T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

SAKARYA
ÜNİVERSİTESİ

MOBİL UYGULAMA GELİŞTİRME ÖDEV RAPORU

Odak Takipli Pomodoro Uyguluaması

HAZIRLAYAN

Enes SOYLU - B221210081 – enes.soylu3@ogr.sakarya.edu.tr

Sakarya

Aralık, 2025

1. Giriş ve Projenin Amacı

Proje Kaynak Kodları (GitHub Repository):

https://github.com/ByThePi/mobil-uygulama-gelistirme-proje.git

Günümüzde mobil cihazların ve sosyal medya uygulamalarının yarattığı "dijital dikkat dağınıklığı", akademik ve profesyonel verimliliği tehdit eden en kritik sorunlardan biri haline gelmiştir. Bu proje kapsamında, kullanıcıların çalışma disiplinini geri kazanmalarına yardımcı olmak amacıyla, React Native (Expo) teknolojisi kullanılarak bir "Odaklanma Takibi ve Raporlama Uygulaması" geliştirilmiştir. Projenin temel çıkış noktası, akıllı telefonları bir dikkat dağıtıcı unsur olmaktan çıkarıp, Pomodoro gibi zaman yönetimi teknikleriyle entegre ederek ölçülebilir bir verimlilik aracına dönüştürmektir.

Uygulamanın teknik amacı, kullanıcının sadece zaman tutmasını sağlamak değil, aynı zamanda odaklanma kalitesini "AppState API" aracılığıyla denetlemektir. Kullanıcı aktif bir seans içerisindeyken uygulamayı arka plana attığında (farklı bir uygulamaya geçtiğinde), sistem bu durumu anlık olarak tespit ederek sayacı duraklatmakta ve bu eylemi bir "odak kaybı" olarak kaydetmektedir. Elde edilen süre ve dikkat dağınıklığı verileri yerel veritabanında (AsyncStorage) saklanarak, kullanıcıya grafiksel raporlar (Dashboard) üzerinden somut gelişim analizleri sunulması hedeflenmiştir.

2. Sistem Mimarisi ve Kullanılan Teknolojiler

2.1. **Sistem Mimarisi** Proje, sürdürülebilirlik ve kodun yeniden kullanılabilirliği (reusability) ilkeleri gözetilerek **Bileşen Bazlı (Component-Based)** bir mimari ile tasarlanmıştır. Uygulama geliştirme sürecinde "Separation of Concerns" (İlgi Alanlarının Ayrımı) prensibi uygulanmış ve proje dizin yapısı işlevlerine göre modüllere ayrılmıştır:

- **Navigasyon Katmanı (src/navigation):** Uygulamanın ana iskeletini oluşturur. TabNavigator, ekranlar arası geçişi ve alt menü yönetimini sağlar.
- **Ekran Katmanı (src/screens):** Kullanıcı ile etkileşime giren ana sayfaları barındırır (HomeScreen ve ReportsScreen). Bu katman, iş mantığını yönetir ve gerekli bileşenleri çağırır.
- **Bileşen Katmanı (src/components):** Uygulama genelinde tekrar eden arayüz elemanları (TimerCircle, CategorySelector, StatCard) burada izole edilmiştir. Bu sayede kod tekrarı önlenmiş ve arayüz tutarlılığı sağlanmıştır.
- **Veri Katmanı (src/utils):** Veritabanı işlemleri ana koddan soyutlanmıştır. Verilerin kaydedilmesi ve okunması işlemleri storage.js içerisindeki asenkron fonksiyonlarla yönetilir.

2.2. **Kullanılan Teknolojiler ve Kütüphaneler** Proje geliştirme sürecinde **React Native** framework'ü ve **Expo** ekosistemi tercih edilmiştir. Kullanılan temel teknolojiler ve görevleri şunlardır:

- **React Native (Expo):** Cross-platform mobil uygulama geliştirme altyapısı olarak kullanılmıştır.
- **React Navigation (Bottom Tabs):** Uygulama içerisinde "Ana Sayfa" ve "Raporlar" ekranları arasında sekmeli geçiş yapısını kurmak için kullanılmıştır.
- **AsyncStorage:** Kullanıcının odaklanma oturumlarını, sürelerini ve kategori bilgilerini cihazın yerel hafızasında kalıcı olarak (persistent) saklamak için kullanılmıştır.
- **React Native Chart Kit & SVG:** Kaydedilen verilerin anlamlı grafiklere (Çubuk ve Pasta Grafik) dönüştürülerek görselleştirilmesi sağlanmıştır.
- **AppState API:** Uygulamanın ön plan (active) ve arka plan (background) durumlarını dinlemek için kullanılmıştır. Kullanıcı odaklanma süresince uygulamadan ayrıldığında, bu API sayesinde durum değişikliği algılanır ve sayaç otomatik olarak durdurulur.
- **React Hooks:** Durum yönetimi (useState), yan etkilerin yönetimi (useEffect) ve referans takibi (useRef) gibi işlemler modern React kancaları ile gerçekleştirilmiştir.

3. Uygulama Modülleri ve İş Akışı

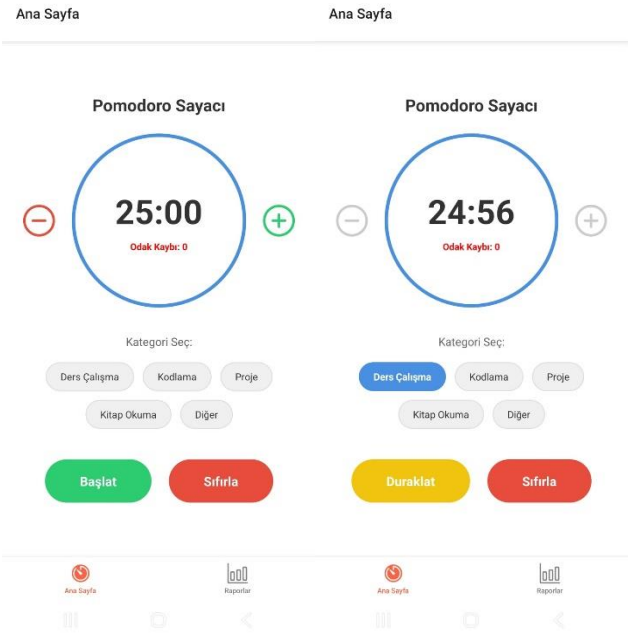
Uygulama, temel işlevlerini yerine getirmek üzere birbirine entegre edilmiş dört ana modülden oluşmaktadır. Her bir modül, kullanıcı deneyimini ve veri bütünlüğünü sağlamak adına belirli bir iş akışını takip eder.

3.1. **Navigasyon Modülü** Kullanıcı uygulamayı başlattığında App.js üzerinden TabNavigator bileşeni devreye girer. Uygulama, kullanıcı dostu bir deneyim sunmak adına "Ana Sayfa" ve "Raporlar" olmak üzere iki temel sekmeye ayrılmıştır. Bu yapı, kullanıcıların odaklanma aracı ile analiz ekranı arasında hızlıca geçiş yapabilmesine olanak tanır.

3.2. **Ana Sayfa ve Sayaç Yönetimi (Home Module)**

Bu modül, kullanıcının odaklanma oturumunu başlattığı ve yönettiği merkezdir. İş akışı şu şekildedir:

- **Kategori Seçimi:** Kullanıcı, oturuma başlamadan önce CategorySelector bileşeni aracılığıyla çalışacağı alanı (Kodlama, Ders, Kitap vb.) belirler.
- **Süre Ayarı:** Varsayılan olarak 25 dakika gelen sayaç, kullanıcı tarafından artırma/azaltma butonları ile kişiselleştirilebilir.
- **Görselleştirme:** Kalan süre ve o anki odak kaybı sayısı, özel olarak tasarlanmış TimerCircle bileşeni içerisinde dinamik olarak gösterilir.
- **Oturum Yönetimi:** Kullanıcı "Başlat", "Duraklat" ve "Sıfırla" butonları ile seansın durumunu kontrol eder.



3.3. **Dikkat Dağınıklığı Takibi (Core Logic)**

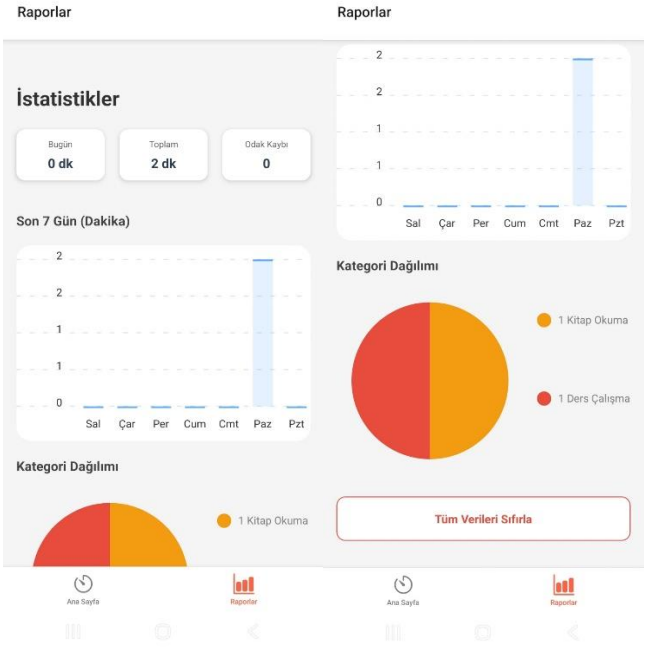
Projenin en ayırt edici özelliği olan bu modül, arka planda çalışan bir AppState dinleyicisi (listener) kullanır. İş akışı şu mantıkla çalışır:

- Sayaç aktif durumdayken (running), sistem telefonun durumunu sürekli izler.
- Kullanıcı uygulamadan çıkarsa (örn: Ana ekrana dönerse veya başka bir uygulamaya geçerse), durum active'den background'a düşer.
- Bu değişim algılandığında algoritma devreye girer: Sayaç otomatik olarak **duraklatılır** ve dikkat dağınıklığı sayacı (distractionCount) **bir artırılır**.
- Kullanıcıya "Dikkat Dağınıklığı" uyarısı verilerek odaklanma sürecine geri dönmesi teşvik edilir.

3.4. **Veri Kaydı ve Raporlama Modülü**

Oturum tamamlandığında veya kullanıcı tarafından sonlandırıldığında, elde edilen veriler (Süre, Kategori, Tarih, Odak Kaybı Sayısı) bir JSON objesi olarak paketlenir ve AsyncStorage modülü ile cihaz hafızasına asenkron olarak kaydedilir. Raporlar ekranı (ReportsScreen) açıldığında:

- Veritabanından tüm geçmiş seanslar çekilir.
- **Günlük ve Toplam İstatistikler:** Hesaplamalar yapılarak özet kartlarda (StatCard) gösterilir.
- **Grafiksel Analiz:** Veriler işlenerek "Son 7 Günlük Odaklanma Süreleri" (Bar Chart) ve "Kategori Dağılımı" (Pie Chart) grafiklerine dökülür.



4. Karşılaşılan Zorluklar ve Çözümleri

Uygulama geliştirme sürecinde hem React Native'in asenkron yapısından hem de mobil cihazların çalışma prensiplerinden kaynaklanan çeşitli teknik zorluklarla karşılaşılmış ve şu çözümler üretilmiştir:

4.1. **Sayaç ve Durum Yönetimi (State Management) Senkronizasyonu:**
- **Sorun:** Geliştirme aşamasında, sayaç sıfıra ulaştığında (00:00) kaydetme fonksiyonunun tetiklenmesi sırasında sonsuz döngü (infinite loop) ve veritabanına mükerrer (çift) kayıt atma sorunu yaşanmıştır. React'in useEffect kancasının bağımlılıkları (dependencies) değiştikçe fonksiyonun tekrar çalışması bu duruma yol açmıştır.
- **Çözüm:** Sayaç mantığına isActive adında bir kontrol değişkeni (flag) eklenmiştir. Süre bittiğinde bu değişken anında false durumuna çekilerek sayacın durması sağlanmış ve finishSession fonksiyonunun sadece bir kez çalışması garanti altına alınmıştır.

4.2. **Arka Plan Takibi ve AppState Yönetimi:**
- **Sorun:** Uygulama arka plana atıldığında (Background state) zamanlayıcının nasıl davranacağı ve bu durumun bir "dikkat dağınıklığı" olarak nasıl ayırt edileceği karmaşık bir yapı gerektirmiştir.
- **Çözüm:** AppState API kullanılarak bir olay dinleyicisi (event listener) oluşturulmuştur. Bu dinleyici, sadece sayaç aktifken (isActive: true) ve uygulama durumu active'den background'a geçtiğinde tetiklenecek şekilde filtrelenmiştir. Böylece kullanıcı sayaç çalışmıyorken uygulamadan çıktığında gereksiz veri işlenmesi engellenmiştir.

4.3. **Kodun Modülerliği ve Temiz Kod Yapısı:**
- **Sorun:** Projenin ilk aşamalarında tüm kodların HomeScreen.js içerisinde toplanması, kodun okunabilirliğini düşürmüş ve yönetilmesini zorlaştırmıştır.
- **Çözüm:** "Temiz Kod" (Clean Code) prensipleri gereği UI parçaları ayrıştırılmıştır. Tekrar eden yapılar (İstatistik kartları, sayaç dairesi, kategori butonları) src/components klasörü altında bağımsız bileşenlere dönüştürülerek ana ekranın yükü hafifletilmiştir.

4.4. **Veri Görselleştirme ve Ekran Uyumluluğu:**
- **Sorun:** Farklı ekran boyutlarına sahip cihazlarda, özellikle sayaç dairesinin ve grafiklerin taşma yapması veya orantısız görünmesi sorunu yaşanmıştır.
- **Çözüm:** Sabit piksel değerleri yerine esnek (flex) tasarım yapısı kullanılmış ve grafik kütüphanesinin (Chart Kit) boyutlandırma parametreleri Dimensions API kullanılarak dinamik hale getirilmiştir.

5. Sonuç

Bu proje kapsamında, güncel bir problem olan dijital dikkat dağınıklığına çözüm üretmek amacıyla React Native tabanlı bir mobil uygulama geliştirilmiştir. Proje süresince, mobil uygulama geliştirme yaşam döngüsüne uygun olarak analiz, tasarım, geliştirme ve test aşamaları gerçekleştirilmiştir.

Uygulama, ödev gereksinimlerinde belirtilen "Minimum Viable Product" (MVP) şartlarının tamamını sağlamaktadır. Ayarlanabilir sayaç, arka plan takibi (AppState), yerel veri kaydı ve detaylı grafiksel raporlama özellikleri başarıyla entegre edilmiştir. Geliştirilen modüler mimari sayesinde, proje ileride yeni özelliklerin (örneğin bulut tabanlı senkronizasyon veya sosyal özellikler) eklenmesine açık, esnek bir yapıdadır.

Sonuç olarak; Mobil Uygulama Geliştirme dersi kapsamında hedeflenen kazanımlar elde edilmiş ve kullanıcıların odaklanma sürelerini takip ederek verimliliklerini artırabileceği, çalışan ve kararlı bir ürün ortaya konmuştur.