# Product Requirements Document (PRD) - Activity Tracker

## 1. Executive Summary

**Product Name:** Activity Tracker

**Version:** 1.0 (MVP)

**Last updated**: July 7, 2025

**Target Platform:** iOS (iPhone)

**Development Timeline:** 3-7 days

**Purpose:** A minimal iOS app that allows users to quickly track activity completion with easy screen-level access and historical insights.

## 2. Product Overview

### 2.1 Problem Statement

Users need a simple way to track when they complete activities (like playing tennis, reading, exercising) without the complexity of full-featured apps. Current solutions either require too many steps or clutter other apps (like using Health app for non-health activities).

### 2.2 Target User

- Individuals who want to maintain awareness of their activity patterns
    - People with recurring habits (hobbies, health goals, learning routines)
- Users who prefer minimal, focused tools over feature-heavy apps
- People who value quick, frictionless interaction

### 2.3 Success Metrics

- Time from screen to marking activity complete: <5 seconds
- Daily active usage for activity tracking
- User retention through consistent activity logging

# 3. Core Features

## 3.1 In-App Features

### Activity Creation Flow

1. Open app → "Add Activity" button
2. Enter activity name / icon (required)
3. Select category from dropdown:
    - Hobby (default)
    - Health (default)
    - Career (default)
    - Education (default)
    - Custom categories (user-defined)
4. Optional details text field
5. Configure screen display settings
6. Save activity

### Main Dashboard

- List view of all created activities
- Each activity shows:
    - Name and category
    - Days since last completion
    - Total completion count
    - Quick complete button

- Sort by user-defined priority ranking

### Activity Management

- Edit activity details
- Delete activities
- Reorder priority ranking
- View completion history with filters:
  - Past 7 days
  - Past 31 days
  - Past 3 months
  - Past 12 months

## 3.2 Widget Features

### Widget Design

- **Size:** Full screen width, height of two app icon rows
- **Layout:** Horizontal swipe navigation between activities
- **Content per activity:**
  - Activity name
  - Icon/emoji representation
  - Days since last completion (e.g., "-1D", "-5D")
  - Completion button (circle icon)

### Widget Behavior

- Shows activities based on user-defined ranking
- Swipe left/right to navigate between activities
- Single activity = no swipe functionality
- Tap completion button = immediate marking (visual feedback: circle icon change from grey to green)

- Button kept showing green after completion
- Button resets to grey after today (use current timezone where user lives in, for MVP, using ETC)

### 3.3 Data Management

- **Storage:** Local device storage (Core Data)
- **Completion Records:** Full history maintained
- **Data Structure:** Activity → Multiple completion timestamps
- **Backup:** None for MVP (local only)

## 4. Technical Specifications

### 4.1 Platform Requirements

- **Minimum iOS Version:** iOS 14.0 (for widget support)
- **Target Devices:** iPhone (all screen sizes)
- **Development Environment:** Xcode with Swift/SwiftUI
- **Widget Framework:** WidgetKit

### 4.2 Architecture Overview

- **Frontend:** SwiftUI for main app interface
- **Widget:** WidgetKit with SwiftUI
- **Data Layer:** Core Data for persistence
- **State Management:** ObservableObject/StateObject pattern

### 4.3 Data Model

```
Activity Entity:
- id: UUID
- name: String
- category: String
- optionalDetails: String?
```

```
- createdDate: Date
- priorityRank: Int
- iconName: String?

Completion Entity:
- id: UUID
- activityId: UUID (relationship)
- completedDate: Date
- source: String ("widget" or "app")
```

# 5. User Experience Flow

## 5.1 First-Time Setup

1. App installation
2. Onboarding (optional skip)
3. Create first activity
4. Add widget to home screen

## 5.2 Daily Usage

1. **Quick Complete:** Home screen widget → tap completion button → visual feedback
2. **Review Progress:** Open app → view dashboard with days since last completion
3. **Add New Activity:** App → "Add Activity" → creation flow
4. **View History:** App → select activity → view completion history

## 5.3 Edge Cases

- No activities created: Show empty state with "Add Activity" CTA
- Widget with no activities: Show "Open app to add activities"
- Multiple rapid completions: Prevent duplicate entries within same day

# 6. Success Criteria

## 6.1 MVP Success Metrics

- **Functionality:** All core features working without crashes
- **Performance:** Widget updates within 2 seconds of completion
- **Usability:** Complete activity tracking flow in under 5 seconds
- **Data Integrity:** All completions accurately recorded and displayed

## 6.2 Post-MVP Considerations

- User feedback integration
- Performance optimization
- App Store publication preparation
- Social media showcase preparation

# 7. Development Phases

## Phase 1: Core App (Days 1-2)

- Activity creation and management
- Main dashboard with completion tracking
- Basic data persistence

## Phase 2: Widget Integration (Days 3-4)

- Widget implementation
- Home screen integration
- Cross-communication between app and widget

## Phase 3: Polish & Testing (Days 5-7)

- UI/UX refinement
- Edge case handling

- Testing across different scenarios

# 8. Technical Constraints

## 8.1 MVP Limitations

- Local storage only (no cloud sync)
- Single device support
- No notifications/reminders
- Basic category system
- No data export functionality

## 8.2 Future Enhancements (Post-MVP)

- iCloud sync for multi-device support
- Custom notification reminders
- Advanced analytics and insights
- Data export/import capabilities
- Social sharing features

# 9. Risk Assessment

## 9.1 Technical Risks

- **Widget complexity:** First-time WidgetKit implementation
- **Data sync:** Ensuring app and widget data consistency
- **Performance:** Widget refresh reliability

## 9.2 Mitigation Strategies

- Start with simple widget functionality
- Implement robust data management early
- Test widget behavior extensively

- Use proven SwiftUI patterns

# 10. Definition of Done

## 10.1 MVP Completion Criteria

- [ ] Activity creation and management functional
- [ ] Widget displaying activities with completion buttons
- [ ] Data persistence working correctly
- [ ] App and widget communicating properly
- [ ] Basic UI/UX polish complete
- [ ] Testing completed without major bugs
- [ ] Ready for personal use and social media showcase