

SAKARYA ÜNİVERSİTESİ  
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ  
BÖLÜMÜ



Öğrenci Adı-soyadı: Burak COŞKUN

Öğrenci Numarası: b231210351

Github: <https://github.com/coskunburak/mobilodev>

## Projenin Amacı

Günümüzün en büyük zorluklarından biri olan dijital dikkat dağınıklığıyla mücadele etmek için tasarlanmış bir mobil uygulama geliştirmeniz beklenmektedir. Kullanıcının kendi başlattığı odaklanma seanslarını takip etmeye odaklanır.

Uygulamanın temel amacı; bir zamanlayıcı (Pomodoro tekniği gibi) başlatmak, bu süre boyunca kullanıcının dikkatini dağıtıp dağıtmadığını (uygulamadan çıkıp çıkmadığını) takip etmek ve bu verileri depolayarak kullanıcıya anlamlı raporlar sunmaktır.

### 1. Zamanlayıcı Ekranı ve Modülü;

Zamanlayıcı ekranı, kullanıcının bir “odaklanma seansı” başlatmasını, seans süresince geri sayımın çalışmasını ve seans sırasında uygulamadan çıkılması durumunda **dikkat dağınıklığı** sayımını yaparak seansı **otomatik duraklatmasını** sağlar.

Seans bitiminde veya durduğunda seansa ilişkin özet bilgiler kullanıcıya gösterilir ve oturum verisi kalıcı olarak saklanır.

Bu ekranda kullandığım yapı ve bileşenler;

- **CategoryPicker (components/CategoryPicker.js):** Seans başlamadan önce kategori seçimi için Picker bileşeni. Varsayılan kategoriler: *Ders Çalışma, Kodlama, Proje, Kitap Okuma.*
- **TimerControls (components/TimerControls.js):** Başlat / Duraklat / Sıfırla butonları. Çalışma durumuna göre Başlat–Duraklat koşullu görünür.
- **SessionSummaryModal (components/SessionSummaryModal.js):** Seans tamamlandığında (veya bitim event’i oluştuğunda) seans özetini modal ile gösterir.
- **SessionProvider (context/SessionProvider.js):** Seans listesini uygulama genelinde yönetir ve yeni seansı addSession() ile ekler.
- **AsyncStorage Katmanı (storage/sessionStorage.js):** Seansların kalıcı olarak cihazda saklanmasını sağlar.

### Süre Yönetimi ve Geri Sayım Mekanizması

Zamanlayıcı ekranında süre, kullanıcıdan dakika cinsinden alınır ve güvenli aralığa çekilerek (min 1 – max 180) saniyeye çevrilir.

- Varsayılan süre: **25 dk** (DEFAULT\_MINUTES)
- Kullanıcı girişi TextInput ile alınır; sadece rakamlara izin verilir.
- Geri sayım, her “tick”te kalan süreyi hesaplamak için **hedef bitiş zaman damgası (targetEndTsRef)** yaklaşımıyla çalışır:
  - Başlatılırken  $targetEndTs = now + remainingSec$
  - Interval 250ms çalışır, kalan süreyi  $targetEndTs - now$  üzerinden hesaplar
  - Kalan süre 0 olduğunda seans bitiş akışı tetiklenir

Bu yaklaşım, klasik “her saniye -1” mantığına göre daha stabil olup küçük zaman sapmalarını azaltır.

## Dikkat Dağınıklığı Takibi (AppState)

Seans aktifken uygulama durum değişiklikleri **React Native AppState API** ile dinlenir:

- AppState.addListener("change", ...) ile durum izlenir.
- Uygulama **active** → **background/inactive** geçtiğinde ve sayaç çalışıyorsa:
  - Bu durum **1 adet dikkat dağınıklığı** olarak sayılır (distractionCount + 1)
  - Sayaç **otomatik duraklatılır** (pauseTimer(true))

Bu sayede kullanıcı odak seansı sırasında uygulamadan ayrıldığında sistem bunu “dikkat dağınıklığı” olarak kayıt altına alır ve zamanlayıcıyı otomatik olarak durdurur. Kullanıcı uygulamaya geri döndüğünde sayaç duraklatılmış şekilde kalır ve kullanıcı isterse “Başlat” ile devam edebilir.

## Seansın Kaydı ve Seans Özeti

Seans tamamlandığında bir “session” nesnesi oluşturulur ve uygulama geneline eklenir:

Oluşturulan seans alanları:

- id: zaman damgası tabanlı benzersiz id
- category: seçilen kategori
- targetDurationSec: hedef süre (dk → sn)
- actualDurationSec: gerçek süre (başlangıç–bitiş farkı)
- distractionCount: seans boyunca dikkat dağınıklığı adedi
- startedAt, endedAt: ISO formatında zaman bilgileri

Seans eklendikten sonra **SessionSummaryModal** açılır ve kullanıcıya:

- Kategori
- Hedef süre (dk)
- Gerçek süre (dk)
- Dikkat dağınıklığı sayısı  
bilgileri gösterilir.

## Veri Saklama (AsyncStorage)

Seans verileri cihazda kalıcı olarak saklanır:

- FOCUS\_SESSIONS anahtarıyla JSON olarak yazılır/okunur.
- Uygulama açıldığında loadSessionsFromStorage() ile geçmiş seanslar yüklenir.
- sessions state'i değiştikçe otomatik olarak saveSessionsToStorage(sessions) ile güncellenir.

Bu sayede rapor ekranı, verisini Timer ekranında oluşturulan seans kayıtlarından okuyarak istatistik ve grafik üretebilir.

## 2. Raporlar Ekranı ve Modülü;

Raporlar ekranı, Timer ekranında kaydedilen odaklanma seanslarını okuyarak kullanıcının odaklanma performansını **istatistikler ve grafikler** üzerinden görünür kılar. Kullanıcı belirli bir zaman aralığı seçerek (Bugün / 7 Gün / 30 Gün / Tümü) raporları filtreleyebilir ve seçili aralığa göre toplam süre, ortalama süre, dikkat dağınıklığı metrikleri ile bar/pie grafiklerini görüntüler.

### Veri Kaynağı ve Okuma Mantığı

Bu ekranın veri kaynağı, uygulama genelinde yönetilen SessionProvider içindeki sessions listesidir:

- useSessions() hook'u ile sessions state'i okunur.
- sessions listesi, Timer ekranında seans tamamlandığında eklenen kayıtların bütünüdür.
- Verilerin kalıcılığı AsyncStorage ile sağlandığı için uygulama kapanıp açılrsa bile geçmiş seanslar kaybolmaz.

### Zaman Aralığı Seçimi ve Filtreleme

Ekranı RANGE\_OPTIONS üzerinden 4 farklı filtreleme seçeneği sunulur:

- **Bugün** (today)
- **7 Gün** (7d) – varsayılan seçim
- **30 Gün** (30d)
- **Tümü** (all)

Seçim, range state'i ile tutulur ve filtreleme useMemo ile performanslı şekilde hesaplanır:

- all seçilirse filtre uygulanmadan tüm seanslar döner.
- today seçilirse isSameDay() ile yalnızca bugüne ait seanslar alınır.
- 7d ve 30d seçeneklerinde:
  - Gün başlangıcı (00:00) baz alınır
  - Seçilen gün sayısına göre “from” tarihi hesaplanır
  - Seans startedAt alanına göre ilgili gün aralığında kalanlar filtrelendir

Bu yapı sayesinde aynı seans verisi, farklı zaman ölçeklerinde analiz edilebilir.

### Özet İstatistik Kartları (StatsCard)

Rapor ekranında üst bölümde **2x2 grid** halinde dört adet özet kart gösterilir. Kart yapısı StatsCard bileşeni ile standartlaştırılmıştır (başlık + değer + opsiyonel alt metin).

## Hesaplanan metrikler (seçili aralığa göre):

- Seçili Aralık Toplam Süre**
  - filteredSessions içindeki actualDurationSec toplamı
  - Oturum sayısı da alt bilgi olarak gösterilir
- Ortalama Oturum Süresi**
  - toplam süre / oturum sayısı (0'a bölmeyi engelleyen kontrol var)
- Dikkat Dağılıklığı**
  - filteredSessions içindeki distractionCount toplamı
  - Oturum başına ortalama dikkat dağılıklığı hesaplanır
- Bugün / Tüm Zamanlar Karşılaştırması**
  - Bugünün toplam süresi: isSameDay() ile bugünkü seanslar üzerinden
  - Tüm zamanların toplamı: tüm sessions üzerinden
  - Tek kart içinde karşılaştırmalı gösterim yapılır

Süreler kullanıcıya daha anlaşılır olması için dk / saat-dk formatına dönüştürülür:

- formatMinutes(sec) → “xx dk”
- formatMinutesWithHours(sec) → “x sa y dk” (60 dk üstünde)

## Bar Chart: Son N Gün Odaklanma Süreleri

Seçilen zaman aralığına göre bar chart gün sayısı dinamik değişir:

- today → **1 gün**
- 7d → **7 gün**
- 30d → **30 gün**

Grafik verisi, “son N gün”ün her günü için şu mantıkla üretilir:

- Her gün için 00:00 – 23:59:59 aralığı belirlenir
- O günün içine düşen seansların actualDurationSec değerleri toplanır
- Grafik değerleri **dakika** cinsinden oluşturulur

Etiket formatı okunabilirlik için gün/\n\tnay şeklindedir (örn. 30/\n11).

**UI/UX detayı:** 30 günlük grafikte sıkışmayı azaltmak için grafik genişliği dinamik hesaplanır:

- chartWidth = max(screenWidth-32, chartDays\*28)
- Böylece 30 günlük görünümde yatay kaydırma ile barlar okunabilir kalır.

Veri yoksa (tüm değerler 0 ise) kullanıcıya bilgi mesajı gösterilir:

- “Son N gün için veri bulunmuyor.”

## Pie Chart: Kategorilere Göre Dağılım

Pie chart, seçili aralığa göre filteredSessions üzerinden üretilir:

- Her kategori için toplam odaklanma süresi (dakika) hesaplanır:
  - `map[category] += actualDurationSec/60`
- Sonuçlar react-native-chart-kit PieChart formatına dönüştürülür:
  - name (kategori adı)
  - population (toplam dakika)
  - color (index'e göre renk)
  - legendFontColor, legendFontSize

Seçili aralıkta seans yoksa kullanıcıya:

- “Seçili aralık için kategori verisi yok.” mesajı gösterilir.

## Boş Durum Yönetimi (Empty State)

Uygulamada hiç seans yoksa (`sessions.length === 0`) rapor ekranında kullanıcıya açıklayıcı bir metin gösterilir:

- “Henüz kayıtlı seans yok... Zamanlayıcıyı kullanmaya başladığında istatistikler burada görünecek.”

Bu sayede kullanıcı ilk kullanımda boş ekran yerine yönlendirici bilgi alır.

## Genel Sistem Mimarisi

### Mimari Yaklaşım ve Katmanlar

Uygulama, odaklanma seanslarının oluşturulması, dikkat dağınıklığının izlenmesi, verinin kalıcı olarak saklanması ve raporlanması süreçlerini ayrı sorumluluklarla yöneten katmanlı bir yapı ile geliştirilmiştir. Mimari; **UI Katmanı**, **Durum Yönetimi (Context) Katmanı**, **Kalıcı Depolama Katmanı** ve **Yardımcı/Ortak Modüller** şeklinde kurgulanmıştır.

### UI Katmanı (Screens & Components)

#### Ekranlar

- **TimerScreen (screens/TimerScreen.js)**  
Odaklanma seansının başlatıldığı ana ekrandır. Süre/kategori seçimi, geri sayım, AppState ile dikkat dağınıklığı takibi, seansın oluşturulması ve özet modalının gösterilmesi bu ekranda yürütülür.
- **ReportsScreen (screens/ReportsScreen.js)**  
Kayıtlı seansların okunarak filtrelenebilir, istatistiklerin hesaplanması ve grafiklerle sunulması bu ekranda yapılır.

## Yeniden Kullanılabilir Bileşenler

- **CategoryPicker**: Seans kategorisi seçimi
- **TimerControls**: Başlat/Duraklat/Sıfırla kontrol butonları
- **SessionSummaryModal**: Seans bitiminde özet gösterimi
- **StatsCard**: Dashboard metrik kartları

UI katmanında amaç, görsel sunum ile etkileşim mantığını ayırtmak ve bileşenleri yeniden kullanılabilir hale getirmektir.

## Durum Yönetimi Katmanı (Context)

Uygulama genelinde seans verisinin tek bir merkezden yönetilmesi için **React Context** yapısı kullanılmıştır:

- **SessionProvider (context/SessionProvider.js)**
  - sessions: Tüm seans kayıtlarını tutan global state
  - addSession(session): Yeni seans listeye ekleyen fonksiyon

## Veri Akışı

- **TimerScreen**, seans tamamlandığında bir session nesnesi üretir ve addSession() ile global listeye ekler.
- **ReportsScreen**, sessions listesini okuyarak filtreleme ve istatistik/grafik üretimi yapar.

Bu yaklaşım, ekranlar arası veri paylaşımını basitleştirir ve tek doğruluk kaynağı (single source of truth) sağlar.

## Kalıcı Depolama Katmanı (AsyncStorage)

Seans verilerinin uygulama kapatılsa bile saklanması için **AsyncStorage** kullanılmıştır:

- **sessionStorage (storage/sessionStorage.js)**
  - FOCUS\_SESSIONS anahtarı ile JSON formatında kayıt yapılır.
  - loadSessionsFromStorage(): Uygulama açılışında geçmiş seansları okur.
  - saveSessionsToStorage(sessions): Seans listesi değiştikçe güncel veriyi yazar.

## Başlatma ve Senkronizasyon

- SessionProvider ilk render'da loadSessionsFromStorage() çağırarak state'i doldurur.
- sessions her değiştiğinde saveSessionsToStorage(sessions) ile otomatik persist edilir.

## İş Kuralları ve Takip Mekanizması (Timer Mantığı)

Timer modülü, seansın güvenilir şekilde ölçülmesi ve dikkat dağınıklığının doğru yakalanması için aşağıdaki mekanizmaları kullanır:

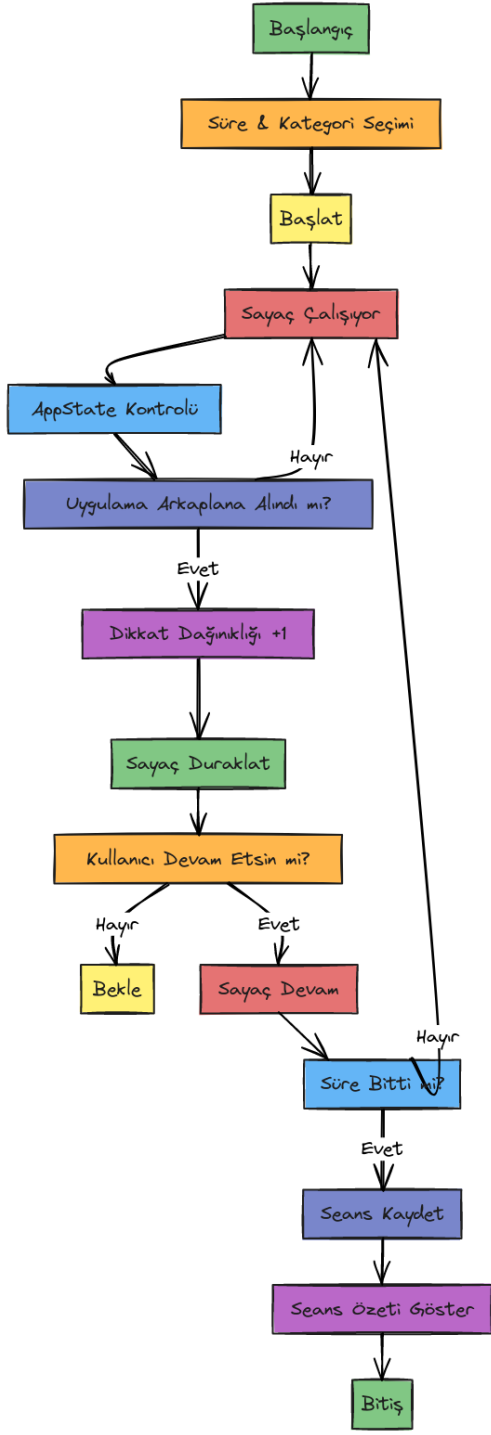
- **Geri sayım**: hedef bitiş zaman damgası (targetEndTsRef) üzerinden kalan süre hesaplanır.
- **AppState dinleme**: sayaç çalışırken uygulama background/inactive olursa:

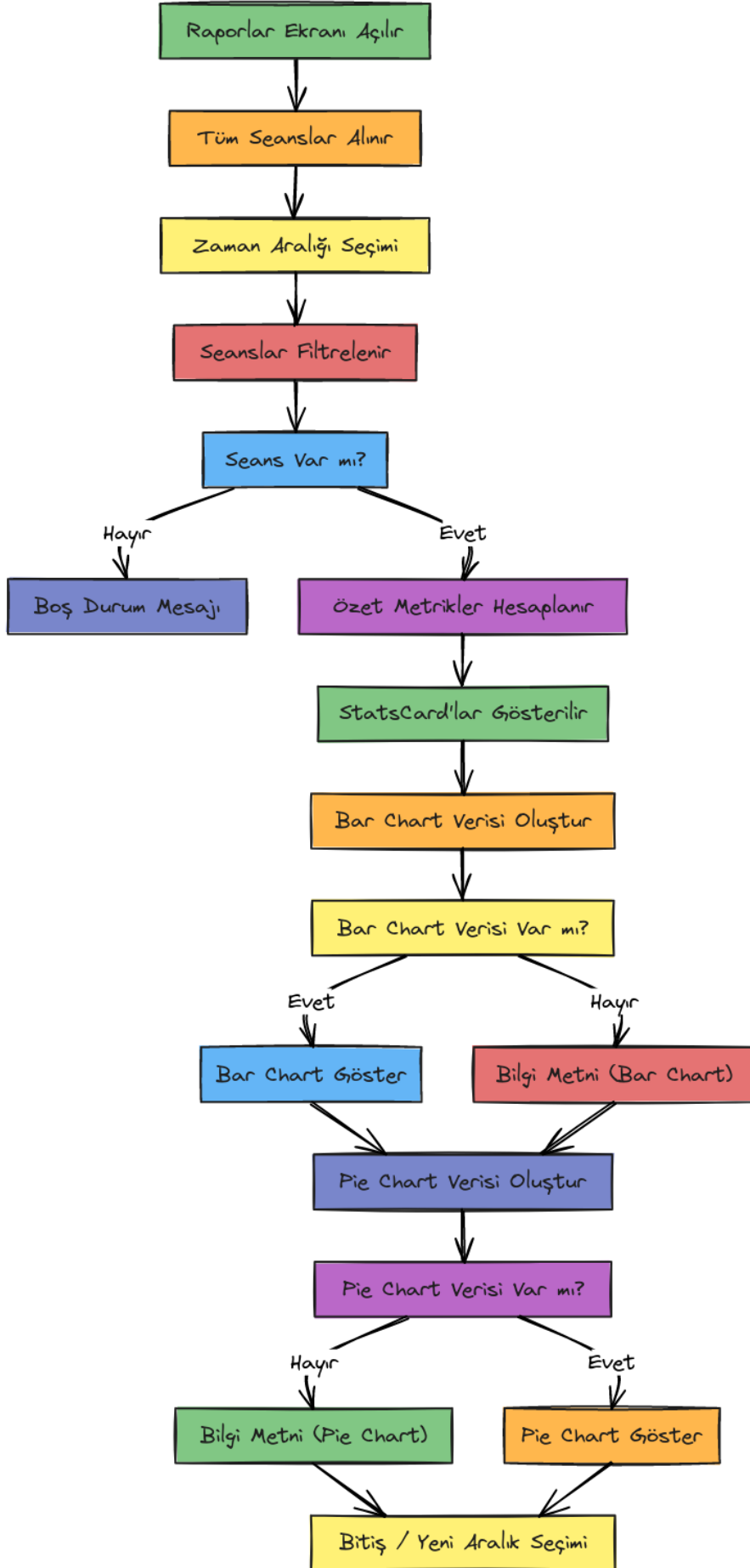
- distractionCount artırılır
  - sayaç otomatik duraklatılır
- **Seans nesnesi üretimi:** category, hedef/gerçek süre, dikkat dağınıklığı ve zaman damgaları ile kayıt hazırlanır.

### 3.6. Yardımcı Modüller

- **dateUtils (utils/dateUtils.js)**
  - isSameDay(a,b): raporlama tarafında gün bazlı filtreleme ve “bugün” hesaplamaları için kullanılır.
  - getLastNDaysLabels(n): gün etiket üretimi için yardımcı fonksiyon (grafik etiketleme ihtiyacına yönelik).
- **theme/colors (theme/colors.js)**
  - Uygulama genelinde tutarlı görünüm için ortak renk paleti.







15:51



## Odaklanma Zamanlayıcısı

Süre (dakika)

25

dk

25:00

Kategori

Ders Çalışma

Kodlama

Proje

Dikkat Dağınıklığı: 0

Başlat

Sıfırla

▼  
Timer

▼  
Reports

15:51

50% 50%

## Raporlar

7 Gün

Odaklanma istatistiklerin ve kategori dağılımın

Bugün

7 Gün

30 Gün

Tümü

Seçili Aralık Toplam

**5 dk**

Oturum sayısı: 5

Ortalama Oturum Süresi

**1 dk**

Seçili aralıktaki oturumlar

Dikkat Dağınıklığı

**2 kez**

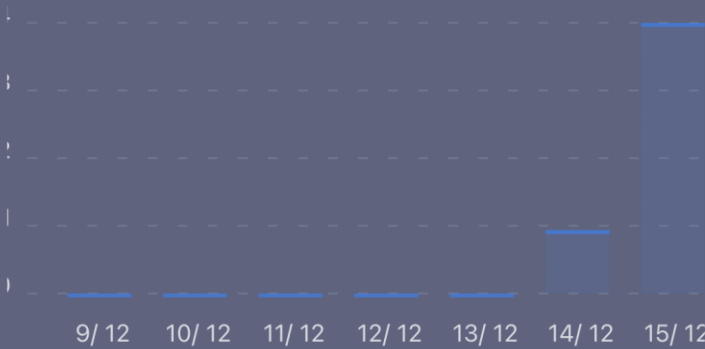
Oturum başına ort.: 0.40

Bugün / Tüm Zamanlar

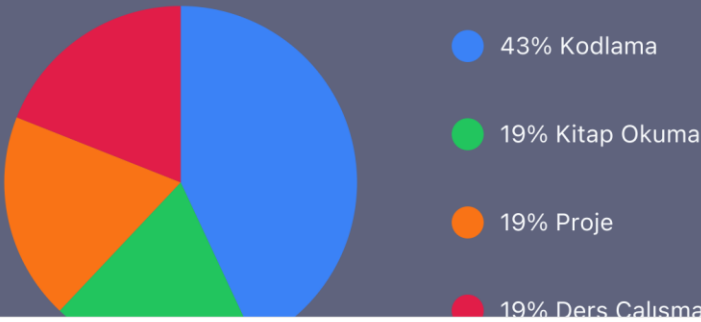
**4 dk • 5 dk**

Bugün / tüm zamanlar  
karşılaştırma

### Son 7 Gün Odaklanma Süresi (dk)



### Kategorilere Göre Dağılım



Timer

Reports