Product Requirements Document: I Go You Go Timer

1. Executive Summary

I Go You Go Timer is a specialized workout timer app designed for athletes who train using the "I Go You Go" (IGYG) methodology. IGYG is a partner-based training style where one person exercises while the other rests, then they switch. The core problem: there is no existing timer app that adequately replaces a human training partner for solo IGYG workouts.

Unlike traditional interval timers with fixed work/rest periods, this app calculates rest periods dynamically based on actual work time multiplied by a configurable ratio. This mimics the natural flow of partner training where your rest duration equals however long your partner took to complete their set.

MVP Goal: Deliver a functional Android timer with configurable work:rest ratios that reliably runs in the background, enabling solo IGYG training sessions.

---

2. Mission

Mission Statement: Enable athletes to perform effective IGYG-style workouts without a training partner by providing a timer that dynamically calculates rest periods based on actual work time.

Core Principles:

1. **Simplicity First** - The app should be immediately usable without a learning curve
2. **Reliability** - The timer must work flawlessly in the background with screen off
3. **Flexibility** - Support various work:rest ratios for different training styles
4. **Gym-Ready** - Large, readable display visible from a distance during workouts
5. **Offline-First** - Never require internet connectivity

---

3. Target Users

Primary Persona: Solo IGYG Athlete

- Trains functional fitness / CrossFit-style workouts
- Familiar with IGYG methodology from partner training
- Wants to maintain IGYG training structure when training alone
- Comfortable sideloading APKs (for MVP)
- Technical comfort: Moderate (can navigate app settings, understands ratios)

Key Pain Points:

- No existing timer apps support dynamic rest calculation based on work time
- Traditional interval timers require guessing work duration in advance
- Fixed intervals don't adapt to natural variation in set completion times

User Needs:

- Tap a button when work is done, have rest automatically calculated
- Configurable ratios to adjust intensity (1:1 for equal work/rest, 1:2 for more recovery)
- Clear audio cues when rest is ending
- Reliable background operation during workouts

---

4. MVP Scope

In Scope (v1 - MVP)

Core Functionality: - [x] Single workout mode: Work -> Rest -> Work -> Rest for N rounds - [x] Dynamic rest calculation (work time x ratio) - [x] Configurable work:rest ratios (presets: 0.5, 1, 1.5, 2 + custom decimal input) - [x] Configurable number of rounds (2-100+) - [x] Manual tap/button to end work phase - [x] Automatic transition from rest to work - [x] Pause/resume functionality - [x] Stop workout with confirmation dialog

Display: - [x] Current round indicator - [x] Elapsed work time (during work phase) - [x] Rest countdown (during rest phase) - [x] Total elapsed time - [x] Large, gym-readable numbers - [x] Dark mode and light mode

Audio: - [x] Beep countdown during final seconds of rest (e.g., 3, 2, 1) - [x] Distinct beeps for final round - [x] Background audio support (screen off, app backgrounded)

Technical: - [x] Android APK (sideloadable) - [x] Offline-first (no internet required) - [x] Local storage for app preferences - [x] State persistence (recover from app kill mid-workout)

Out of Scope (Future Phases)

Deferred Features: - [ ] Group/Ladder mode (sets within groups with group rest) - [ ] Saved workout templates - [ ] Customizable display (show/hide specific elements) - [ ] Customizable colors for work/rest phases - [ ] Voice commands to end work phase - [ ] iOS build - [ ] Web app - [ ] Workout history/logging - [ ] Health app integrations - [ ] Sharing/exporting templates - [ ] App Store / Play Store deployment - [ ] User accounts/login - [ ] Analytics/telemetry

---

5. User Stories

Primary User Stories

US-1: Start a Simple Workout > As an athlete, I want to quickly start a workout with my preferred ratio and round count, so that I can begin training without complex setup.

Example: User opens app, selects 1:1 ratio, sets 10 rounds, taps "Start". Workout begins immediately.

US-2: Complete Work Phase > As an athlete, I want to tap a button when I finish my reps, so that my rest period is calculated based on how long I actually worked.

Example: User completes 10 burpees in 47 seconds, taps "Done". At 1:1 ratio, rest countdown shows 47 seconds.

US-3: Receive Rest Warning > As an athlete, I want audio beeps as my rest period ends, so that I can prepare for the next work phase without watching the screen.

Example: At 3 seconds remaining, user hears beep…beep…beep, then begins next round.

US-4: Train with Screen Off > As an athlete, I want the timer to work with my phone screen off, so that I can save battery and not worry about accidental touches.

Example: User starts workout, locks phone, places it nearby. Timer continues running, beeps play through speaker.

US-5: Pause Mid-Workout > As an athlete, I want to pause the timer if interrupted, so that I can handle interruptions without losing my workout progress.

Example: User gets a phone call, taps pause, handles call, taps resume. Timer continues from where it left off.

US-6: Adjust Intensity via Ratio > As an athlete, I want to choose different work:rest ratios, so that I can vary workout intensity based on my training goals.

Example: For conditioning, user selects 1:0.5 (less rest). For strength, user selects 1:2 (more rest).

US-7: Know My Progress > As an athlete, I want to see which round I'm on and total elapsed time, so that I can track my progress through the workout.

Example: Display shows "Round 7 of 10" and "Total: 12:34".

US-8: End Workout Safely > As an athlete, I want a confirmation before stopping a workout, so that I don't accidentally lose progress.

Example: User taps "Stop", sees "End workout? You're on round 7 of 10", can confirm or cancel.

---

6. Core Architecture & Patterns

High-Level Architecture

```
+------------------------------------------------------------+
|                    React Native App                         |
+------------------------------------------------------------+
|  Screens        |  Features (Vertical Slices)              |
|  - Home         |  - /timer (core workout logic)           |
|  - Workout      |  - /settings (preferences)               |
|  - Settings     |  - /shared (cross-cutting utilities)     |
+------------------------------------------------------------+
|  State: React Context + useReducer                         |
+------------------------------------------------------------+
|  Persistence: AsyncStorage                                 |
+------------------------------------------------------------+
|  Native Modules: Background Timer, Audio, Wake Lock        |
+------------------------------------------------------------+
```

Directory Structure (Vertical Slice Architecture)

```
/src
  /features
    /timer
      TimerScreen.tsx         # Main workout screen
      useTimer.ts             # Core timer logic hook
      timerReducer.ts         # State management
      timerContext.tsx        # Context provider
      timerTypes.ts           # TypeScript types
      timerUtils.ts           # Pure utility functions
      /components
        WorkPhase.tsx         # Work phase display
        RestPhase.tsx         # Rest phase display
        TimerControls.tsx     # Start/pause/stop buttons
        RoundIndicator.tsx    # Current round display
        TimeDisplay.tsx       # Large time display component
    /settings
      SettingsScreen.tsx
      useSettings.ts
      settingsContext.tsx
      settingsTypes.ts
      /components
        RatioPicker.tsx
        ThemeToggle.tsx
    /home
      HomeScreen.tsx
      /components
        WorkoutSetup.tsx      # Ratio + rounds configuration
        StartButton.tsx
  /shared
    /components
      Button.tsx
      Modal.tsx
      ConfirmDialog.tsx
    /hooks
      useBackgroundTimer.ts   # Background execution
      useAudio.ts             # Beep sounds
      useWakeLock.ts          # Keep CPU awake
      usePersistence.ts       # State recovery
    /utils
      formatTime.ts
      calculateRest.ts
    /constants
      ratioPresets.ts
      audioAssets.ts
  /navigation
    AppNavigator.tsx
  App.tsx
```

Key Design Patterns

1. **Vertical Slice Architecture** - Each feature owns its complete stack (UI, logic, types)
2. **Custom Hooks** - Encapsulate complex logic (timer, audio, background execution)
3. **Reducer Pattern** - Predictable state transitions for timer phases
4. **Context Providers** - Share state without prop drilling
5. **Pure Utility Functions** - Testable calculations (rest time, formatting)

Timer State Machine

```
              start              tap done
  [IDLE] ------------> [WORK] ------------> [REST]
    ^                    |                     |
    |                    | pause               | countdown
    |                    v                     | complete
    |                [PAUSED]                  |
    |                    |                     |
    |      stop          | resume              |
    +--------------------+---------------------+
                         |
                   (if final round)
                         |
                         v
                    [COMPLETE]
```

7. Features

7.1 Workout Setup

Purpose: Configure workout parameters before starting

Operations: - Select work:rest ratio from presets (0.5, 1, 1.5, 2) - Enter custom ratio (decimal, e.g., 1.267) - Set number of rounds (2-100+, numeric input or stepper) - Start workout

Key Features: - Ratio presets displayed as tappable chips - "Custom" option reveals decimal input - Sensible defaults (1:1 ratio, 10 rounds) - Persist last-used settings

7.2 Work Phase

Purpose: Track time during active exercise

Display Elements: - Phase indicator: "WORK" - Current round: "Round 3 of 10" - Elapsed work time: Large, counting up (e.g., "0:47") - Total workout time: Smaller, secondary

Controls: - "DONE" button (large, prominent) - ends work phase - "PAUSE" button - pauses timer - "STOP" button - ends workout (with confirmation)

Behavior: - Timer counts up from 0:00 - No maximum work time - Work time is recorded for rest calculation

7.3 Rest Phase

Purpose: Countdown rest period calculated from work time

Display Elements: - Phase indicator: "REST" - Current round: "Round 3 of 10" - Rest countdown: Large, counting down (e.g., "0:32") - Total workout time: Smaller, secondary

Controls: - "PAUSE" button - pauses countdown - "STOP" button - ends workout (with confirmation)

Behavior: - Rest time = work time x ratio - Countdown from calculated rest to 0 - Audio beeps at 3, 2, 1 seconds remaining - Auto-transition to next work phase (or complete if final round)

7.4 Audio System

Purpose: Provide audio cues without requiring visual attention

Sounds: - Rest countdown beeps: Short beep at 3, 2, 1 seconds - Final round indicator: Distinct beep pattern when starting final round - Work start: Optional brief tone when rest ends

Requirements: - Must play with screen off - Must play over other audio (workout music) - Respect device volume settings

7.5 Background Execution

Purpose: Ensure timer runs reliably when app is not in foreground

Requirements: - Timer continues when screen is off - Timer continues when app is backgrounded - Audio plays when backgrounded - State survives app being killed by OS - Display notification showing workout status (Android requirement)

7.6 Settings

Purpose: Configure app preferences

Options: - Theme: Dark / Light / System - Default ratio: Set preferred starting ratio - Default rounds: Set preferred starting rounds - (Future: display customization, colors)

---

8. Technology Stack

Core Framework

- **React Native** (0.73+) - Cross-platform mobile framework
- **Expo** (SDK 50+) - Managed workflow for easier development
- **TypeScript** (5.0+) - Type safety

State Management

- **React Context** - Global state sharing
- **useReducer** - Predictable state transitions

Storage

- **@react-native-async-storage/async-storage** - Persist settings and state

Background Execution

- **expo-task-manager** - Background task registration
- **expo-background-fetch** - Periodic background execution
- **react-native-background-timer** - Accurate background timing
  - *[FLAG FOR RESEARCH: Evaluate expo-background-fetch vs react-native-background-timer vs react-native-background-actions for this use case]*

Audio

- **expo-av** - Audio playback
  - *[FLAG FOR RESEARCH: Audio mixing modes, playing over other apps, audio focus handling]*

Wake Lock

- **expo-keep-awake** - Prevent screen sleep during workout (optional feature)
  - *[FLAG FOR RESEARCH: CPU wake lock for background, Android Doze mode handling]*

Navigation

- **@react-navigation/native** - Screen navigation
- **@react-navigation/stack** - Stack navigator

Development

- **Jest** - Unit testing
- **React Native Testing Library** - Component testing
- **ESLint** - Code linting
- **Prettier** - Code formatting

Build & Deployment

- **EAS Build** (Expo Application Services) - APK generation
  - *[FLAG FOR RESEARCH: APK signing for sideloading, EAS Build configuration]*

---

9. Security & Configuration

Security Scope

In Scope: - [x] No sensitive data collection (no accounts, no personal info) - [x] Local-only storage (no network transmission) - [x] No analytics or telemetry

Out of Scope (Not Needed): - [ ] Authentication/authorization - [ ] API security - [ ] Data encryption (no sensitive data)

Configuration

Environment Variables: - None required for MVP (fully offline app)

App Configuration: - `app.json` / `app.config.js` - Expo configuration - Package name: `com.igygtimer.app` (or similar) - Minimum Android SDK: 21 (Android 5.0) - *[FLAG FOR RESEARCH: Confirm minimum SDK for background execution features]*

Permissions (Android)

```json
{
  "android": {
    "permissions": [
      "FOREGROUND_SERVICE",
      "WAKE_LOCK",
      "RECEIVE_BOOT_COMPLETED"
    ]
  }
}
```

[FLAG FOR RESEARCH: Exact permissions needed for reliable background timer execution]

---

10. API Specification

Not Applicable - This is an offline-only mobile app with no backend API.

---

11. Success Criteria

MVP Success Definition

The MVP is successful when a user can: 1. Configure a workout (ratio + rounds) 2. Complete a full IGYG workout with the timer running reliably 3. Hear audio cues during rest without looking at the screen 4. Have the timer work correctly with the screen off

Functional Requirements

- ☒ Timer accurately tracks work time (+/-100ms precision)
- ☒ Rest calculation is mathematically correct (work x ratio)
- ☒ Rounds increment correctly
- ☒ Pause/resume maintains accurate time
- ☒ Audio beeps play at correct countdown moments
- ☒ Background execution works for minimum 30-minute workout
- ☒ App recovers state after being killed by OS
- ☒ Settings persist between app launches

Quality Indicators

- Timer drift < 1 second over 30-minute workout
- Audio latency < 200ms
- App launch to workout start < 5 seconds
- No crashes during normal workout flow
- Battery usage reasonable (< 5% for 30-minute workout)
  - *[FLAG FOR RESEARCH: Baseline battery usage benchmarks for background timer apps]*

User Experience Goals

- Zero learning curve - usable on first launch
- Large touch targets for sweaty hands / gym use
- Readable from 6+ feet away
- Works reliably - user trusts it won't fail mid-workout

---

12. Implementation Phases

Phase 1: Core Timer (MVP)

Goal: Functional timer with basic UI

Deliverables: - [x] Project setup (Expo, TypeScript, folder structure) - [x] Timer state machine (idle, work, rest, paused, complete) - [x] Work phase with elapsed time display - [x] Rest phase with countdown display - [x] Ratio configuration (presets + custom) - [x] Rounds configuration - [x] Basic UI (functional, not polished) - [x] Pause/resume functionality - [x] Stop with confirmation

Validation: - Can complete a 5-round workout with 1:1 ratio - Timer counts accurately - State transitions work correctly

Phase 2: Audio & Background

Goal: Reliable background operation with audio cues

Deliverables: - [x] Audio beep system (3, 2, 1 countdown) - [x] Final round audio indicator - [x] Background timer execution - [x] Foreground service notification (Android) - [x] State persistence (survive app kill) - [x] Wake lock handling

Validation: - Complete workout with screen off - Audio plays reliably in background - App recovers after force-close

Phase 3: Polish & Settings

Goal: Production-ready MVP

Deliverables: - [x] Dark/light theme support - [x] Settings screen (theme, defaults) - [x] Improved UI (large numbers, gym-readable) - [x] Persist last-used settings - [x] Error handling and edge cases - [x] APK build and sideload testing

Validation: - Real-world gym testing - Multiple device testing - Battery usage acceptable

Phase 4: Post-MVP (Future)

Goal: Enhanced functionality

Deliverables: - [ ] Group/Ladder mode - [ ] Saved templates - [ ] Display customization - [ ] Color customization - [ ] iOS build - [ ] App store deployment

---

13. Future Considerations

Group/Ladder Mode (v2 Priority)

Structure for ladder workouts:

```
Group 1:
  Set 1: work -> rest (set ratio)
  Set 2: work -> rest (set ratio)
  ...
  Set N: work -> rest (set ratio)
  -> Group Rest (group ratio x total group time)
Group 2: (repeat)
```

Configuration needed: - Sets per group (configurable) - Set-level work:rest ratio - Group-level work:rest ratio - Number of groups - Group rest calculation: work time only OR work+rest time (user choice)

Saved Templates (v2)

- Save workout configurations with names
- Quick-start from template
- Edit/delete templates

Display Customization (v2)

- Toggle visibility of: current round, elapsed time, total time
- Choose which metrics to show during work vs rest

Color Customization (v2)

- Custom colors for work phase background
- Custom colors for rest phase background
- High contrast options

Platform Expansion (v3+)

- iOS build via Expo
- Web app via React Native Web

- Wear OS companion (show timer on watch)

Advanced Features (Future)

- Voice command to end work phase (“Done!”)
- Haptic feedback option
- Widget for quick-start
- Integration with fitness trackers

---

14. Risks & Mitigations

Risk 1: Background Execution Unreliability

Risk: Android aggressively kills background apps; timer may stop unexpectedly.

Likelihood: High

Impact: Critical - core functionality broken

Mitigation: - Use foreground service with persistent notification - Implement proper wake locks - Test on multiple Android versions and manufacturers (Samsung, Xiaomi known for aggressive battery management) - Provide user guidance for disabling battery optimization for the app - Persist state frequently for recovery

[FLAG FOR RESEARCH: Android manufacturer-specific battery optimization behaviors]

Risk 2: Audio Playback Issues

Risk: Audio may not play reliably when backgrounded or when other audio is playing.

Likelihood: Medium

Impact: High - user misses cues, workout flow broken

Mitigation: - Use proper audio focus handling - Test with Spotify/podcast apps running - Implement audio session category for “playback” that mixes with other audio - Fallback: vibration pattern as backup cue

[FLAG FOR RESEARCH: expo-av audio mixing configuration]

Risk 3: State Loss on App Kill

Risk: User loses workout progress if Android kills the app.

Likelihood: Medium

Impact: Medium - frustrating but recoverable

Mitigation: - Persist state to AsyncStorage on every phase transition - Persist periodically during long phases (every 5 seconds) - On app launch, check for in-progress workout and offer to resume

Risk 4: Timer Drift

Risk: Timer becomes inaccurate over long workouts due to JS timing limitations.

Likelihood: Low-Medium

Impact: Medium - affects rest calculations

Mitigation: - Use native background timer module, not JS setInterval - Base calculations on absolute timestamps, not accumulated deltas - Test accuracy over 60+ minute sessions

Risk 5: Poor Gym Usability

Risk: UI is hard to use with sweaty hands or read from distance.

Likelihood: Medium

Impact: Medium - degrades user experience

Mitigation: - Large touch targets (minimum 48dp, prefer 64dp+) - High contrast colors - Large fonts (timer display 80pt+) - Real-world gym testing during development

---

15. Appendix

A. Ratio Examples

Ratio	Work Time	Rest Time	Use Case
1:0.5	60s	30s	High intensity conditioning
1:1	60s	60s	Standard IGYG, balanced
1:1.5	60s	90s	Moderate recovery
1:2	60s	120s	Strength focus, full recovery
1:3	60s	180s	Heavy lifting, max recovery

B. State Recovery Data Structure

```
interface PersistedWorkoutState {
  workoutConfig: {
    ratio: number;
    totalRounds: number;
  };
  currentState: {
    phase: 'work' | 'rest' | 'paused';
    currentRound: number;
    workStartTime: number;      // timestamp
    workElapsedMs: number;      // if paused during work
    restStartTime: number;      // timestamp
    restDurationMs: number;     // calculated rest time
    restElapsedMs: number;      // if paused during rest
    totalElapsedMs: number;
  };
  savedAt: number;              // timestamp
}
```

C. Research Items Summary

Items flagged throughout this document for further research during implementation:

1. **Background Timer Libraries** - Evaluate `expo-background-fetch` vs `react-native-background-timer` vs `react-native-background-actions`
2. **Audio Mixing** - Configure `expo-av` for playing over other audio apps
3. **Audio Focus** - Handling audio interruptions (phone calls)
4. **Wake Locks** - CPU wake lock for background, Android Doze mode
5. **Minimum SDK** - Confirm Android SDK level for all required features
6. **Android Permissions** - Exact permission set for background execution

7. **Battery Benchmarks** - Baseline battery usage for background timer apps
8. **Manufacturer Behaviors** - Samsung, Xiaomi, etc. battery optimization quirks
9. **APK Signing** - EAS Build configuration for sideloadable APK
10. **Expo vs Bare** - Confirm Expo managed workflow supports all background features (may need to eject)

D. Glossary

- **IGYG (I Go You Go)** - Partner workout format where partners alternate work/rest
- **Work Phase** - Active exercise period, user-terminated
- **Rest Phase** - Recovery period, timer-terminated based on ratio
- **Ratio** - Multiplier applied to work time to calculate rest time
- **Round** - One complete work + rest cycle
- **Group** - (Future) Collection of sets with group-level rest
- **Ladder** - (Future) Workout structure with incrementing sets (1, 2, 3, 4, 5 reps)

Document Version: 1.0 Created: 2026-01-21 Status: Draft - Pending Review