2025-07-02 16:34

Status:

Tags: # GoonScroll - Systemanalyse und Testkonzept (Aktualisiert nach Repository-Review)

1. Schichtentrennung

A. Paketstruktur und Modularisierung

Aktuelle Repository-Struktur:

```
├── App.js               # Hauptkomponente mit Onboarding
├── screens/             # Bildschirme
│   ├── HomeScreen.js       # Hauptbildschirm mit WebViews
│   ├── PowerModeScreen.js  # Landscape-Modus für 3 Streams
│   ├── AnalyticsScreen.js  # Analytics Dashboard
│   └── TrackingService.js  # Business Logic für Analytics
├── components/          # UI-Komponenten (teilweise leer)
│   ├── LoginModal.js       # (leer)
│   └── VideoCard.js        # (leer)
├── navigation/          # Navigation (leer)
│   └── AppNavigator.js     # (leer)
└── index.js            # Entry Point
```

Bewertung der Schichtentrennung:

- □ **Trennung erreicht**: Screens und Services sind getrennt
- □ **Verbesserungsbedarf**: TrackingService ist im screens/ Ordner statt services/
- □ **Incomplete**: Viele Komponenten-Dateien sind leer
- □ **Keine expliziten Modelle**: Datenstrukturen sind inline definiert

B. Separate Modellbeschreibung

Aktuelle Implementation:

- □ **Keine separaten Modell-Dateien**: Datenstrukturen sind direkt in Components definiert
- □ **Implizite Modelle vorhanden**: TrackingService definiert Analytics-Datenstrukturen
- □ **Platform-Konfiguration**: Wird in jedem Screen separat definiert

Empfohlene Verbesserung:

```
// models/Platform.js
export const Platform = {
  YOUTUBE: {
    id: 'youtube',
    name: 'YouTube',
    color: '#FF0000',
    url: 'https://m.youtube.com/playlist?list=PLrAXtmRdnEQy8VtkaWvaJnCMjj_
    icon: '
  },
  TIKTOK: {
    id: 'tiktok',
    name: 'TikTok',
    color: '#000000',
    url: 'https://www.tiktok.com/foryou',
    icon: '
  },
  INSTAGRAM: {
    id: 'instagram',
    name: 'Instagram',
    color: '#E4405F',
    url: 'https://www.instagram.com/reels/',
    icon: '
  }
};

// models/AnalyticsData.js
export class AnalyticsSession {
  constructor(platform, startTime, endTime, duration, date, isPowerMode =
    this.platform = platform;
    this.startTime = startTime;
    this.endTime = endTime;
    this.duration = duration;
    this.date = date;
    this.isPowerMode = isPowerMode;
  }
}
```

C. UI ohne Business Logic

Aktuelle Implementation:

- ✅ **Gut getrennt**: HomeScreen delegiert Tracking an TrackingService
- ✅ **Clean Components**: AnalyticsScreen rendert nur, berechnet nichts
- ⚠️ **Kleine Vermischung**: Orientation-Handling in PowerModeScreen

Beispiel für gute Trennung aus dem Code:

```
// HomeScreen.js - UI delegiert an Service
useEffect(() => {
    TrackingService.startSession(activeTab);
}, []);

// TrackingService.js - Pure Business Logic
async saveSession(sessionData) {
    // Komplexe Berechnungen und Datenspeicherung
}
```

2. Systemaufbau - Klassendiagramm (Aktualisiert)

```
classDiagram
    class App {
        +OnboardingScreen
        +NavigationContainer
        +render()
    }
```

```
class HomeScreen {
    +activeTab: string
    +isLoading: boolean
    +webViewRefs: object
    +handleTabSwitch()
    +onSwipeGesture()
    +handleShare()
    +render()
}

class PowerModeScreen {
    +orientation: string
    +handleExit()
    +render()
}

class AnalyticsScreen {
    +selectedPeriod: string
    +analyticsData: object
    +loadAnalytics()
    +render()
}

class TrackingService {
    +isTracking: boolean
    +currentSession: object
    +startSession(platform): void
    +endSession(): void
    +switchPlatform(platform): void
    +saveSession(data): void
    +trackShare(platform): void
    +getUsageStats(): object
    +formatTime(seconds): string
    +startPowerModeSession(): void
    +endPowerModeSession(): void
}

class SimpleStorage {
    +getItem(key): Promise
    +setItem(key, value): Promise
    +getAllKeys(): Promise
    +multiRemove(keys): Promise
}

class PlatformConfig {
    +youtube: object
    +tiktok: object
    +instagram: object
}

class WebView {
    +source: object
    +onLoadStart(): void
    +onLoadEnd(): void
    +render()
}

App --> HomeScreen
App --> PowerModeScreen
App --> AnalyticsScreen

HomeScreen --> TrackingService
HomeScreen --> WebView
HomeScreen --> PlatformConfig

PowerModeScreen --> WebView
PowerModeScreen --> PlatformConfig

AnalyticsScreen --> TrackingService

TrackingService --> SimpleStorage
TrackingService ..> PlatformConfig

WebView --> PlatformConfig
```

Beziehungen der aktuellen Implementation:

- **Komposition**: App komponiert verschiedene Screens über React Navigation
- **Verwendung**: Screens nutzen TrackingService für Business Logic
- **Abhängigkeit**: TrackingService nutzt SimpleStorage für Persistierung
- **Konfiguration**: Platform-Definitionen werden in jedem Screen dupliziert (Verbesserungsbedarf)

3. Testkonzept (Angepasst an aktuelle Implementation)

A. Testumfeld

Hardware:

- iOS: iPhone 12/13/14 (iOS 15+)
- Android: Samsung Galaxy S21, Google Pixel 6 (Android 11+)

Software:

- React Native 0.72+ mit Expo
- Jest für Unit Tests
- Detox für E2E Tests
- React Native Testing Library

Externe Abhängigkeiten:

- WebView-basierte Plattform-Integration
- Screen Orientation API
- AsyncStorage/Simple Storage
- React Navigation
- PanGestureHandler

Netzwerk:

- WiFi: 50 Mbps+
- 4G/5G Mobilfunk
- Offline-Szenarios für Storage-Tests

B. Testmethode

1. Unit Tests (Jest)

- TrackingService Methoden isoliert testen
- SimpleStorage Mock-Tests
- Utility-Funktionen (formatTime, getDateString)

2. Component Tests (React Native Testing Library)

- Screen-Rendering ohne Business Logic
- User Interaction Events
- Navigation zwischen Screens

3. Integration Tests

- TrackingService mit SimpleStorage
- WebView Integration
- Platform-Switching Flow

4. End-to-End Tests (Detox)

- Komplette User Journeys
- Orientation Changes
- Cross-Platform WebView Funktionalität

5. Performance Tests

- Memory Usage in PowerMode
- WebView Performance
- Battery Impact Assessment

C. Testfälle (Basierend auf aktueller Implementation)

Testfall 1: Platform Switching mit Tracking

Identifikation: TC001_PlatformSwitchTracking **Vorbedingungen:**

- App ist gestartet (Onboarding abgeschlossen)
- TrackingService ist initialisiert
- Alle drei Platform-URLs sind erreichbar

Schritte:

1. Öffne HomeScreen
2. Verifiziere Start auf YouTube-Tab
3. Warte 10 Sekunden
4. Wechsle zu TikTok-Tab
5. Verifiziere TrackingService.switchPlatform() Aufruf
6. Warte 15 Sekunden
7. Wechsle zu Instagram-Tab
8. Prüfe Analytics-Daten
9. Öffne AnalyticsScreen
10. Verifiziere korrekte Zeiterfassung

Erwartetes Resultat:

- YouTube: ~10 Sekunden erfasst
- TikTok: ~15 Sekunden erfasst
- Instagram: Aktive Session läuft
- Daten werden in SimpleStorage persistiert

Testfall 2: PowerMode mit drei parallelen Streams

Identifikation: TC002_PowerModeOperation **Vorbedingungen:**

- App ist im HomeScreen
- Gerät ist im Portrait-Modus
- Screen Orientation API ist verfügbar

Schritte:

1. Navigiere zu PowerMode über Header-Button
2. Verifiziere Orientation-Lock zu Landscape
3. Prüfe gleichzeitige Darstellung von 3 WebViews
4. Teste Scroll-Funktionalität in jedem Stream
5. Verifiziere TrackingService.startPowerModeSession()
6. Warte 30 Sekunden
7. Drücke Exit-Button
8. Verifiziere TrackingService.endPowerModeSession()
9. Prüfe Analytics für alle drei Plattformen

Erwartetes Resultat:

- Landscape-Modus aktiviert
- Drei funktionale WebViews nebeneinander
- 30 Sekunden gleichmäßig auf alle Plattformen verteilt (10s pro Platform)
- Smooth Exit zurück zu HomeScreen

Testfall 3: Analytics Data Aggregation

Identifikation: TC003_AnalyticsAggregation **Vorbedingungen:**

- TrackingService hat Session-Daten von verschiedenen Tagen
- SimpleStorage enthält Analytics-Einträge
- AnalyticsScreen ist zugänglich

Schritte:

1. Simuliere mehrere Sessions über mehrere Tage
2. Füge verschiedene Share-Events hinzu
3. Öffne AnalyticsScreen
4. Teste "Heute" vs "Diese Woche" Ansicht
5. Verifiziere Platform-Verteilungs-Balken
6. Prüfe Wöchentliche Aktivitäts-Chart
7. Teste Pull-to-Refresh Funktionalität
8. Verifiziere Insights-Berechnungen

Erwartetes Resultat:

- Korrekte Aggregation von Session-Daten
- Animierte Balken-Charts
- Richtige Prozentverteilung
- Plausible Insights und Statistiken

Testfall 4: Swipe Gesture Navigation

Identifikation: TC004_SwipeNavigation **Vorbedingungen:**

- HomeScreen ist aktiv
- PanGestureHandler ist konfiguriert
- Platform-Reihenfolge: YouTube → TikTok → Instagram

Schritte:

1. Starte auf YouTube-Tab
2. Führe Swipe-left Geste aus
3. Verifiziere Wechsel zu TikTok
4. Führe weitere Swipe-left aus
5. Verifiziere Wechsel zu Instagram
6. Führe Swipe-right aus
7. Verifiziere Wechsel zurück zu TikTok
8. Teste zu schnelle/kurze Swipes (sollten ignoriert werden)
9. Teste Swipe von Instagram nach rechts (wrap-around zu YouTube)

Erwartetes Resultat:

- Responsive Swipe-Erkennung
- Korrekte Platform-Reihenfolge
- Wrap-around Funktionalität
- Filtering von unbeabsichtigten Gesten

Testfall 5: WebView Error Handling

Identifikation: TC005_WebViewErrorHandling **Vorbedingungen:**

- App ist gestartet
- Netzwerkverbindung kann kontrolliert werden

Schritte:

1. Starte App mit aktiver Internetverbindung
2. Wechsle zu einer Plattform
3. Deaktiviere Internetverbindung während des Ladens
4. Verifiziere Error-Dialog
5. Drücke "Wiederholen" Button
6. Reaktiviere Internetverbindung

7. Verifiziere erfolgreiche Wiederherstellung
8. Teste mit verschiedenen HTTP-Error-Codes (404, 503)

Erwartetes Resultat:

- Benutzerfreundliche Error-Dialoge
- Retry-Funktionalität
- Graceful Degradation bei Netzwerkproblemen
- Keine App-Crashes

Testfall 6: Onboarding Flow

Identifikation: TC006_OnboardingFlow **Vorbedingungen:**

- App wird zum ersten Mal gestartet
- Keine vorherigen Login-Daten vorhanden

Schritte:

1. Öffne App
2. Verifiziere Onboarding-Screen
3. Simuliere "Anmeldung" bei YouTube
4. Prüfe Progress-Bar Update (33%)
5. Simuliere Anmeldung bei TikTok und Instagram
6. Verifiziere Progress-Bar bei 100%
7. Drücke "App starten" Button
8. Verifiziere Navigation zu HomeScreen
9. Teste Zurück-Navigation (sollte blockiert sein)

Erwartetes Resultat:

- Intuitive Onboarding-Experience
- Korrekte Progress-Verfolgung
- Smooth Transition zu HomeScreen
- Login-Status wird korrekt gespeichert

Testmetriken (Angepasst)

Akzeptanzkriterien:

- App-Start: < 3 Sekunden
- Platform-Wechsel: < 2 Sekunden
- PowerMode-Transition: < 1 Sekunde
- WebView Load Time: < 5 Sekunden
- Memory Usage PowerMode: < 500 MB
- Battery Impact: Maximal 15% höher als native Apps
- Tracking Accuracy: 95% korrekte Zeiterfassung
- Crash Rate: < 0.1%

Testabdeckung:

- Unit Tests: > 80% (TrackingService, Utils)
- Component Tests: > 70% (Screen Components)
- Integration Tests: 100% (kritische User Flows)
- E2E Tests: 100% (Hauptfunktionen)

4. Funktionale und nicht-funktionale Anforderungen

A. Funktionale Anforderungen - Use Case Diagramm

```
graph TD
    User((Heavy Social Media User<br/>16-35 Jahre))

    subgraph "GoonScroll System"
        UC1[Plattformen durchnavigieren]
        UC2[Video-Content konsumieren]
        UC3[PowerMode aktivieren]
        UC4[Analytics einsehen]
        UC5[Content teilen]
        UC6[App-Einstellungen verwalten]
        UC7[Onboarding durchlaufen]
        UC8[Nutzungszeit verfolgen]
        UC9[Plattform wechseln via Swipe]
        UC10[Drei Streams parallel nutzen]
    end

    subgraph "Externe Systeme"
        YouTube[YouTube Shorts API]
        TikTok[TikTok Web Interface]
        Instagram[Instagram Reels API]
        ShareAPI[Native Share API]
        OrientationAPI[Screen Orientation API]
    end

    User --> UC1
    User --> UC2
    User --> UC3
    User --> UC4
    User --> UC5
    User --> UC6
    User --> UC7

    UC1 --> UC9
    UC2 --> UC8
    UC3 --> UC10
    UC5 --> ShareAPI
    UC3 --> OrientationAPI

    UC2 --> YouTube
    UC2 --> TikTok
    UC2 --> Instagram

    UC8 -.-> UC4
    UC9 -.-> UC8
    UC10 -.-> UC8
```

B. Akteur-Beschreibungen

Heavy Social Media User (16-35 Jahre): Hauptnutzer der App, der täglich mehrere Stunden auf verschiedenen Video-Plattformen verbringt und eine effizientere, bewusstere Nutzung seiner Zeit wünscht, ohne zwischen mehreren Apps wechseln zu müssen.

YouTube Shorts API: Externe Plattform, die kurze vertikale Videos bereitstellt und über WebView-Integration in die App eingebunden wird.

TikTok Web Interface: Externe Social Media Plattform, die über mobile Web-Schnittstelle Zugang zu kurzen, viralen Video-Inhalten bietet.

Instagram Reels API: Meta-Plattform, die Story-basierte kurze Video-Inhalte über Web-Interface zur Verfügung stellt.

Native Share API: Betriebssystem-Service von iOS/Android, der systemweite Teilen-Funktionalität für App-übergreifende Content-Verteilung ermöglicht.

Screen Orientation API: Expo/React Native Service, der die Bildschirmausrichtung programmatisch steuert und für PowerMode-Funktionalität benötigt wird.

C. Nicht-funktionale Anforderungen nach FURPS

Functionality (Funktionalität)

NF-01 Platform Integration Reliability:

- Messbare Anforderung: 99.5% erfolgreiche WebView-Ladeoperationen bei stabiler Internetverbindung (>10 Mbps)
- Messverfahren: Automated Loading Tests über 1000 Ladevorgänge pro Plattform
- Akzeptanzkriterium: <0.5% Fehlerrate bei Platform-Loads

NF-02 Analytics Data Accuracy:

- Messbare Anforderung: 98% Genauigkeit bei Zeiterfassung (max. 2% Abweichung von tatsächlicher Viewing-Zeit)
- Messverfahren: Vergleich mit manuell gestoppter Zeit über 50 Test-Sessions
- Akzeptanzkriterium: Durchschnittliche Abweichung <2% über alle gemessenen Sessions

Usability (Benutzerfreundlichkeit)

NF-03 Navigation Response Time:

- Messbare Anforderung: Platform-Wechsel erfolgt in <2 Sekunden (95% der Fälle)
- Messverfahren: Automated UI Tests mit Zeitstempel-Messung
- Akzeptanzkriterium: 95% aller Tab-Switches unter 2 Sekunden

NF-04 Gesture Recognition Accuracy:

- Messbare Anforderung: 90% korrekte Swipe-Gesten-Erkennung bei normalem Nutzungsverhalten
- Messverfahren: User Testing mit 20 Probanden, je 50 Swipe-Gesten
- Akzeptanzkriterium: <10% false positives/negatives bei Swipe-Navigation

NF-05 PowerMode Transition Speed:

- Messbare Anforderung: Landscape-zu-PowerMode Übergang in <1 Sekunde
- Messverfahren: Automated Tests mit High-Speed Screen Recording
- Akzeptanzkriterium: Komplette UI-Transition unter 1000ms

Reliability (Zuverlässigkeit)

NF-06 App Stability:

- Messbare Anforderung: Crash-Rate <0.1% bei normaler Nutzung
- Messverfahren: Crash Analytics über 10.000 App-Sessions
- Akzeptanzkriterium: <1 Crash pro 1000 App-Starts

NF-07 Data Persistence Reliability:

- Messbare Anforderung: 99.9% erfolgreiche Analytics-Datenspeicherung
- Messverfahren: Automated Tests mit simulierten App-Kills und Restarts
- Akzeptanzkriterium: <0.1% Datenverlust bei unerwarteten App-Beendigungen

NF-08 Network Error Recovery:

- Messbare Anforderung: 95% erfolgreiche Auto-Recovery nach Netzwerkunterbrechungen <30 Sekunden
- Messverfahren: Automated Tests mit simulierten Netzwerkausfällen
- Akzeptanzkriterium: App funktionsfähig binnen 5 Sekunden nach Netzwerk-Wiederherstellung

Performance (Leistung)

NF-09 App Launch Time:

- Messbare Anforderung: Cold Start <3 Sekunden auf Mittelklasse-Geräten (iPhone 12, Android SDK 30)

- Messverfahren: Automated Performance Tests mit App Launch Metrics
- Akzeptanzkriterium: 90% der Cold Starts unter 3 Sekunden

NF-10 Memory Usage PowerMode:

- Messbare Anforderung: RAM-Verbrauch <500 MB während PowerMode mit drei aktiven WebViews
- Messverfahren: Memory Profiling über 30-Minuten PowerMode Sessions
- Akzeptanzkriterium: Durchschnittlicher Peak Memory <500 MB

NF-11 Battery Efficiency:

- Messbare Anforderung: Max. 15% höherer Batterieverbrauch als native Apps bei vergleichbarer Nutzung
- Messverfahren: 2-Stunden Nutzungstests vs. native YouTube/TikTok/Instagram Apps
- Akzeptanzkriterium: <15% zusätzlicher Battery Drain verglichen mit nativen Apps

Supportability (Unterstützbarkeit)

NF-12 Error Logging Coverage:

- Messbare Anforderung: 100% kritischer Fehler werden mit Stack Trace und Context geloggt
- Messverfahren: Code Review und Error Injection Tests
- Akzeptanzkriterium: Alle Crashes und kritische Fehler haben vollständige Log-Einträge

NF-13 Debug Information Availability:

- Messbare Anforderung: Analytics und Tracking-Status in <3 Taps vom Hauptscreen erreichbar
- Messverfahren: UI Navigation Tests
- Akzeptanzkriterium: Debug/Analytics Info maximal 3 Screen-Navigationen entfernt

NF-14 Platform Compatibility:

- Messbare Anforderung: 100% Funktionalität auf iOS 15+ und Android 11+ (API Level 30+)
- Messverfahren: Device Compatibility Testing auf 10 verschiedenen Geräten
- Akzeptanzkriterium: Alle Core Features funktional auf Minimum OS Versionen

Qualitätssicherungsmaßnahmen

Continuous Integration Requirements:

- Automated Testing: Alle NF-Anforderungen werden in CI/CD Pipeline getestet
- Performance Monitoring: Realtime Tracking von Response Times und Memory Usage
- Crash Reporting: Integration mit Crashlytics/Sentry für Production Monitoring
- User Analytics: Heatmaps und User Journey Tracking für Usability Verification

Acceptance Criteria Matrix:

Anforderung	Messverfahren	Tool	Frequency	Threshold
NF-03	UI Response Tests	Detox	Every Build	<2s
NF-06	Crash Analytics	Crashlytics	Continuous	<0.1%
NF-09	Launch Performance	Flipper	Daily	<3s
NF-10	Memory Profiling	Xcode Instruments	Weekly	<500MB

4. Funktionale und nicht-funktionale Anforderungen

A. Funktionale Anforderungen - Use Case Diagramm

```
graph TD
    User((Heavy Social Media User<br/>16-35 Jahre))

    subgraph "GoonScroll System"
        UC1[Plattformen durchnavigieren]
        UC2[Video-Content konsumieren]
        UC3[PowerMode aktivieren]
        UC4[Analytics einsehen]
        UC5[Content teilen]
        UC6[App-Einstellungen verwalten]
        UC7[Onboarding durchlaufen]
        UC8[Nutzungszeit verfolgen]
        UC9[Plattform wechseln via Swipe]
        UC10[Drei Streams parallel nutzen]
    end

    subgraph "Externe Systeme"
        YouTube[YouTube Shorts API]
        TikTok[TikTok Web Interface]
        Instagram[Instagram Reels API]
        ShareAPI[Native Share API]
        OrientationAPI[Screen Orientation API]
    end

    User --> UC1
    User --> UC2
    User --> UC3
    User --> UC4
    User --> UC5
    User --> UC6
    User --> UC7

    UC1 --> UC9
    UC2 --> UC8
    UC3 --> UC10
    UC5 --> ShareAPI
    UC3 --> OrientationAPI

    UC2 --> YouTube
    UC2 --> TikTok
    UC2 --> Instagram

    UC8 -.-> UC4
    UC9 -.-> UC8
    UC10 -.-> UC8
```

B. Akteur-Beschreibungen

Heavy Social Media User (16-35 Jahre): Hauptnutzer der App, der täglich mehrere Stunden auf verschiedenen Video-Plattformen verbringt und eine effizientere, bewusstere Nutzung seiner Zeit wünscht, ohne zwischen mehreren Apps wechseln zu müssen.

YouTube Shorts API: Externe Plattform, die kurze vertikale Videos bereitstellt und über WebView-Integration in die App eingebunden wird.

TikTok Web Interface: Externe Social Media Plattform, die über mobile Web-Schnittstelle Zugang zu kurzen, viralen Video-Inhalten bietet.

Instagram Reels API: Meta-Plattform, die Story-basierte kurze Video-Inhalte über Web-Interface zur Verfügung stellt.

Native Share API: Betriebssystem-Service von iOS/Android, der systemweite Teilen-Funktionalität für App-übergreifende Content-Verteilung ermöglicht.

Screen Orientation API: Expo/React Native Service, der die Bildschirmausrichtung programmatisch steuert und für PowerMode-Funktionalität benötigt wird.

C. Nicht-funktionale Anforderungen nach FURPS

Functionality (Funktionalität)

NF-01 Platform Integration Reliability:

- Messbare Anforderung: 99.5% erfolgreiche WebView-Ladeoperationen bei stabiler Internetverbindung (>10 Mbps)
- Messverfahren: Automated Loading Tests über 1000 Ladevorgänge pro Plattform
- Akzeptanzkriterium: <0.5% Fehlerrate bei Platform-Loads

NF-02 Analytics Data Accuracy:

- Messbare Anforderung: 98% Genauigkeit bei Zeiterfassung (max. 2% Abweichung von tatsächlicher Viewing-Zeit)
- Messverfahren: Vergleich mit manuell gestoppter Zeit über 50 Test-Sessions
- Akzeptanzkriterium: Durchschnittliche Abweichung <2% über alle gemessenen Sessions

Usability (Benutzerfreundlichkeit)

NF-03 Navigation Response Time:

- Messbare Anforderung: Platform-Wechsel erfolgt in <2 Sekunden (95% der Fälle)
- Messverfahren: Automated UI Tests mit Zeitstempel-Messung
- Akzeptanzkriterium: 95% aller Tab-Switches unter 2 Sekunden

NF-04 Gesture Recognition Accuracy:

- Messbare Anforderung: 90% korrekte Swipe-Gesten-Erkennung bei normalem Nutzungsverhalten
- Messverfahren: User Testing mit 20 Probanden, je 50 Swipe-Gesten
- Akzeptanzkriterium: <10% false positives/negatives bei Swipe-Navigation

NF-05 PowerMode Transition Speed:

- Messbare Anforderung: Landscape-zu-PowerMode Übergang in <1 Sekunde
- Messverfahren: Automated Tests mit High-Speed Screen Recording
- Akzeptanzkriterium: Komplette UI-Transition unter 1000ms

Reliability (Zuverlässigkeit)

NF-06 App Stability:

- Messbare Anforderung: Crash-Rate <0.1% bei normaler Nutzung
- Messverfahren: Crash Analytics über 10.000 App-Sessions
- Akzeptanzkriterium: <1 Crash pro 1000 App-Starts

NF-07 Data Persistence Reliability:

- Messbare Anforderung: 99.9% erfolgreiche Analytics-Datenspeicherung
- Messverfahren: Automated Tests mit simulierten App-Kills und Restarts
- Akzeptanzkriterium: <0.1% Datenverlust bei unerwarteten App-Beendigungen

NF-08 Network Error Recovery:

- Messbare Anforderung: 95% erfolgreiche Auto-Recovery nach Netzwerkunterbrechungen <30 Sekunden
- Messverfahren: Automated Tests mit simulierten Netzwerkausfällen
- Akzeptanzkriterium: App funktionsfähig binnen 5 Sekunden nach Netzwerk-Wiederherstellung

Performance (Leistung)

NF-09 App Launch Time:

- Messbare Anforderung: Cold Start <3 Sekunden auf Mittelklasse-Geräten (iPhone 12, Android SDK 30)

- Messverfahren: Automated Performance Tests mit App Launch Metrics
- Akzeptanzkriterium: 90% der Cold Starts unter 3 Sekunden

NF-10 Memory Usage PowerMode:

- Messbare Anforderung: RAM-Verbrauch <500 MB während PowerMode mit drei aktiven WebViews
- Messverfahren: Memory Profiling über 30-Minuten PowerMode Sessions
- Akzeptanzkriterium: Durchschnittlicher Peak Memory <500 MB

NF-11 Battery Efficiency:

- Messbare Anforderung: Max. 15% höherer Batterieverbrauch als native Apps bei vergleichbarer Nutzung
- Messverfahren: 2-Stunden Nutzungstests vs. native YouTube/TikTok/Instagram Apps
- Akzeptanzkriterium: <15% zusätzlicher Battery Drain verglichen mit nativen Apps

Supportability (Unterstützbarkeit)

NF-12 Error Logging Coverage:

- Messbare Anforderung: 100% kritischer Fehler werden mit Stack Trace und Context geloggt
- Messverfahren: Code Review und Error Injection Tests
- Akzeptanzkriterium: Alle Crashes und kritische Fehler haben vollständige Log-Einträge

NF-13 Debug Information Availability:

- Messbare Anforderung: Analytics und Tracking-Status in <3 Taps vom Hauptscreen erreichbar
- Messverfahren: UI Navigation Tests
- Akzeptanzkriterium: Debug/Analytics Info maximal 3 Screen-Navigationen entfernt

NF-14 Platform Compatibility:

- Messbare Anforderung: 100% Funktionalität auf iOS 15+ und Android 11+ (API Level 30+)
- Messverfahren: Device Compatibility Testing auf 10 verschiedenen Geräten
- Akzeptanzkriterium: Alle Core Features funktional auf Minimum OS Versionen

Qualitätssicherungsmaßnahmen

Continuous Integration Requirements:

- Automated Testing: Alle NF-Anforderungen werden in CI/CD Pipeline getestet
- Performance Monitoring: Realtime Tracking von Response Times und Memory Usage
- Crash Reporting: Integration mit Crashlytics/Sentry für Production Monitoring
- User Analytics: Heatmaps und User Journey Tracking für Usability Verification

Acceptance Criteria Matrix:

Anforderung	Messverfahren	Tool	Frequency	Threshold
NF-03	UI Response Tests	Detox	Every Build	<2s
NF-06	Crash Analytics	Crashlytics	Continuous	<0.1%
NF-09	Launch Performance	Flipper	Daily	<3s
NF-10	Memory Profiling	Xcode Instruments	Weekly	<500MB

Diese Aufgabe wurde mit KI gemeistert. ## References