EMIR iOS/iPadOS App — Software Design Document (SDD)

# 1. Purpose

The EMIR iOS/iPadOS app collects location and activity data, works as fallback if no paired Apple Watch is present, and manages push notifications for commands/messages. When paired with an Apple Watch, it complements the watch by handling permissions, fallback location tracking, and data relay if needed. Fully supports both iPhone and iPad devices with Wi-Fi or LTE.

# 2. Platform & Tech Stack

• OS: iOS 15+, iPadOS 15+  
• Language: Swift  
• Location: CoreLocation  
• Activity: CoreMotion  
• Health: HealthKit (when Apple Watch paired)  
• Background: BackgroundTasks  
• Push Notifications: Apple Push Notification Service (APNs)  
• Secure Storage: iOS Keychain  
• Minimum SDK: iOS/iPadOS 15  
• IDE: Xcode

# 3. Key Features

• Smart location tracking using CoreLocation.  
• Activity recognition with CoreMotion.  
• Heart rate data via HealthKit if Apple Watch is paired.  
• Significant-change location updates for battery saving.  
• Batch upload of telemetry data via HTTPS.  
• Receive commands/messages via APNs push notifications.  
• Runs as fallback if no watch is connected.  
• Full iPad support — same behavior as iPhone.  
• Secure storage of tokens with Keychain.  
• Local offline buffer for retrying uploads.

# 4. Scenarios Supported

• iPhone + Apple Watch: Watch handles health/location, phone relays/fallback.  
• iPhone alone: Gathers location/activity directly.  
• iPad + Apple Watch: Same as iPhone, rare but supported.  
• iPad alone: Gathers location/activity directly without watch.

# 5. Main Components

• LocationService: Uses CoreLocation with smart triggers.  
• ActivityService: Uses CoreMotion to detect walk/run/drive.  
• HealthMonitor: Interfaces with HealthKit when paired.  
• DataUploader: Batches telemetry for HTTPS upload.  
• PushHandler: Handles incoming APNs payloads.  
• LocalStore: Caches data offline.  
• Security: Manages Keychain tokens and auth.

# 6. Push Notifications

• Uses Apple Push Notification Service (APNs).  
• Registers device token and uploads to backend.  
• Backend sends minimal payloads with command ID only.  
• App wakes on push, fetches full command securely via HTTPS.

# 7. Security

• All data transfer uses HTTPS (TLS 1.2+).  
• Enforces HSTS.  
• Optional certificate pinning for extra protection.  
• Keychain used for secure API keys/JWTs.  
• Push payloads never contain raw health/location data.

# 8. Battery Saving Strategy

• Uses significant-change location service to reduce GPS usage.  
• Batches uploads and uses BackgroundTasks to defer work.  
• No persistent sockets — push notifications wake app on demand.

# 9. Data Fields

• Location: timestamp, latitude, longitude, accuracy, activity.  
• Health: timestamp, heart rate, anomaly flag.  
• Activity: walk/run/drive.  
• Commands: command ID, status.

# 10. Build & Deployment

• Developed in Xcode.  
• APNs certificates and provisioning profiles configured.  
• Distributed via App Store or enterprise MDM solutions.

# 11. Future Extensions

• BLE HR sensor fallback if no watch is paired.  
• Local map preview for user.  
• Enhanced local status and log view.