu ekran üzerinden kullanıcı şehir ve ilçe bilgilerini girerek harita üzerinde konum seçebilir, hastalığa dair başlık ve açıklama ekleyerek bildirimi gönderebilir.

**Teknik Özellikler:**

* **Veri Kaynağı:** Firestore – hastalik\_bildirimleri koleksiyonu
* **Harita:** flutter\_map ve latlong2 paketleri kullanılarak OpenStreetMap tabanlı dinamik harita gösterimi yapılır
* **Konum Dönüştürme:** geocoding paketi kullanılarak şehir/ilçe metni koordinata çevrilir
* **Kullanıcı Etkileşimi:** Haritaya dokunarak konum seçilebilir, ayrıca şehir ve ilçe girilerek otomatik olarak o bölgeye yakınlaştırma yapılabilir

**Kodun İşleyişi:**

1. **Konum Seçimi:**

* Kullanıcı haritaya tıklayarak selectedLocation değişkenine bir koordinat atar.
* Alternatif olarak şehir ve ilçe alanları doldurulup “Git” butonuna basıldığında, geocoding paketi ile bu metin adres bilgisi koordinata dönüştürülerek harita o bölgeye odaklanır.

1. **Harita Üzerinde İşaretleme:**

* Kullanıcı konum seçtiğinde FlutterMap bileşenine MarkerLayer ile kırmızı bir iğne (marker) eklenir.
* Bu marker haritada kullanıcının seçtiği noktayı temsil eder.

1. **Form Alanları ve Veri Girişi:**

* Başlık ve açıklama alanları doldurularak kullanıcı tarafından hastalık bildirimi hazırlanır.
* Eksik bilgi olması durumunda SnackBar ile kullanıcı uyarılır.

1. **Veri Kaydetme:**

* \_submitReport() fonksiyonu ile form verileri Firebase Firestore’a kaydedilir.
* Kaydedilen veri içinde: başlık, açıklama, latitude/longitude, şehir, ilçe, tarih ve onay durumu yer alır.
* Konuma ait şehir ve ilçe bilgileri ters geocoding işlemi ile otomatik alınır.

1. **Geri Bildirim ve Ekran Kapanışı:**

* Başarılı bildirim sonrası kullanıcıya bilgi verilir.
* Bildirim gönderildikten sonra ekran Navigator.pop(context) ile kapatılır.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 33. Hastalık Onaylama

(Admin) Ekranı

**Şekil 33. Hastalık Bildirimleri Yönetim Ekranı (Admin Paneli)**

Bu ekran, yalnızca yönetici yetkisine sahip kullanıcılar tarafından erişilebilen hastalık bildirimi onaylama panelidir. Kullanıcılar tarafından bildirilen tüm hastalıklar burada listelenir. Admin, her bildirimi ayrı ayrı inceleyerek “Onayla” butonuna tıklayarak bildirimleri doğrulayabilir. Onaylanan bildirime ait bölgedeki tüm kullanıcılar anlık olarak bilgilendirilir.

**Teknik Özellikler:**

* **Veri Kaynağı:** Firestore – hastalik\_bildirimleri koleksiyonu
* **Bildirim Servisi:** Firebase Cloud Messaging (FCM) – googleapis\_auth paketi üzerinden erişim
* **Kimlik Denetimi:** Firebase Auth – Admin yetkili kullanıcı girişi ile erişim sağlanır
* **Bildirim Hedeflemesi:** Firestore’daki users koleksiyonundaki city alanına göre filtreleme yapılır

**Kodun İşleyişi:**

* 1. **Verilerin Listelenmesi:**
* StreamBuilder ile hastalik\_bildirimleri koleksiyonu anlık olarak dinlenir.
* Kayıtlar tarihe göre tersten sıralanarak listelenir.
* Her kayıt bir Card bileşeni içinde başlık, açıklama ve onay durumu ile gösterilir.
  1. **Bildirimi Onaylama:**
* Henüz onaylanmamış bir kayıt için “Onayla” butonu görünür.
* Butona basıldığında önce Firestore’da ilgili belgeye {onayli: true} güncellemesi yapılır.
  1. **Kullanıcılara Bildirim Gönderme:**
* Bildirilen hastalığın city alanına göre Firestore’daki users koleksiyonundan FCM token’ları toplanır.
* Her kullanıcıya ayrı ayrı, “Hastalık Bildirimi” ve “Şehir” bölgesinde onaylanan hastalık bildirimi var içeriğiyle FCM bildirimi gönderilir.

(Test Edilmedi!)

* Bildirim gönderme işlemi googleapis\_auth ile servis hesabı kullanılarak yapılır.
  1. **Durum Geri Bildirimi:**
* Eğer o şehirde kullanıcı bulunmazsa, admin'e "Bu şehirde bildirim alacak kullanıcı yok." uyarısı gösterilir.
* Aksi durumda bildirim gönderim sonuçları konsola yazdırılır (print).

metin, harita, atlas, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 34. Hastalık Duyuruları

Harita Ekranı

**Şekil 34. Hastalık Duyuruları Harita Ekranı**

Bu ekran, kullanıcıların onaylanmış hastalık bildirimlerini coğrafi konumlarına göre harita üzerinde görmelerini sağlayan bir arayüzdür. Bildirimler harita üzerinde kırmızı renkle işaretlenmiş pinler ve çevresinde 3 kilometrelik yarıçap çemberler ile gösterilir. Her bir işaretlenmiş bölgeye dokunulduğunda bildirimle ilgili detaylar açılır.

**Teknik Özellikler:**

* **Veri Kaynağı:** Firestore – hastalik\_bildirimleri koleksiyonu
* **Harita Kütüphanesi:** flutter\_map ve latlong2
* **Harita Sağlayıcısı:** OpenStreetMap (TileLayer)
* **Kapsam Göstergesi:** CircleMarker – 3000 metre yarıçapla hastalıklı alanın çevresi belirtilir

**Kodun İşleyişi:**

1. **Verilerin Haritaya Aktarılması:**

* StreamBuilder kullanılarak onayli alanı true olan tüm bildirimler dinlenir.
* Firestore'dan gelen her kayıt için:
  + - Marker bileşeni ile harita üzerinde konum işaretlenir.
    - CircleMarker bileşeni ile aynı noktaya 3 km yarıçaplı şeffaf kırmızı alan çizilir.

1. **Harita Konumlandırması:**

* Eğer veri varsa ilk bildirimin konumu harita merkezine alınır.
* Aksi takdirde varsayılan merkez (LatLng(38.0, 35.0)) ve zoom değeri (6.0) kullanılır.

1. **Detay Bilgi Görüntüleme:**

* Kullanıcı herhangi bir işaretçi simgesine dokunduğunda \_showDetail() fonksiyonu çalışır.
* Açılan ModalBottomSheet içerisinde bildirimin başlığı, açıklaması, konum bilgisi ve tarih/saati görüntülenir.