Wearable LIA Health Monitoring System

**Complete Technical Documentation***Mobile Application & Backend API*with Proprietary Three-Layer Processing System

Version 1.0.0  
October 19, 2025

# Table of Contents

1. 1. Executive Summary
2. 2. System Overview
3. 3. Architecture
4. 4. Mobile Application
5. 5. Backend API System
6. 6. Proprietary Processing Layers
7. 7. LIA Integration
8. 8. Data Flow & Communication
9. 9. API Reference
10. 10. Data Stream Simulation
11. 11. Testing & Demonstration
12. 12. Deployment Guide
13. 13. Appendix

# 1. Executive Summary

The Wearable LIA Health Monitoring System is a comprehensive health analytics platform that combines cutting-edge mobile technology with advanced backend processing to deliver real-time health insights. The system integrates wearable biosignal devices through BLE connectivity and processes data through three proprietary layers—Clarity™, iFRS™, and Timesystems™—before generating Lifestyle Intelligence Analysis (LIA) insights.

## Key Achievements

* Complete integration of LIA engine into React Native mobile application
* Real-time BLE connectivity with wearable devices (both physical and simulated)
* Implementation of proprietary three-layer processing system (Clarity™, iFRS™, Timesystems™)
* FastAPI-based backend with REST and WebSocket support
* Real-time biosignal streaming at 10 Hz update rate
* Comprehensive health condition classification with 10 distinct states
* Multi-dimensional wellness scoring across 5 health metrics

## Technology Stack

|  |  |
| --- | --- |
| Component | Technology |
| Mobile Frontend | React Native with Expo, TypeScript |
| UI Framework | React Native Paper (Material Design) |
| State Management | Zustand |
| Backend | Python 3.10+, FastAPI, Uvicorn |
| Data Processing | NumPy, SciPy |
| API Protocol | REST + WebSocket |
| BLE Communication | Expo BLE SDK |
| Data Persistence | AsyncStorage (mobile), In-memory (backend) |

# 2. System Overview

## 2.1 Purpose and Scope

The Wearable LIA Health Monitoring System provides comprehensive health monitoring through continuous biosignal analysis. The system collects heart rate, blood oxygen saturation (SpO2), body temperature, and activity data from wearable devices, processes this information through multiple proprietary layers, and generates actionable health insights using the LIA engine.

## 2.2 Core Components

|  |  |  |
| --- | --- | --- |
| Component | Purpose | Technology |
| Mobile Application | User interface and device connectivity | React Native + Expo |
| Backend API | Data processing and analysis | FastAPI + Python |
| Clarity™ Layer | Signal quality enhancement | Adaptive filtering |
| iFRS™ Layer | Frequency analysis | FFT + HRV extraction |
| Timesystems™ Layer | Temporal pattern recognition | Circadian analysis |
| LIA Engine | Health insights generation | Multi-factor classification |
| BLE Simulator | Device data simulation | Realistic biosignal generation |

## 2.3 Key Features

* Real-time biosignal streaming and processing
* BLE device connectivity with automatic reconnection
* Three-layer proprietary signal processing
* LIA health insights with 10 condition classifications
* Multi-dimensional wellness scoring
* REST API and WebSocket support
* Session management and history tracking
* Comprehensive processing logs
* Demonstration and testing endpoints

# 3. Architecture

## 3.1 High-Level Architecture

The system follows a client-server architecture with the mobile application acting as the client and the FastAPI backend serving as the processing engine. Communication occurs via HTTP REST API for request-response patterns and WebSocket for real-time streaming.

### Architecture Diagram

┌──────────────────────────────────┐  
│ Mobile Application │  
│ (React Native + Expo) │  
│ - UI Components │  
│ - BLE Service │  
│ - State Management (Zustand) │  
└────────────┬─────────────────────┘  
 │ HTTP/WebSocket  
 ▼  
┌──────────────────────────────────┐  
│ FastAPI Backend Server │  
│ - REST API Endpoints │  
│ - WebSocket Streaming │  
│ - Session Management │  
└────────────┬─────────────────────┘  
 │  
 ▼  
┌──────────────────────────────────┐  
│ Processing Pipeline │  
│ │  
│ ┌──────────────────────────┐ │  
│ │ 1. BLE Simulator │ │  
│ │ (Biosignal Generation)│ │  
│ └──────────┬───────────────┘ │  
│ ▼ │  
│ ┌──────────────────────────┐ │  
│ │ 2. Clarity™ Layer │ │  
│ │ (Signal Enhancement) │ │  
│ └──────────┬───────────────┘ │  
│ ▼ │  
│ ┌──────────────────────────┐ │  
│ │ 3. iFRS™ Layer │ │  
│ │ (Frequency Analysis) │ │  
│ └──────────┬───────────────┘ │  
│ ▼ │  
│ ┌──────────────────────────┐ │  
│ │ 4. Timesystems™ Layer │ │  
│ │ (Temporal Analysis) │ │  
│ └──────────┬───────────────┘ │  
│ ▼ │  
│ ┌──────────────────────────┐ │  
│ │ 5. LIA Engine │ │  
│ │ (Health Insights) │ │  
│ └──────────────────────────┘ │  
│ │  
└──────────────────────────────────┘

## 3.2 Communication Flow

1. 1. Mobile app connects to backend via POST /api/v1/connect
2. 2. Backend establishes session and initializes BLE simulator
3. 3. Mobile app requests stream data via GET /api/v1/stream or WebSocket
4. 4. Backend processes current biosignal data through all layers
5. 5. Processed data with LIA insights returned to mobile app
6. 6. Mobile app displays real-time biosignals, quality metrics, and health insights

# 4. Mobile Application

## 4.1 Application Structure

The mobile application is built with React Native and Expo, providing cross-platform compatibility for iOS and Android devices. The app follows a modular architecture with clear separation of concerns.

### Directory Structure

BLE-wearable-App/  
├── App.tsx # Main entry point  
├── src/  
│ ├── components/ # Reusable UI components  
│ │ ├── BiosignalCard.tsx # Biosignal display  
│ │ ├── WellnessCard.tsx # Wellness assessment  
│ │ └── LayerProcessingCard.tsx # Layer details  
│ ├── screens/ # Application screens  
│ │ ├── dashboard/ # Dashboard screen  
│ │ ├── livedata/ # Live data screen  
│ │ ├── logs/ # Processing logs screen  
│ │ ├── demo/ # API demo screen  
│ │ ├── session/ # Session management  
│ │ ├── history/ # Session history  
│ │ └── settings/ # Settings screen  
│ ├── services/ # Backend services  
│ │ ├── backendApi.ts # Backend API integration  
│ │ ├── bluetooth.ts # BLE device management  
│ │ └── openai.ts # AI chat service  
│ ├── store/ # State management  
│ │ └── index.ts # Zustand store  
│ ├── navigation/ # Navigation config  
│ │ └── AppNavigator.tsx # Main navigator  
│ └── types/ # TypeScript definitions  
└── assets/ # Static assets

## 4.2 Key Features

### Dashboard Screen

* Connection status display with device information
* Today's summary (sessions, active time, insights)
* Backend Integration section with quick access cards
* Recent sessions history
* Quick action buttons for starting sessions

### Live Data Screen

* Real-time biosignal cards (Heart Rate, SpO2, Temperature, Activity)
* Quality metrics from Clarity™ layer for each signal
* LIA wellness assessment with 5-dimensional scoring
* Expandable layer processing details (Clarity™, iFRS™, Timesystems™)
* Auto-refresh every 1 second with toggle control
* Pull-to-refresh functionality
* Connection status indicator

### Processing Logs Screen

* Live backend processing logs display
* Auto-refresh every 2 seconds
* Filterable by log level (ALL, INFO, SUCCESS, WARNING, ERROR)
* Color-coded log entries
* Layer-specific color highlighting
* Expandable log details with data inspection
* Manual refresh capability

### API Demo Screen

* System health status monitoring
* Active sessions and connected clients display
* Service availability checking
* 6 individual endpoint tests
* Bulk "Run All Tests" functionality
* Real-time response inspection
* Performance metrics (response time in milliseconds)
* Error handling demonstration

## 4.3 Backend Integration

The mobile app integrates with the FastAPI backend through a comprehensive API service located in src/services/backendApi.ts. This service provides:

* Platform-specific URL configuration (iOS/Android/Web)
* Health check endpoint integration
* Device connection management
* Real-time stream data retrieval
* LIA prediction endpoints
* Session management
* Processing logs retrieval
* Layer demonstration endpoint
* WebSocket support for real-time streaming
* HTTP polling as fallback (1-second intervals)
* Automatic error handling and retries

# 5. Backend API System

## 5.1 Backend Architecture

The backend is built with FastAPI, a modern Python web framework that provides automatic API documentation, data validation, and async support. The backend runs on port 8000 and provides both REST and WebSocket endpoints.

### Backend Structure

backend/  
├── main.py # FastAPI application  
├── requirements.txt # Python dependencies  
├── models/  
│ └── schemas.py # Pydantic data models  
├── services/  
│ ├── ble\_simulator.py # BLE device simulation  
│ ├── clarity.py # Clarity™ layer  
│ ├── ifrs.py # iFRS™ layer  
│ ├── timesystems.py # Timesystems™ layer  
│ ├── lia\_integration.py # LIA engine  
│ └── session\_manager.py # Session management  
└── utils/  
 └── logger.py # Logging utilities

## 5.2 Core Services

### BLE Simulator

Generates realistic biosignal data simulating a wearable device. Provides continuous streaming at 10 Hz (100ms intervals).

|  |  |  |
| --- | --- | --- |
| Signal | Range | Description |
| Heart Rate | 45-180 BPM | Physiologically realistic variation |
| SpO2 | 90-100% | Normal resting and activity levels |
| Temperature | 35.5-38.5°C | Body temperature variation |
| Activity | 0-150 steps/min | Activity intensity |

### Session Manager

Manages client sessions, tracking connected devices, session metadata, and historical data. Provides session creation, retrieval, and lifecycle management.

## 5.3 API Endpoints

The backend exposes the following endpoints:

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Purpose |
| / | GET | API information and version |
| /api/v1/health | GET | Health check and system status |
| /api/v1/connect | POST | Connect device and create session |
| /api/v1/stream | GET | Get processed biosignal data |
| /api/v1/predict | GET | Get LIA health prediction |
| /api/v1/sessions | POST | Create new monitoring session |
| /api/v1/sessions/{id} | GET | Get session details |
| /api/v1/demo/layers | GET | Complete layer processing demo |
| /api/v1/logs/processing | GET | Get processing logs |
| /ws/stream | WS | Real-time WebSocket streaming |

# 6. Proprietary Processing Layers

The system implements three proprietary layers that sequentially process biosignal data, each adding value through specialized analysis techniques.

## 6.1 Clarity™ Layer - Signal Quality Enhancement

### Purpose

Enhance biosignal quality through adaptive filtering and provide comprehensive quality assessment metrics. This layer ensures that downstream processing works with clean, reliable data.

### Key Features

* Adaptive Noise Reduction: Wavelet-inspired smoothing based on signal quality
* Quality Metrics: Per-channel quality scoring for each biosignal type
* SNR Calculation: Signal-to-noise ratio in decibels (dB)
* Artifact Detection: Motion artifacts, electrode noise, saturation, and dropouts
* Quality Assessment: Categorical rating (Excellent, Good, Fair, Poor)

### Algorithms

* Weighted moving average with exponential decay
* Historical consistency analysis (evaluates signal stability over time)
* Coefficient of variation for stability assessment
* Multi-factor quality scoring combining multiple metrics

### Output Metrics

|  |  |  |
| --- | --- | --- |
| Metric | Range | Interpretation |
| Quality Score | 0.0 - 1.0 | Overall signal quality |
| SNR | 15 - 60 dB | Signal-to-noise ratio |
| Heart Rate Quality | 0.0 - 1.0 | HR signal quality |
| SpO2 Quality | 0.0 - 1.0 | Oxygen saturation quality |
| Temperature Quality | 0.0 - 1.0 | Temperature sensor quality |
| Activity Quality | 0.0 - 1.0 | Activity sensor quality |
| Noise Reduction Applied | Boolean | Whether filtering was applied |

## 6.2 iFRS™ Layer - Intelligent Frequency Response System

### Purpose

Perform comprehensive frequency domain analysis to extract Heart Rate Variability (HRV) features and classify cardiac rhythms. This layer provides insights into autonomic nervous system function and cardiovascular health.

### Key Features

* FFT Analysis: Fast Fourier Transform for frequency content extraction
* HRV Extraction: RMSSD, SDNN, pNN50, and composite HRV score
* Frequency Bands: VLF (Very Low Frequency), LF (Low Frequency), HF (High Frequency)
* Rhythm Classification: Normal Sinus, Athletic, Elevated, Low, Irregular
* Respiratory Rate: Estimation from HF band analysis
* LF/HF Ratio: Autonomic balance indicator

### Algorithms

* Hanning window application for spectral leakage reduction
* R-R interval computation from heart rate data
* Power spectral density calculation
* Dominant frequency detection
* HRV time-domain and frequency-domain metrics

### Output Metrics

|  |  |  |
| --- | --- | --- |
| Metric | Unit | Description |
| Dominant Frequency | Hz | Primary frequency component |
| HRV Score | 0-100 | Overall heart rate variability |
| RMSSD | milliseconds | Root mean square of successive differences |
| SDNN | milliseconds | Standard deviation of NN intervals |
| pNN50 | percentage | Percentage of successive intervals > 50ms |
| LF/HF Ratio | ratio | Sympathetic/parasympathetic balance |
| Respiratory Rate | breaths/min | Estimated breathing rate |
| Rhythm Type | classification | Cardiac rhythm category |

### Rhythm Classifications

* Normal Sinus: Regular rhythm, HR 60-100 BPM
* Athletic: Regular rhythm, HR 45-60 BPM (trained heart)
* Elevated: Regular rhythm, HR 100-180 BPM (exercise/stress)
* Low: Regular rhythm, HR < 45 BPM
* Irregular: Inconsistent R-R intervals

## 6.3 Timesystems™ Layer - Temporal Analysis

### Purpose

Perform time-domain pattern recognition and circadian rhythm analysis. This layer identifies temporal trends, assesses circadian alignment, and evaluates rhythm health.

### Key Features

* Circadian Phase Identification: Morning, Afternoon, Evening, Night
* Pattern Recognition: Stable, Increasing, Decreasing, Oscillating, Irregular
* Temporal Consistency: Signal stability measurement over time
* Circadian Alignment: Comparison with expected physiological rhythms
* Rhythm Scoring: Overall rhythm health score (0-100)
* Phase Shift Detection: Deviation from expected circadian patterns

### Algorithms

* Linear regression for trend detection
* Autocorrelation analysis for periodicity
* Time-of-day physiological expectations
* Pattern confidence scoring
* Circadian phase calculation based on local time

### Output Metrics

|  |  |  |
| --- | --- | --- |
| Metric | Range/Type | Description |
| Temporal Consistency | 0.0 - 1.0 | Signal stability over time |
| Circadian Alignment | 0.0 - 1.0 | Match with expected rhythms |
| Rhythm Score | 0 - 100 | Overall rhythm health |
| Phase Shift | minutes | Deviation from expected timing |
| Pattern Type | categorical | Identified temporal pattern |
| Circadian Phase | categorical | Time-of-day category |
| Pattern Confidence | 0.0 - 1.0 | Confidence in pattern detection |

### Pattern Types

* Stable: Consistent values over time
* Increasing: Upward trend detected
* Decreasing: Downward trend detected
* Oscillating: Regular periodic variation
* Irregular: No clear pattern identified

# 7. LIA Integration - Lifestyle Intelligence Analysis

## 7.1 Overview

LIA (Lifestyle Intelligence Analysis) is the final processing stage that synthesizes outputs from all three proprietary layers to generate comprehensive health insights. LIA provides condition classification, multi-dimensional wellness scoring, risk assessment, and personalized recommendations.

## 7.2 Health Condition Classification

LIA classifies the current health state into one of 10 distinct conditions based on biosignal patterns, quality metrics, and contextual factors.

|  |  |
| --- | --- |
| Condition | Characteristics |
| Normal Resting | HR 60-80, good HRV, low activity |
| Light Activity | HR 80-100, moderate activity, stable |
| Moderate Exercise | HR 100-140, elevated activity, increasing pattern |
| Intense Exercise | HR 140-180, high activity, elevated stress |
| Deep Rest | HR < 60, minimal activity, excellent HRV |
| Sleep State | HR < 60, no activity, circadian alignment |
| Elevated Stress | HR > 100, low HRV, irregular pattern |
| Relaxation | Decreasing HR, improving HRV, stable |
| Recovery Mode | HR normalizing, activity decreasing |
| Optimal Wellness | Excellent metrics across all dimensions |

## 7.3 Multi-Dimensional Wellness Scoring

LIA evaluates health across five key dimensions, each scored from 0-100:

|  |  |
| --- | --- |
| Dimension | Evaluation Factors |
| Cardiovascular Health | Heart rate, HRV, rhythm classification, Clarity quality |
| Respiratory Health | SpO2 levels, respiratory rate, breathing pattern consistency |
| Activity Level | Steps/min, activity intensity, movement patterns |
| Stress Level | HRV metrics, LF/HF ratio, temporal consistency |
| Overall Wellness | Weighted average of all dimensions |

## 7.4 Risk Assessment

LIA automatically identifies potential risk factors based on pattern analysis:

* Low SpO2: Blood oxygen below 95%
* High Resting Heart Rate: HR > 100 BPM at rest
* Low HRV: Poor heart rate variability indicating stress
* Poor Signal Quality: Unreliable sensor data
* Irregular Rhythm: Inconsistent heart rhythm patterns
* Temperature Abnormality: Outside normal range (36-37.5°C)
* Circadian Misalignment: Significant phase shift from expected patterns

## 7.5 Positive Indicators

LIA also highlights positive health signs:

* Excellent HRV: RMSSD > 40ms, SDNN > 60ms
* Optimal SpO2: Blood oxygen 98-100%
* Strong Circadian Alignment: Rhythm score > 80
* Good Signal Quality: Clarity score > 0.9
* Healthy Resting HR: 60-75 BPM
* Consistent Patterns: High temporal consistency > 0.85

## 7.6 Personalized Recommendations

Based on the current condition and wellness assessment, LIA generates context-aware recommendations:

* Normal Resting: "Maintain current activity levels and hydration"
* Elevated Stress: "Consider breathing exercises or brief meditation"
* Low Activity: "Try a short walk to increase movement"
* Post-Exercise: "Focus on hydration and recovery"
* Poor Sleep Quality: "Establish consistent sleep schedule"

# 8. Data Flow & Communication

## 8.1 Complete Processing Pipeline

Data flows through the system in a sequential pipeline, with each stage adding value through specialized processing:

Step 1: BLE Simulator (or Physical Device)  
├─ Generates/receives biosignal data  
├─ Rate: 10 Hz (100ms intervals)  
└─ Output: {heart\_rate, spo2, temperature, activity}  
 ↓  
Step 2: Clarity™ Layer  
├─ Input: Raw biosignals  
├─ Processing:  
│ ├─ Calculate quality metrics for each channel  
│ ├─ Assess signal stability from history  
│ ├─ Apply adaptive noise reduction if quality < 0.7  
│ ├─ Calculate SNR  
│ └─ Detect artifacts  
└─ Output: Enhanced signals + quality assessment  
 ↓  
Step 3: iFRS™ Layer  
├─ Input: Clarity-enhanced signals  
├─ Processing:  
│ ├─ Apply FFT with Hanning window  
│ ├─ Extract R-R intervals from heart rate  
│ ├─ Calculate HRV features (RMSSD, SDNN, pNN50)  
│ ├─ Analyze frequency bands (VLF, LF, HF)  
│ ├─ Classify cardiac rhythm  
│ └─ Estimate respiratory rate  
└─ Output: Frequency features + HRV metrics  
 ↓  
Step 4: Timesystems™ Layer  
├─ Input: iFRS-enhanced signals  
├─ Processing:  
│ ├─ Identify circadian phase (time of day)  
│ ├─ Analyze temporal patterns  
│ ├─ Detect trends (linear regression)  
│ ├─ Calculate temporal consistency  
│ ├─ Assess circadian alignment  
│ └─ Score rhythm health  
└─ Output: Temporal analysis + circadian metrics  
 ↓  
Step 5: LIA Engine  
├─ Input: All layer outputs  
├─ Processing:  
│ ├─ Multi-factor condition classification  
│ ├─ Calculate wellness dimensions  
│ ├─ Identify risk factors  
│ ├─ Identify positive indicators  
│ └─ Generate personalized recommendation  
└─ Output: Health insights + wellness scores  
 ↓  
Final Response to Mobile App

## 8.2 Communication Protocols

### HTTP REST API

Primary communication method for request-response patterns. Used for:

* Device connection establishment
* Single data snapshots
* Session management
* Configuration retrieval
* Processing log access
* Layer demonstration

### WebSocket Streaming

Real-time bidirectional communication for continuous data streaming. Features:

* Update rate: 10 Hz (100ms intervals)
* Automatic reconnection on disconnect
* Lower latency than polling
* Full pipeline data in each message
* Connection status events

### HTTP Polling

Fallback method for environments where WebSocket is unavailable. Characteristics:

* Update interval: 1 second (configurable)
* Compatible with all platforms
* Simple implementation
* Reliable error recovery
* Manual refresh available

## 8.3 Data Update Cycle

1. 1. BLE Simulator generates new data every 100ms
2. 2. Each HTTP request triggers complete pipeline processing
3. 3. WebSocket streams continuous updates at 10 Hz
4. 4. Processing logs capture each layer's activity
5. 5. Mobile app updates UI with latest data
6. 6. Historical data maintained for trend analysis

# 9. API Reference

## 9.1 Base Configuration

Base URL: http://localhost:8000  
WebSocket URL: ws://localhost:8000/ws/stream  
  
Platform-Specific URLs:  
- iOS Simulator: http://localhost:8000  
- Android Emulator: http://10.0.2.2:8000  
- Physical Device: http://YOUR\_IP\_ADDRESS:8000

## 9.2 Endpoint Details

### GET /api/v1/health

Health check and system status

#### Response:

{  
 "status": "healthy",  
 "timestamp": "2024-01-15T10:30:00",  
 "services": {  
 "ble\_simulator": true,  
 "timesystems": true,  
 "ifrs": true,  
 "clarity": true,  
 "lia": true  
 },  
 "connected\_clients": 2,  
 "active\_sessions": 1  
}

### POST /api/v1/connect

Connect mobile client and establish session

#### Request Body:

{  
 "device\_id": "MOBILE\_APP\_001",  
 "device\_type": "mobile\_app",  
 "app\_version": "1.0.0",  
 "user\_id": "user\_123"  
}

#### Response:

{  
 "success": true,  
 "message": "Connected successfully",  
 "session\_id": "session\_abc123",  
 "device\_status": {  
 "device\_id": "WEARABLE\_SIM\_001",  
 "is\_connected": true,  
 "battery\_level": 87.5,  
 "signal\_strength": -55,  
 "firmware\_version": "2.1.4"  
 },  
 "available\_features": [...]  
}

### GET /api/v1/stream

Get current processed biosignal data through all layers

#### Response Structure:

{  
 "timestamp": "2024-01-15T10:30:00",  
 "raw\_signals": {  
 "heart\_rate": 75.2,  
 "spo2": 98.1,  
 "temperature": 36.8,  
 "activity": 32.5  
 },  
 "clarity\_layer": {  
 "processed\_data": {...},  
 "quality\_score": 0.92,  
 "signal\_to\_noise\_ratio": 38.5,  
 "quality\_assessment": "excellent",  
 "quality\_metrics": {...},  
 "artifacts\_detected": [],  
 "noise\_reduction\_applied": false  
 },  
 "ifrs\_layer": {  
 "enhanced\_data": {...},  
 "dominant\_frequency": 1.25,  
 "hrv\_features": {  
 "rmssd": 42.3,  
 "sdnn": 68.1,  
 "pnn50": 28.5,  
 "hrv\_score": 75.2  
 },  
 "rhythm\_classification": "normal\_sinus",  
 "respiratory\_rate": 14.5,  
 "lf\_hf\_ratio": 1.2  
 },  
 "timesystems\_layer": {  
 "pattern\_type": "stable",  
 "temporal\_consistency": 0.88,  
 "circadian\_phase": "afternoon",  
 "circadian\_alignment": 0.92,  
 "rhythm\_score": 82.5,  
 "phase\_shift": 0  
 },  
 "lia\_insights": {  
 "condition": "Normal Resting",  
 "confidence": 0.92,  
 "wellness\_score": 85.3,  
 "probabilities": {...},  
 "recommendation": "...",  
 "wellness\_assessment": {  
 "cardiovascular\_health": 88.5,  
 "respiratory\_health": 92.1,  
 "activity\_level": 78.3,  
 "stress\_level": 82.0,  
 "overall\_wellness": 85.3  
 },  
 "risk\_factors": [],  
 "positive\_indicators": [...]  
 }  
}

### GET /api/v1/logs/processing

Get recent processing logs from all layers

#### Query Parameters:

limit (optional): Number of logs to return (default: 100)

#### Response:

{  
 "total": 100,  
 "logs": [  
 {  
 "timestamp": "2024-01-15T10:30:00.123",  
 "level": "INFO",  
 "message": "CLARITY\_LAYER | quality=0.92 | snr=38.5dB",  
 "data": {...}  
 },  
 ...  
 ]  
}

## 9.3 Response Status Codes

|  |  |
| --- | --- |
| Status Code | Meaning |
| 200 OK | Request successful |
| 201 Created | Resource created successfully |
| 400 Bad Request | Invalid request data |
| 404 Not Found | Resource not found |
| 500 Internal Server Error | Server processing error |
| 503 Service Unavailable | Service temporarily unavailable |

# 10. Data Stream Simulation Process

## 10.1 Overview

The system demonstrates real-time data streaming through a comprehensive simulation that accurately replicates how biosignals are dynamically received, processed through all layers, and delivered to the mobile application.

## 10.2 Simulation Components

### BLE Device Simulation

The BLE simulator generates physiologically realistic biosignal data:

* Continuous generation at 10 Hz (100ms intervals)
* Realistic value ranges for each biosignal type
* Natural variation patterns (e.g., heart rate variability)
* Simulated noise and artifacts for testing Clarity™ layer
* Synchronized signal generation (all signals updated together)
* Configurable parameters for different scenarios (rest, exercise, sleep)

### Signal Generation Details

|  |  |  |  |
| --- | --- | --- | --- |
| Signal | Base Range | Variation | Update Method |
| Heart Rate | 45-180 BPM | ±10% | Sine wave with noise |
| SpO2 | 90-100% | ±2% | Stable with occasional dips |
| Temperature | 35.5-38.5°C | ±0.3°C | Slow drift |
| Activity | 0-150 steps/min | Variable | Random walk |

## 10.3 Real-Time Processing Demonstration

The system demonstrates dynamic signal processing through:

* Live updates every 100ms showing current processing state
* Processing logs displaying layer-by-layer analysis
* Quality metrics updating in real-time as signals vary
* Pattern recognition adapting to changing signal characteristics
* Condition classification changing based on current state
* Wellness scores recalculating with each update

## 10.4 Streaming Methods

### WebSocket Streaming

#### Example WebSocket client code:

const ws = new WebSocket('ws://localhost:8000/ws/stream');  
  
ws.onopen = () => {  
 console.log('Connected to real-time stream');  
};  
  
ws.onmessage = (event) => {  
 const message = JSON.parse(event.data);  
 if (message.type === 'stream\_data') {  
 // Data updated every 100ms  
 updateUI(message.data);  
 }  
};

### HTTP Polling

#### Example polling code:

// Poll for updates every 1 second  
setInterval(async () => {  
 const response = await fetch('http://localhost:8000/api/v1/stream');  
 const data = await response.json();  
 updateUI(data);  
}, 1000);

# 11. Testing & Demonstration

## 11.1 Mock Connection Demonstrations

The system includes comprehensive demonstration features showing how the mobile app communicates with the backend:

### API Demo Screen

* Health Check: Verifies backend is running and all services are operational
* Device Connection: Establishes session and retrieves device status
* Stream Data: Fetches real-time processed biosignal data
* LIA Prediction: Retrieves current health prediction and wellness scores
* Layer Demo: Shows step-by-step processing through all layers
* Processing Logs: Retrieves backend processing activity logs

### Test Execution

Each test displays:

* Request endpoint and parameters
* Response data (full JSON)
* Response time in milliseconds
* Success/failure status with color coding
* Detailed error messages if failures occur

## 11.2 Using Postman

A complete Postman collection is provided with pre-configured requests for all endpoints. The collection includes:

* Environment variables for easy URL configuration
* Automatic session ID extraction and reuse
* Example requests for all endpoints
* Response validation tests
* Organized folder structure by endpoint category

## 11.3 Command Line Testing

#### Example cURL commands:

# Health Check  
curl http://localhost:8000/api/v1/health  
  
# Connect Device  
curl -X POST http://localhost:8000/api/v1/connect \  
 -H "Content-Type: application/json" \  
 -d '{"device\_id":"TEST\_001","device\_type":"mobile\_app"}'  
  
# Get Stream Data  
curl http://localhost:8000/api/v1/stream  
  
# Layer Demonstration  
curl http://localhost:8000/api/v1/demo/layers  
  
# Processing Logs  
curl http://localhost:8000/api/v1/logs/processing?limit=50

## 11.4 Python Demo Client

A comprehensive Python demo client (demo\_client.py) is provided that demonstrates:

* Complete workflow from connection to data retrieval
* Continuous streaming with real-time updates
* Error handling and retry logic
* Formatted console output showing all layer results
* Performance metrics and timing information

# 12. Deployment Guide

## 12.1 Backend Deployment

### Local Development

1. Install Dependencies:  
 cd backend  
 pip install -r requirements.txt  
  
2. Run Server:  
 python main.py  
  
 Or using uvicorn:  
 uvicorn main:app --reload --host 0.0.0.0 --port 8000  
  
3. Access API Documentation:  
 - Swagger UI: http://localhost:8000/docs  
 - ReDoc: http://localhost:8000/redoc

### Docker Deployment

1. Build Docker Image:  
 docker build -t wearable-backend .  
  
2. Run Container:  
 docker run -p 8000:8000 wearable-backend  
  
3. Docker Compose (if using):  
 docker-compose up

## 12.2 Mobile App Deployment

### Development Build

1. Install Dependencies:  
 cd BLE-wearable-App  
 npm install  
  
2. Start Development Server:  
 npx expo start --clear  
  
3. Run on Platform:  
 - iOS Simulator: Press 'i'  
 - Android Emulator: Press 'a'  
 - Physical Device: Scan QR code with Expo Go

### Production Build

1. iOS Build:  
 eas build --platform ios  
  
2. Android Build:  
 eas build --platform android  
  
3. Submit to App Stores:  
 eas submit --platform ios  
 eas submit --platform android

## 12.3 Environment Configuration

Mobile App (.env):  
EXPO\_PUBLIC\_API\_URL=http://localhost:8000  
EXPO\_PUBLIC\_WS\_URL=ws://localhost:8000  
  
Backend (environment variables):  
PORT=8000  
HOST=0.0.0.0  
LOG\_LEVEL=INFO

# 13. Appendix

## 13.1 Performance Metrics

|  |  |
| --- | --- |
| Metric | Value |
| API Response Time (avg) | < 50ms |
| WebSocket Update Rate | 10 Hz (100ms interval) |
| Processing Latency per Layer | < 20ms |
| Total Pipeline Latency | < 50ms |
| Concurrent Clients Supported | 100+ |
| Data Update Frequency | 10 Hz |
| Mobile App Polling Interval | 1 second (configurable) |
| Logs Auto-Refresh Interval | 2 seconds |

## 13.2 File Structure Reference

#### Complete project structure:

wearable-model/  
├── BLE-wearable-App/ # Mobile Application  
│ ├── App.tsx  
│ ├── src/  
│ │ ├── components/  
│ │ ├── screens/  
│ │ │ ├── dashboard/  
│ │ │ ├── livedata/  
│ │ │ ├── logs/  
│ │ │ ├── demo/  
│ │ │ ├── session/  
│ │ │ ├── history/  
│ │ │ └── settings/  
│ │ ├── services/  
│ │ │ ├── backendApi.ts  
│ │ │ ├── bluetooth.ts  
│ │ │ └── openai.ts  
│ │ ├── store/  
│ │ ├── navigation/  
│ │ └── types/  
│ ├── assets/  
│ ├── INTEGRATION\_GUIDE.md  
│ ├── FEATURES\_SUMMARY.md  
│ └── package.json  
│  
└── backend/ # Backend API  
 ├── main.py  
 ├── models/  
 │ └── schemas.py  
 ├── services/  
 │ ├── ble\_simulator.py  
 │ ├── clarity.py  
 │ ├── ifrs.py  
 │ ├── timesystems.py  
 │ ├── lia\_integration.py  
 │ └── session\_manager.py  
 ├── utils/  
 │ └── logger.py  
 ├── README.md  
 ├── TECHNICAL\_DOCUMENTATION.md  
 ├── requirements.txt  
 └── demo\_client.py

## 13.3 Key Dependencies

### Mobile App Dependencies

* react-native: Core framework
* expo: Development platform
* react-navigation: Navigation library
* zustand: State management
* react-native-paper: UI components
* expo-bluetooth: BLE connectivity
* typescript: Type safety

### Backend Dependencies

* fastapi: Web framework
* uvicorn: ASGI server
* pydantic: Data validation
* numpy: Numerical computing
* scipy: Scientific computing
* python-docx: Document generation

## 13.4 Glossary

|  |  |
| --- | --- |
| Term | Definition |
| BLE | Bluetooth Low Energy - wireless communication protocol |
| HRV | Heart Rate Variability - variation in time between heartbeats |
| FFT | Fast Fourier Transform - algorithm for frequency analysis |
| SNR | Signal-to-Noise Ratio - measure of signal quality |
| RMSSD | Root Mean Square of Successive Differences - HRV metric |
| SDNN | Standard Deviation of NN intervals - HRV metric |
| pNN50 | Percentage of successive intervals > 50ms - HRV metric |
| SpO2 | Peripheral oxygen saturation - blood oxygen level |
| LIA | Lifestyle Intelligence Analysis - health insights engine |
| REST | Representational State Transfer - API architecture |
| WebSocket | Protocol for real-time bidirectional communication |

**End of Document**

Generated: 2025-10-19 10:05:15