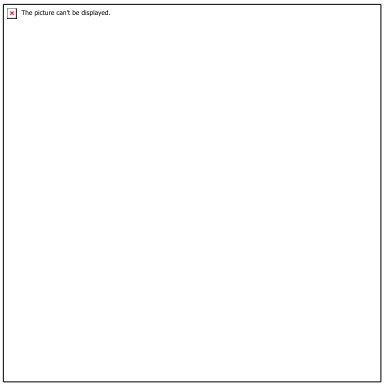
ANKARA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



BLM 4538

PROJE RAPORU

Anlatım Video Linki : https://youtu.be/JPC00LLgf9w

Proje Github linki : https://github.com/seyman123/fitness-app

Fitness Mobil Uygulama

İbrahim Seyman

20290286

Enver Bağcı

Ocak, 2026

İÇİNDEKİLER

1. GİRİŞ

Günümüzde teknolojinin hızla gelişmesiyle birlikte mobil uygulamalar hayatımızın vazgeçilmez bir parçası haline gelmiştir. Özellikle sağlık ve fitness alanında mobil uygulamaların kullanımı her geçen gün artmaktadır. İnsanlar artık spor salonlarına gitmeden, kişisel antrenör tutmadan kendi sağlık ve fitness hedeflerini takip edebilmek istemektedir.

Bu proje kapsamında, kullanıcıların günlük sağlık ve fitness aktivitelerini tek bir platformdan yönetebilecekleri kapsamlı bir mobil uygulama geliştirilmiştir. Uygulama, Flutter framework kullanılarak cross-platform olarak geliştirilmiş olup hem Android hem de iOS işletim sistemlerinde çalışabilmektedir.

1.1. Projenin Amacı

Bu projenin temel amacı, kullanıcılara sağlıklı yaşam hedeflerine ulaşmalarında yardımcı olacak, kullanımı kolay ve kapsamlı bir fitness takip uygulaması geliştirmektir. Uygulama aşağıdaki temel hedefleri gerçekleştirmeyi amaçlamaktadır:

- Kullanıcıların günlük su tüketimini takip etmesi ve su içme alışkanlığı kazanmasına yardımcı olmak
- Beslenme alışkanlıklarının kaydedilmesi ve kalori takibi yapılması
- Kişiselleştirilmiş antrenman programları oluşturulması ve takip edilmesi
- Kilo değişimlerinin izlenmesi ve BMI hesaplaması
- Günlük adım sayısının takibi
- Tüm verilerin grafiksel olarak analiz edilmesi

1.2. Projenin Kapsamı

Proje, tam kapsamlı bir fitness takip sistemi olarak tasarlanmıştır. Sistem, iki ana bileşenden oluşmaktadır:

1. Mobil Uygulama (Frontend): Flutter framework kullanılarak geliştirilen, kullanıcı dostu arayüze sahip mobil uygulama. Bu uygulama Android ve iOS platformlarında çalışabilmektedir.

2. Sunucu (Backend): Node.js ve Express.js kullanılarak geliştirilen RESTful API servisi. Bu servis, tüm veri işlemlerini yönetmekte ve güvenli bir şekilde veritabanı ile iletişim kurmaktadır.

1.3. Problemin Tanımı

Mevcut fitness uygulamalarının çoğu ya çok karmaşık ya da çok basit kalmaktadır. Kullanıcılar genellikle birden fazla uygulama kullanmak zorunda kalmaktadır: bir uygulama su takibi için, başka bir uygulama kalori sayımı için, bir diğeri de antrenman takibi için. Bu durum kullanıcı deneyimini olumsuz etkilemekte ve verilerin bütünleşik bir şekilde analiz edilmesini zorlaştırmaktadır.

Bu proje, tüm bu özellikleri tek bir uygulamada birleştirerek kullanıcılara bütünleşik bir çözüm sunmayı hedeflemektedir. Ayrıca yerli bir uygulama olarak Türkçe dil desteği ve Türk kullanıcıların alışkanlıklarına uygun özellikler içermektedir.

2. KULLANILAN TEKNOLOJİLER

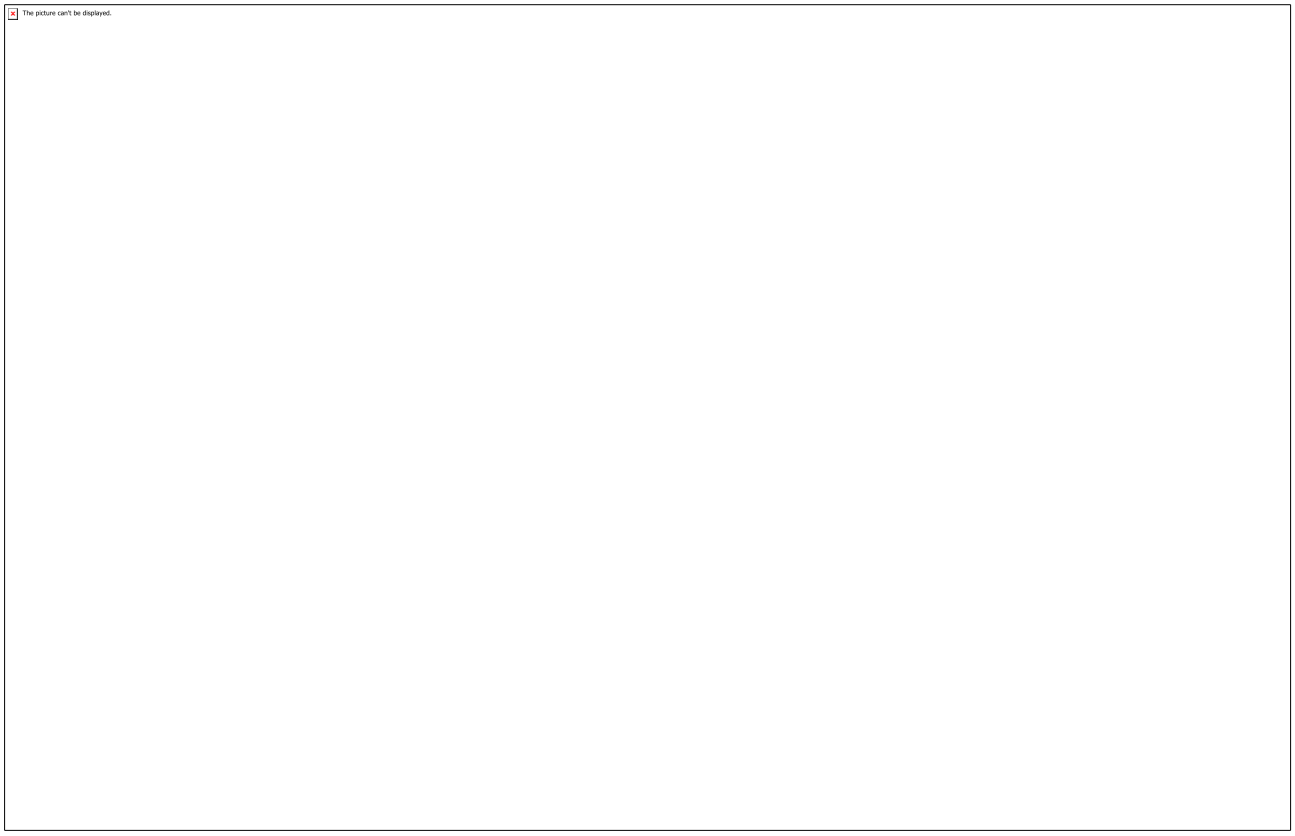
Bu bölümde, projenin geliştirilmesinde kullanılan teknolojiler, framework'ler ve kütüphaneler detaylı olarak açıklanmaktadır.

2.1. Flutter Framework

Flutter, Google tarafından geliştirilen açık kaynaklı bir UI (Kullanıcı Arayüzü) framework'üdür. Tek bir kod tabanı ile hem Android hem de iOS için native performansta uygulamalar geliştirmeye olanak sağlamaktadır. Bu projede Flutter tercih edilmesinin başlıca nedenleri şunlardır:

- **Cross-Platform Geliştirme:** Tek kod tabanı ile birden fazla platformda çalışabilme
- **Hot Reload:** Kod değişikliklerinin anında görülebilmesi, geliştirme sürecini hızlandırması
- **Zengin Widget Kütüphanesi:** Material Design ve Cupertino widget'ları ile modern arayüzler
- **Yüksek Performans:** Native koda derlenmesi sayesinde yüksek performans
- **Geniş Topluluk Desteği:** Aktif geliştirici topluluğu ve zengin paket ekosistemi

Flutter, Dart programlama dilini kullanmaktadır. Dart, nesne yönelimli, sınıf tabanlı bir programlama dilidir ve öğrenmesi kolay bir sözdizimine sahiptir.



2.2. Node.js ve Express.js

Backend tarafında Node.js runtime ortamı ve Express.js web framework'ü kullanılmıştır.

Node.js, Chrome V8 JavaScript motoru üzerine inşa edilmiş bir JavaScript runtime ortamıdır. Event-driven, non-blocking I/O modeli sayesinde hafif ve verimli uygulamalar geliştirmeye olanak sağlamaktadır.

Express.js, Node.js için minimal ve esnek bir web uygulama framework'üdür. RESTful API geliştirmek için sıklıkla tercih edilmektedir. Bu projede Express.js kullanılarak tüm API endpoint'leri oluşturulmuştur.

2.3. Veritabanı (SQLite ve Prisma)

Projede veritabanı yönetimi için SQLite veritabanı ve Prisma ORM kullanılmıştır.

SQLite, sunucu gerektirmeyen, dosya tabanlı hafif bir veritabanı motorudur. Kurulumu kolay olması ve mobil uygulamalar için ideal olması nedeniyle tercih edilmiştir.

Prisma, modern bir ORM (Object-Relational Mapping) aracıdır. Tip güvenli veritabanı erişimi sağlamakta ve veritabanı şemasını kod olarak tanımlamaya olanak vermektedir. Prisma'nın sağladığı avantajlar:

- Otomatik tip oluşturma
- Güvenli veritabanı sorguları
- Kolay migration yönetimi
- Veritabanı şemasının görselleştirilmesi

2.4. Diğer Kütüphaneler

Projede kullanılan diğer önemli kütüphaneler aşağıdaki tabloda listelenmiştir:

Kütüphane	**Versiyon**	**Kullanım Amacı**
flutter_bloc	8.1.3	State management için BLoC pattern implementasyonu
get_it	7.6.0	Dependency injection için service locator
fl_chart	0.68.0	Grafik ve chart gösterimi

http	1.1.0	HTTP istekleri için
shared_preferences	2.2.2	Yerel veri saklama
jsonwebtoken (Backend)	9.0.2	JWT token oluşturma ve doğrulama
bcryptjs (Backend)	3.0.3	Şifre hashleme

3. SİSTEM ANALİZİ VE TASARIMI

3.1. Gereksinim Analizi

Proje geliştirme sürecinin başında kapsamlı bir gereksinim analizi yapılmıştır. Bu analiz sonucunda belirlenen fonksiyonel ve fonksiyonel olmayan gereksinimler aşağıda listelenmiştir.

Fonksiyonel Gereksinimler:

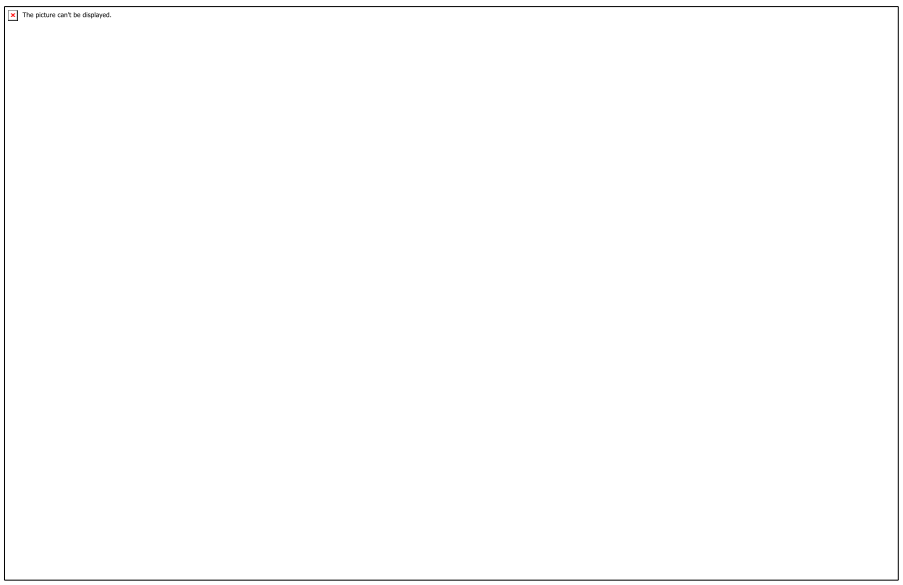
- Kullanıcılar sisteme kayıt olabilmeli ve giriş yapabilmeli
- Kullanıcılar profil bilgilerini (yaş, boy, kilo, hedef) girebilmeli
- Günlük su tüketimi kaydedilebilmeli ve takip edilebilmeli
- Öğün bazlı beslenme kaydı yapılabilmeli
- Antrenman programları oluşturulabilmeli ve düzenlenebilmeli
- Hazır antrenman programları sunulmalı
- Kilo değişimleri kaydedilebilmeli
- Tüm veriler grafiksel olarak görüntülenebilmeli

Fonksiyonel Olmayan Gereksinimler:

- Uygulama hızlı ve responsive olmalı
- Kullanıcı verileri güvenli bir şekilde saklanmalı
- Arayüz kullanıcı dostu ve sezgisel olmalı
- Uygulama çevrimdışı modda temel işlevleri yerine getirebilmeli

3.2. Yazılım Mimarisi (Clean Architecture)

Projede, yazılım kalitesini ve sürdürülebilirliğini artırmak amacıyla Clean Architecture (Temiz Mimari) prensibi uygulanmıştır. Bu mimari yaklaşım, Robert C. Martin (Uncle Bob) tarafından önerilmiştir ve yazılımın bağımsız, test edilebilir ve bakımı kolay katmanlara ayrılmasını sağlamaktadır.



Projede her bir özellik (feature) modülü üç ana katmandan oluşmaktadır:

1. Data Katmanı: Bu katman, veri kaynaklarıyla (API, veritabanı) iletişimi yönetir. Data source'lar, model sınıfları ve repository implementasyonları bu katmanda yer alır.

2. Domain Katmanı: İş mantığının yer aldığı katmandır. Entity sınıfları ve repository arayüzleri bu katmanda tanımlanır. Bu katman herhangi bir framework'e bağımlı değildir.

3. Presentation Katmanı: Kullanıcı arayüzü ve state management bu katmanda yer alır. Widget'lar, BLoC sınıfları ve sayfa bileşenleri bu katmandadır.

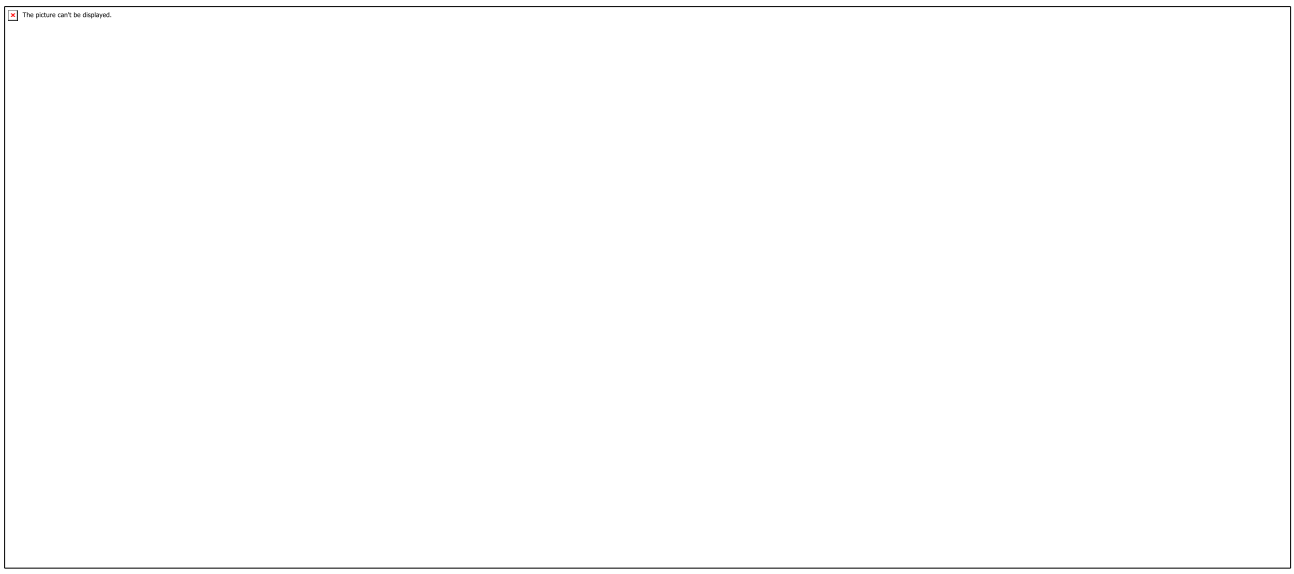
Proje klasör yapısı aşağıdaki gibidir:

```
lib/
├── core/ # Ortak kullanılan kodlar
│  ├── config/ # API yapılandırması
│  ├── di/ # Dependency Injection
│  ├── error/ # Hata yönetimi sınıfları
│  ├── network/ # API istemcisi
│  └── theme/ # Uygulama teması
│
├── features/ # Özellik modülleri
│  ├── auth/ # Kimlik doğrulama
│  ├── nutrition/ # Beslenme takibi
│  ├── profile/ # Kullanıcı profili
│  ├── water/ # Su takibi
│  ├── weight/ # Kilo takibi
│  ├── workout/ # Antrenman
│  ├── steps/ # Adım takibi
│  ├── statistics/ # İstatistikler
│  └── tracking/ # Dashboard
```

3.3. Veritabanı Tasarımı

Veritabanı tasarımında ilişkisel veritabanı modeli kullanılmıştır. Prisma ORM ile tanımlanan veritabanı şeması aşağıdaki tabloları içermektedir:

Tablo Adı	**Açıklama**	**Temel Alanlar**
User	Kullanıcı hesap bilgileri	id, email, password, name, createdAt
UserProfile	Kullanıcı profil detayları	age, gender, height, weight, goalWeight, activityLevel
WaterTracking	Su tüketim kayıtları	userId, amount, date
NutritionLog	Beslenme günlüğü	userId, mealType, foodName, calories, protein, carbs, fat
Workout	Antrenman programları	userId, name, description, isTemplate
WorkoutExercise	Egzersiz detayları	workoutId, name, sets, reps, restSeconds
WorkoutLog	Tamamlanan antrenmanlar	userId, workoutId, date, duration, completed
StepTracking	Adım sayısı kayıtları	userId, steps, date
WeightHistory	Kilo geçmişi	userId, weight, bmi, date



3.4. API Tasarımı

Backend tarafında RESTful API tasarım prensipleri uygulanmıştır. Tüm endpoint'ler JSON formatında veri alışverişi yapmaktadır. API endpoint'leri aşağıdaki gibi organize edilmiştir:

Endpoint	**Metot**	**Açıklama**
/api/auth/register	POST	Yeni kullanıcı kaydı
/api/auth/login	POST	Kullanıcı girişi
/api/profile	GET/PUT	Profil bilgileri
/api/water	GET/POST/DELETE	Su takibi işlemleri
/api/nutrition	GET/POST/DELETE	Beslenme kayıtları

/api/workouts	GET/POST/PUT/DELETE	Antrenman CRUD işlemleri
/api/workouts/logs	GET/POST	Antrenman tamamlama kayıtları
/api/weight	GET/POST/DELETE	Kilo takibi
/api/statistics	GET	İstatistik verileri

4. UYGULAMA GELİŞTİRME

4.1. Backend Geliştirme

Backend geliştirme sürecinde öncelikle proje yapısı oluşturulmuş, ardından veritabanı şeması tasarlanmıştır. Geliştirme aşamaları şu şekilde ilerlemiştir:

1. Proje Kurulumu: Node.js projesi oluşturulmuş ve gerekli bağımlılıklar yüklenmiştir. Express.js framework'ü temel alınarak uygulama yapısı kurulmuştur.

2. Veritabanı Entegrasyonu: Prisma ORM kurulmuş ve veritabanı şeması tanımlanmıştır. Migration işlemleri ile veritabanı tabloları oluşturulmuştur.

3. Kimlik Doğrulama: JWT (JSON Web Token) tabanlı kimlik doğrulama sistemi implementete edilmiştir. Şifreler bcrypt algoritması ile hashlenerek güvenli bir şekilde saklanmaktadır.

4. API Geliştirme: Her özellik için controller ve route dosyaları oluşturulmuştur. Joi kütüphanesi ile input validasyonu sağlanmıştır.

Backend klasör yapısı:

```
backend/src/
├── app.js # Express uygulama konfigürasyonu
├── server.js # Sunucu başlatma
├── config/
│ └── database.js # Veritabanı bağlantısı
├── controllers/ # İş mantığı
├── middleware/ # Auth middleware
├── routes/ # API endpoint tanımları
└── utils/ # Yardımcı fonksiyonlar
```

4.2. Frontend Geliştirme

Flutter ile frontend geliştirme sürecinde Clean Architecture prensiplerine uygun bir yapı kurulmuştur. Her özellik kendi klasöründe izole edilmiş ve bağımsız olarak geliştirilebilir hale getirilmiştir.

Geliştirme sürecinde dikkat edilen noktalar:

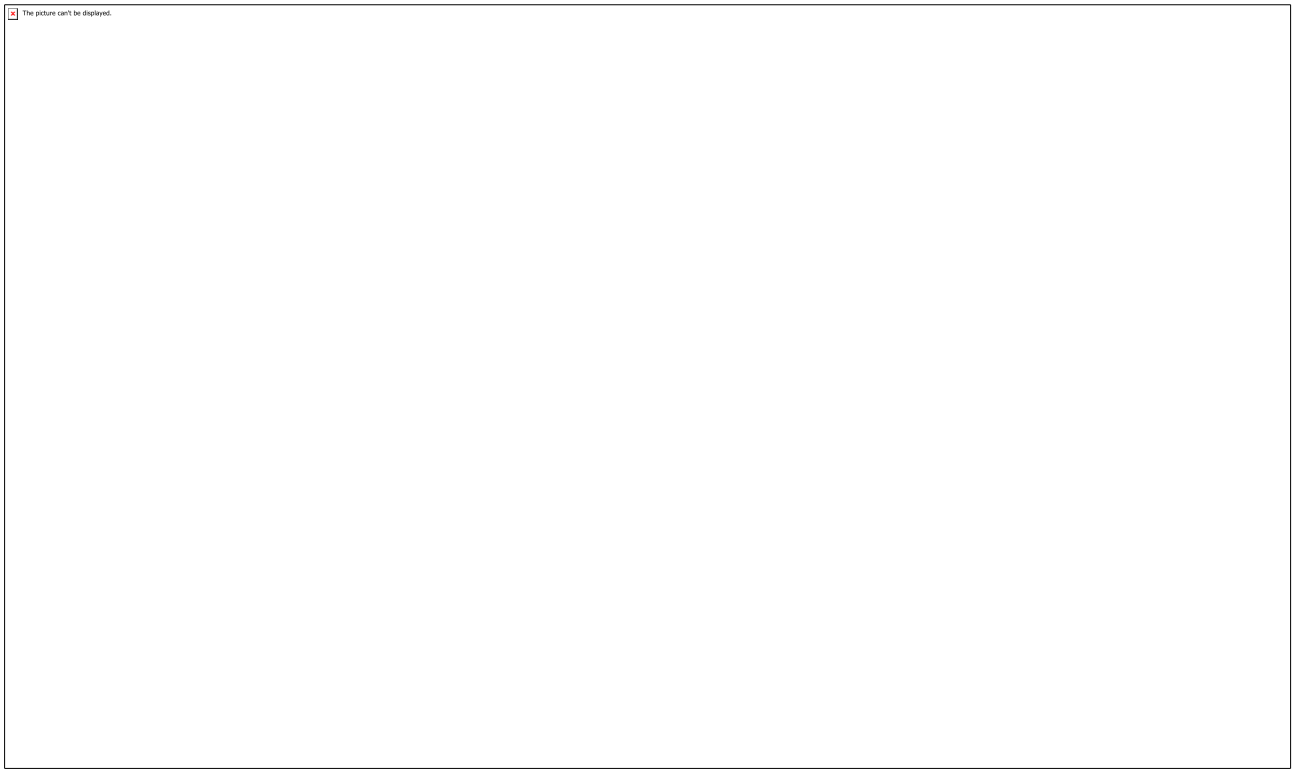
- **Widget Ayrımı:** Büyük widget'lar küçük, yeniden kullanılabilir parçalara bölünmüştür
- **Tema Kullanımı:** Tutarlı bir görünüm için merkezi tema dosyası oluşturulmuştur
- **Hata Yönetimi:** API hatalarının kullanıcıya uygun şekilde gösterilmesi sağlanmıştır
- **Loading States:** Veri yüklenirken kullanıcıya geri bildirim verilmektedir

4.3. State Management (BLoC Pattern)

Projede state management için BLoC (Business Logic Component) pattern kullanılmıştır. BLoC, Flutter uygulamalarında yaygın olarak kullanılan, reaktif programlama prensiplerine dayanan bir mimari desendir.

BLoC pattern'in temel bileşenleri:

- **Event:** Kullanıcı etkileşimleri veya sistem olaylarını temsil eder
- **State:** Uygulamanın o anki durumunu temsil eder
- **Bloc:** Event'leri alır, iş mantığını uygular ve yeni State üretir



Örnek bir BLoC yapısı (Water modülü):

```
// water_event.dart
abstract class WaterEvent {}
class LoadTodayWater extends WaterEvent {}
class AddWater extends WaterEvent {
  final int amount;
}

// water_state.dart
abstract class WaterState {}
class WaterLoading extends WaterState {}
class WaterLoaded extends WaterState {
  final int totalAmount;
  final List<WaterEntry> entries;
}

// water_bloc.dart
class WaterBloc extends Bloc<WaterEvent, WaterState> {
  // Event'leri dinler ve State üretir
}
```

5. UYGULAMA ÖZELLİKLERİ

5.1. Kullanıcı Yönetimi

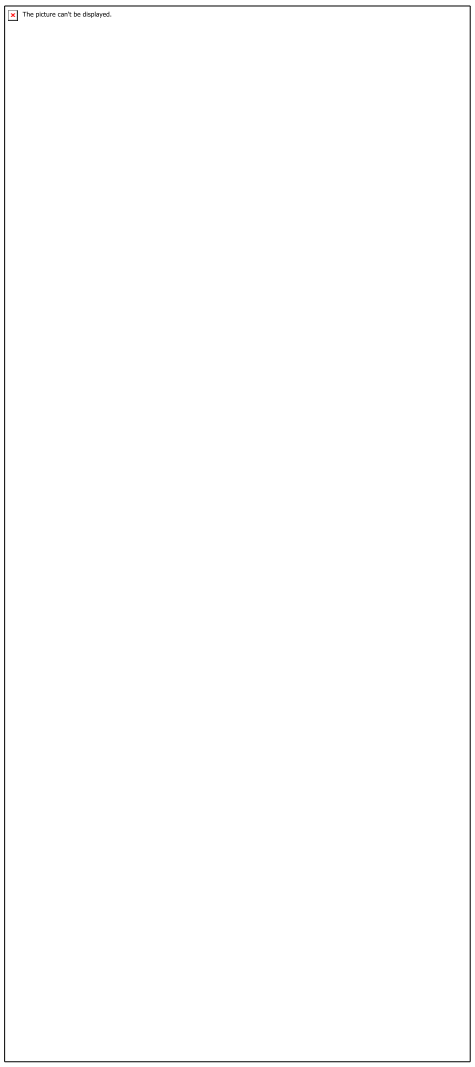
Uygulama, güvenli bir kullanıcı yönetim sistemi içermektedir. Kullanıcılar e-posta adresleri ve şifreleri ile kayıt olabilmekte ve giriş yapabilmektedir.

Kayıt İşlemi: Kullanıcı, e-posta adresi, şifre ve isim bilgilerini girerek kayıt olur. Şifre, bcrypt algoritması ile hashlenerek veritabanına kaydedilir.

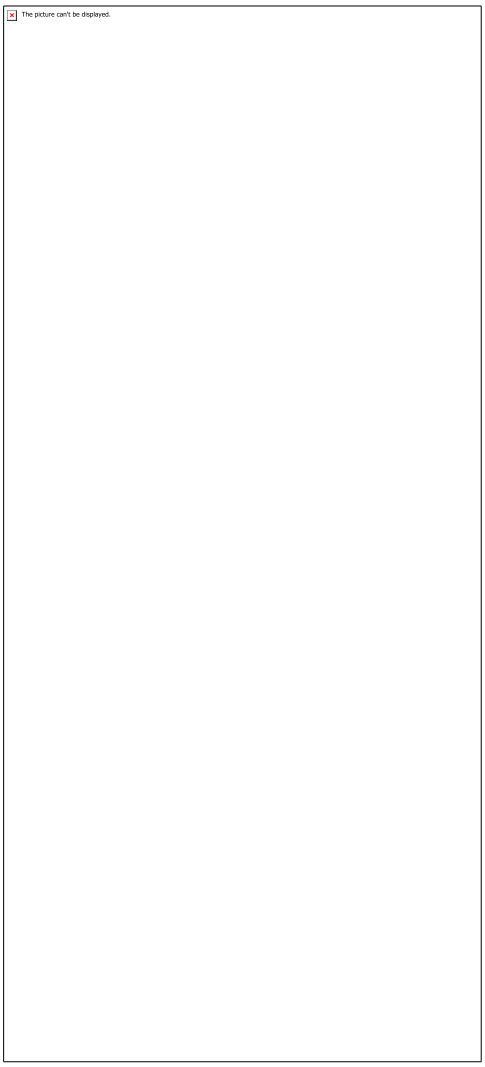
Giriş İşlemi: Kullanıcı, e-posta ve şifresini girerek giriş yapar. Başarılı girişte JWT token oluşturulur ve cihazda saklanır. Bu token, sonraki API isteklerinde kimlik doğrulama için kullanılır.

Profil Kurulumu: İlk girişte kullanıcıdan yaş, cinsiyet, boy, kilo ve hedef bilgileri alınır. Bu bilgiler, kalori hesaplaması ve BMI değerlendirmesi için kullanılır.

Login Ekranı



Kayıt Olma Ekranı

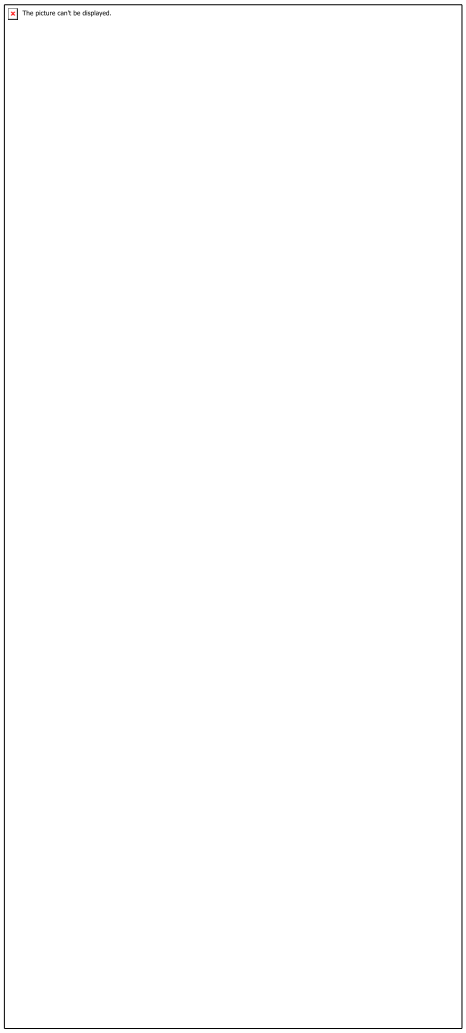


5.2. Su Takibi Modülü

Su takibi modülü, kullanıcıların günlük su tüketimini kaydetmelerini ve takip etmelerini sağlar. Modülün özellikleri:

- Hızlı su ekleme butonları (100ml, 200ml, 300ml, 500ml)
- Özel miktar girişi
- Günlük hedef gösterimi (varsayılan 2000ml)
- İlerleme çubuğu ile görsel takip
- Günlük kayıt geçmişi
- Kayıt silme özelliği

Su Takibi Sayfası

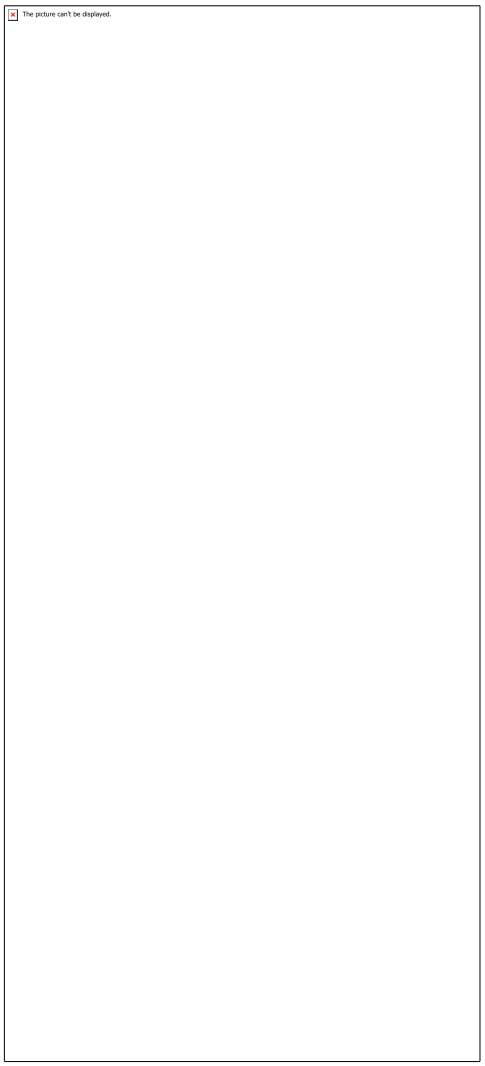


5.3. Beslenme Takibi Modülü

Beslenme modülü, kullanıcıların öğün bazlı yemek kayıtları tutmasını sağlar. Özellikler:

- Öğün kategorileri: Kahvaltı, Öğle Yemeği, Akşam Yemeği, Atıştırmalık
- Yemek adı ve kalori girişi
- Makro besin takibi (protein, karbonhidrat, yağ)
- Günlük toplam kalori hesabı
- Öğün bazlı listeleme

Beslenme Takibi Ekranı



5.4. Antrenman Modülü

Antrenman modülü, uygulamanın en kapsamlı modülüdür. Kullanıcılar kendi antrenman programlarını oluşturabilir veya hazır programlardan seçim yapabilir.

Kişisel Program Oluşturma:

- Program adı ve açıklama girişi
- Egzersiz ekleme (isim, set, tekrar, dinlenme süresi)
- Egzersiz notları
- Sıralama ve düzenleme

Hazır Antrenman Programları:

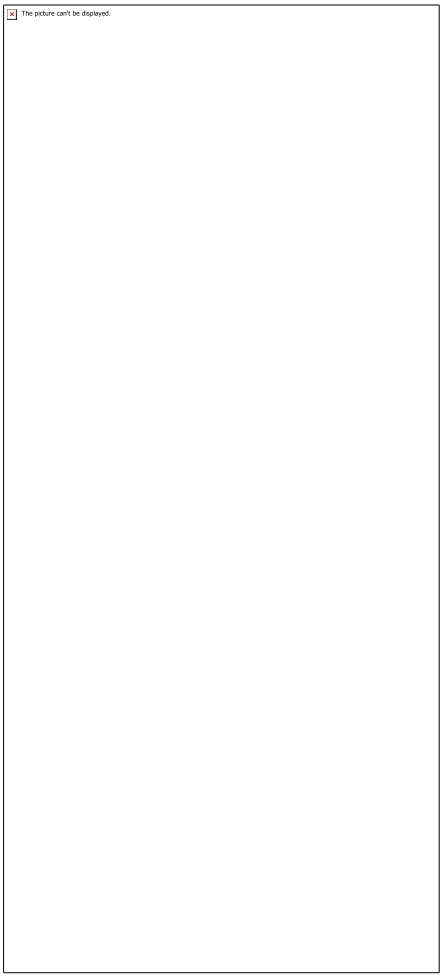
Uygulama, farklı seviyelerde 11 adet hazır antrenman programı içermektedir:

Program Adı	Seviye	Kategori
Full Body Başlangıç	Başlangıç	Full Body
Kardiyo Başlangıç	Başlangıç	Kardiyo
Evde (Ekipmansız)	Başlangıç	Home
Push Day (İtme)	Orta	Split
Pull Day (Çekme)	Orta	Split
Leg Day (Bacak)	Orta	Split
HIIT Kardiyo	Orta	Kardiyo

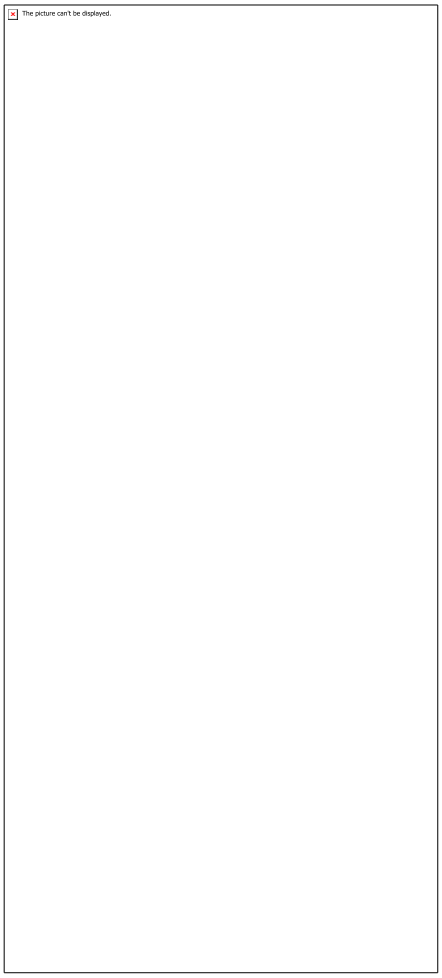
Core & Karın	Orta	Core
Üst Vücut İleri	İleri	Upper Body
Alt Vücut İleri	İleri	Lower Body
Esneme & Mobilite	Başlangıç	Recovery

Antrenman Tamamlama: Kullanıcı bir antrenmanı başlattığında süre ve not girebilir. Tamamlanan antrenmanlar kaydedilir ve dashboard'da gösterilir.

Kullanıcının Eklediği Antrenmanlar



Hazır Antrenman Programları

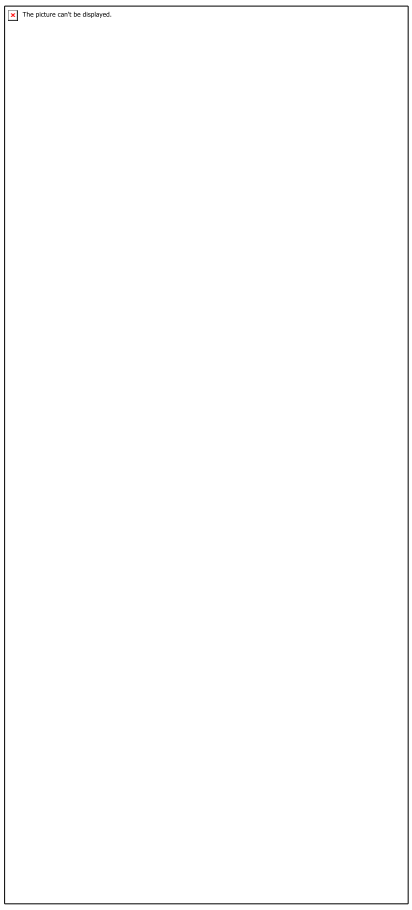


5.5. Kilo Takibi Modülü

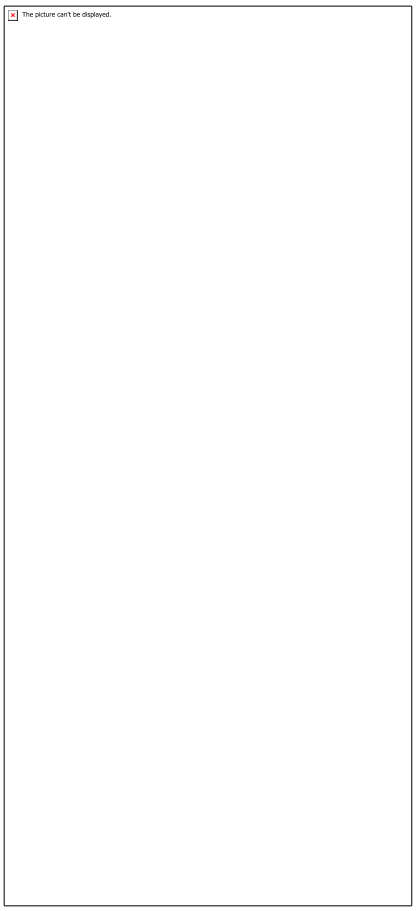
Kilo takibi modülü, kullanıcıların kilo değişimlerini izlemelerini sağlar. Özellikler:

- Kilo girişi ve tarih seçimi
- Otomatik BMI hesaplama
- Hedef kiloya göre ilerleme
- Grafik ile kilo değişim takibi
- Geçmiş kayıtların listelenmesi

Kilo Takibi Sayfası



Kilo Ekleme Alanı



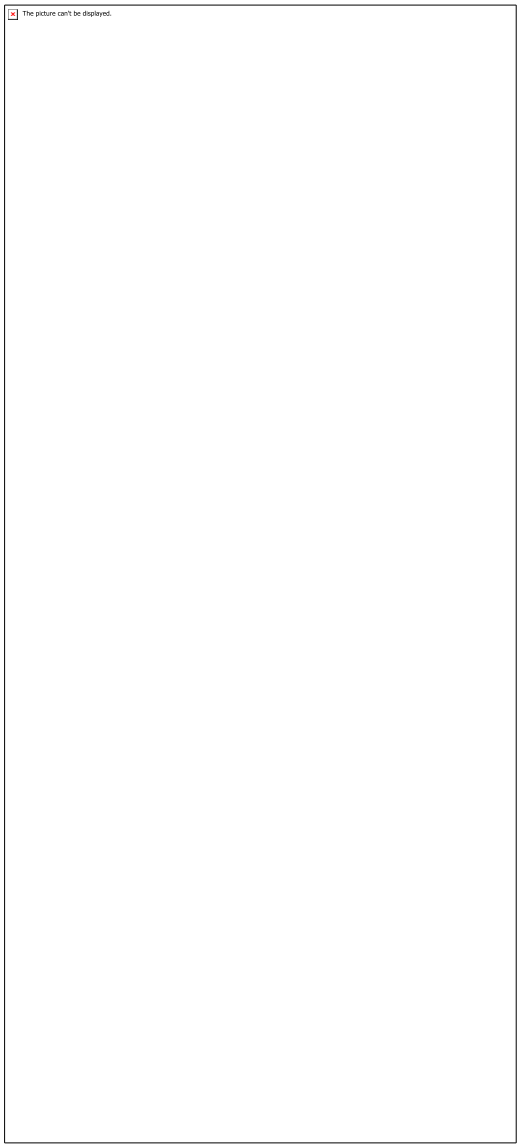
5.6. İstatistikler ve Raporlama

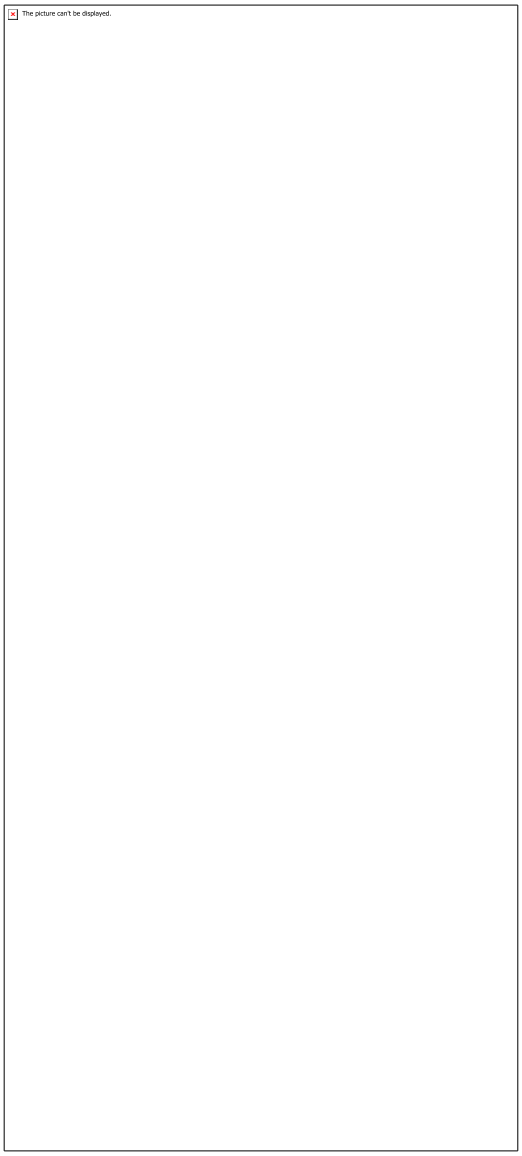
İstatistikler modülü, tüm aktivitelerin grafiksel analizini sunar. fl_chart kütüphanesi kullanılarak çizgi grafikleri, çubuk grafikleri oluşturulmuştur.

Gösterilen istatistikler:

- Haftalık su tüketimi grafiği
- Kalori alımı trendi
- Antrenman süreleri
- Kilo değişim grafiği
- Adım sayısı istatistikleri

İstatistik/Grafik Ekranı

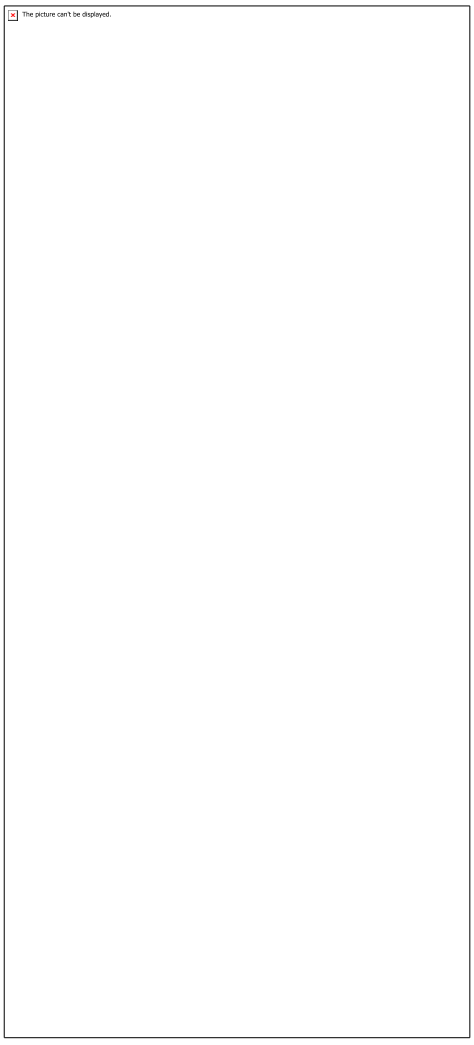




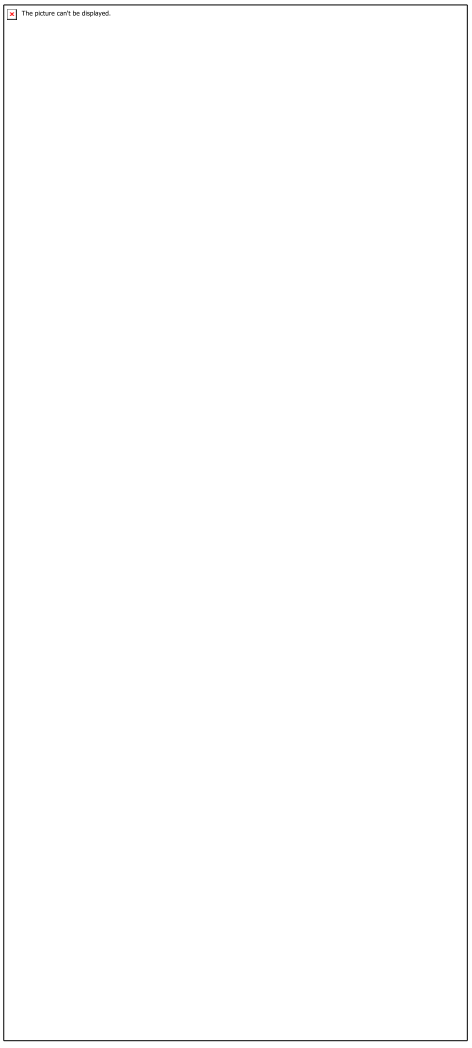
6. EKRAN GÖRÜNTÜLERİ

Bu bölümde uygulamanın çeşitli ekranlarının görüntüleri yer almaktadır.

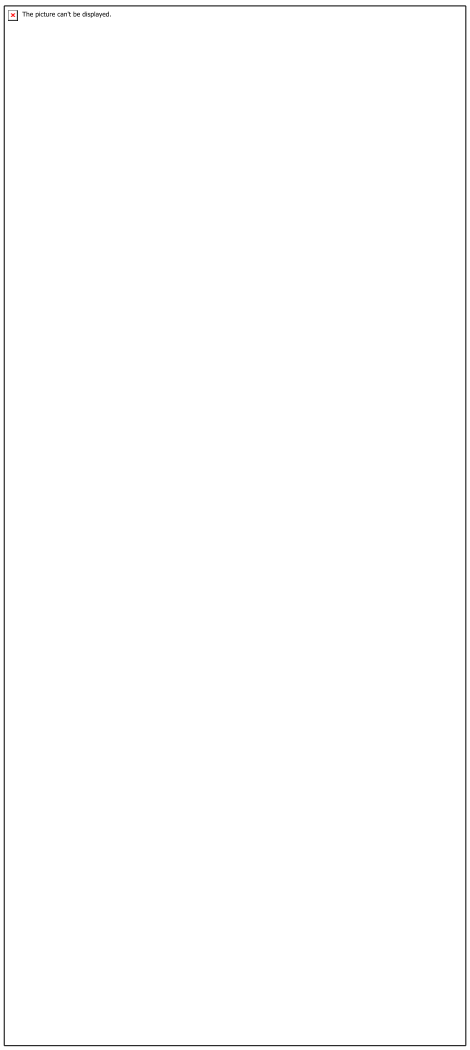
Dashboard



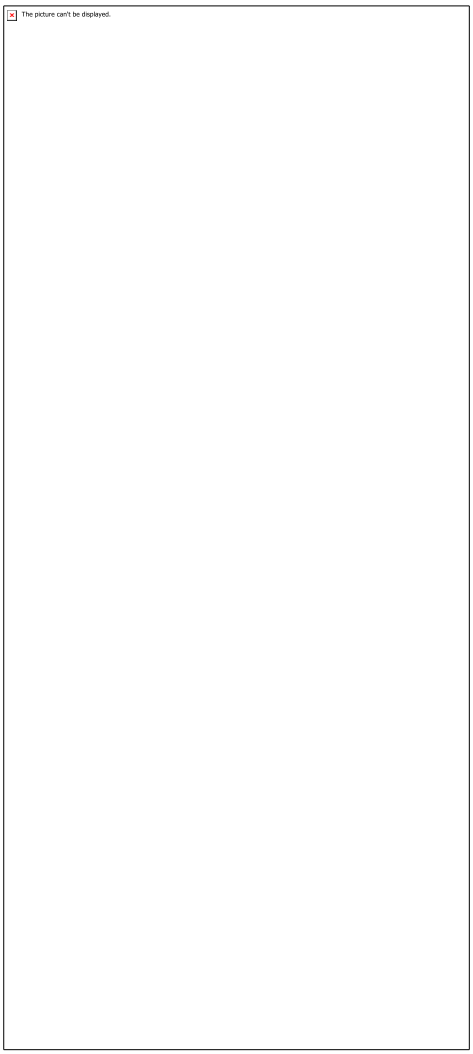
Profil Sayfası



Antrenman Ekleme Sayfası



Bildirim Ayarları Sayfası



7. SONUÇ VE DEĞERLENDİRME

Bu proje kapsamında, kullanıcıların sağlık ve fitness aktivitelerini takip edebilecekleri kapsamlı bir mobil uygulama başarıyla geliştirilmiştir. Uygulama, Flutter framework'ü kullanılarak cross-platform olarak geliştirilmiş olup hem Android hem de iOS platformlarında çalışabilmektedir.

Proje sürecinde Clean Architecture prensibi uygulanmış, bu sayede kod kalitesi ve sürdürülebilirlik artırılmıştır. BLoC pattern ile state management sağlanmış, kullanıcı deneyimi optimize edilmiştir.

7.1. Karşılaşılan Zorluklar

Proje geliştirme sürecinde çeşitli teknik zorluklarla karşılaşılmıştır:

- **State Management:** BLoC pattern'in öğrenilmesi ve doğru implementasyonu başlangıçta zorlayıcı olmuştur. Özellikle event ve state akışının yönetimi deneyim gerektirmiştir.
- **Clean Architecture:** Katmanlı mimari yapının kurulması ve her katmanın sorumluluklarının belirlenmesi zaman almıştır. Ancak bu yapı, projenin ilerleyen aşamalarında büyük kolaylık sağlamıştır.
- **API Entegrasyonu:** Flutter Web ve mobil platformlar için farklı network konfigürasyonları gerekmektedir. CORS ayarları ve localhost/IP adresi farklılıkları çözülmesi gereken sorunlar olmuştur.
- **Veritabanı İlişkileri:** Prisma ORM ile karmaşık ilişkilerin (one-to-many, many-to-many) yönetimi dikkatli planlama gerektirmiştir.

7.2. Gelecek Geliştirmeler

Uygulamanın gelecek versiyonlarında aşağıdaki özellikler eklenebilir:

- **Sosyal Özellikler:** Arkadaş ekleme, ilerleme paylaşma
- **Egzersiz Videoları:** Egzersizlerin doğru yapılışını gösteren videolar
- **Yapay Zeka Entegrasyonu:** Kişiselleştirilmiş antrenman ve beslenme önerileri
- **Giyilebilir Cihaz Entegrasyonu:** Akıllı saat ve fitness bantları ile senkronizasyon
- **Çoklu Dil Desteği:** İngilizce ve diğer diller

Sonuç olarak, bu proje modern mobil uygulama geliştirme tekniklerinin uygulamalı olarak öğrenilmesine ve kapsamlı bir fitness takip sisteminin geliştirilmesine olanak sağlamıştır. Elde edilen deneyim ve bilgiler, gelecekteki yazılım projelerinde değerli bir temel oluşturacaktır.

KAYNAKÇA

Flutter Documentation. (2024). Flutter - Beautiful native apps in record time. https://flutter.dev/docs

Dart Programming Language. (2024). Dart Documentation. https://dart.dev/guides

BLoC Library. (2024). BLoC State Management. https://bloclibrary.dev

Node.js Foundation. (2024). Node.js Documentation. https://nodejs.org/docs

Express.js. (2024). Express - Node.js web application framework. https://expressjs.com

Prisma. (2024). Prisma Documentation - Next-generation ORM. https://www.prisma.io/docs

Martin, R. C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.

Soares, P. (2023). Flutter BLoC Pattern Tutorial. Medium. https://medium.com/flutter-community