# Shadow Wellness Platform: Refined Software Architecture Design (C4)

## 1. Introduction

This document presents a refined software architecture for the Shadow Wellness Platform, incorporating principles from the C4 model for software architecture visualization. The C4 model provides a hierarchical approach to diagramming software systems, allowing for progressive decomposition from high-level context down to code-level details. This refinement addresses the need for a more optimal design, improved clarity, and a more integrated representation of security as a pervasive concern rather than a repetitive explicit connection.

Shadow remains a privacy-first, open-source wellness solution designed for solo professionals, emphasizing local data processing, peer-to-peer communication, and user control over personal data, eliminating reliance on cloud services. This architecture design is based on the provided Software Requirements Specification (SRS) and subsequent clarifications regarding device types, communication protocols, data streams, and security considerations.

## 2. Core Architectural Principles and Refinements

The foundational principles of the Shadow platform remain consistent, but their architectural realization is refined:

- **Privacy by Design:** Reaffirmed as the paramount principle, ensuring all data processing, analytics, and machine learning occur exclusively on the user's local devices. This is achieved through an

edge-first processing paradigm and a decentralized P2P network.

- **Offline-First Operation:** The system is inherently designed for full functionality without continuous internet connectivity, with robust mechanisms for data synchronization and recovery.
- **Decentralized Peer-to-Peer Network:** Devices form a resilient mesh network, communicating directly to synchronize data and distribute computational loads, eliminating central points of failure.
- **User Sovereignty:** Users maintain complete control over their data through granular controls and transparent data handling practices.
- **Resource Pooling:** Intelligent distribution of computational and storage tasks across connected devices optimizes resource utilization.

- **Modular Extensibility:** A plugin-based architecture facilitates easy integration of new devices, sensors, and analytical modules.
- **Integrated Security:** Instead of explicit, repetitive security links, security is now an inherent quality of all interactions and components, enforced through secure protocols, authentication, authorization, and data integrity checks at every layer. This aligns with the C4 model's focus on showing *what* components do, with security being an attribute of *how* they do it.

# 3. C4 Model - Level 1: System Context Diagram

At the highest level, the System Context diagram shows the Shadow Wellness Platform as a single system, and how it interacts with its users and any external systems. This provides a high-level overview for a non-technical audience.

## 3.1. Elements:

- **Shadow Wellness Platform (System):** The core software system.
- **Solo Professional (Person):** The primary user of the system, who interacts with it to manage their wellness data.
- **External Sensors (System):** Represents various physical sensors (e.g., heart rate monitors, accelerometers) that provide raw data to the platform.

## 3.2. Relationships:

- **Solo Professional** uses **Shadow Wellness Platform** to manage personal wellness data.
- **Shadow Wellness Platform** collects data from **External Sensors**.

# 4. C4 Model - Level 2: Container Diagram

The Container diagram zooms in on the Shadow Wellness Platform and shows the high-level technology choices and how responsibilities are distributed across them. In the context of Shadow, containers are the deployable units, which are the different device types.

## 4.1. Containers (Device Types):

Each of these containers hosts a full instance of the Shadow Node architecture, adapted to its specific capabilities:

- **Linux Laptop (Raspberry Pi) (Container):** A powerful, always-on local processing hub. It hosts the Shadow application, local data storage, and participates in the P2P network. It can handle more complex analytics and serve as a data aggregation point.
    - **Technology:** Raspberry Pi OS, Python, SQLite/other local database.

- **Android Phone (Container):** A mobile, sensor-rich device. It hosts the Shadow mobile application, collects data from its internal sensors and connected wearables, and participates in the P2P network.
    - **Technology:** Android OS, Kotlin/Java, SQLite.
- **T-display-S3 Dev Board + Sensors (Container):** A lightweight, low-power embedded device primarily focused on continuous sensor data collection. It hosts a stripped-down version of the Shadow agent and communicates with the Android Phone.
    - **Technology:** Embedded OS/Custom Firmware (e.g., ESP-IDF), C/C++.

### 4.2. Relationships (P2P Communication):

- **Linux Laptop** communicates with **Android Phone** via **High-bandwidth P2P (Wi-Fi/Wi-Fi Direct)** for data synchronization and task offloading.
- **Android Phone** communicates with **T-display-S3 Dev Board + Sensors** via **Low-power P2P (BLE)** for sensor data collection.
- All containers communicate with **Other Shadow Nodes** (representing other instances of these containers) via **Secure P2P Protocols** (e.g., TLS 1.3 over Wi-Fi/BLE/Custom UDP/TCP).

## 5. C4 Model - Level 3: Component Diagram

This diagram zooms into a single container (e.g., the Android Phone or Linux Laptop) and shows the components within it. For Shadow, the core components within each

Shadow Node are consistent, though their implementation details may vary. This section describes the logical components within a generic Shadow Node.

### 5.1. Components within a Shadow Node:

- **Data Collector (Component):** Responsible for acquiring raw data from various internal and external sensors and data sources connected to the specific device. It interfaces with device-specific APIs and sensor drivers.
    - **Responsibilities:** Sensor data acquisition, data preprocessing (e.g., filtering, normalization), data validation.
    - **Interfaces:** Provides data streams to the Intelligence Engine and Storage.
- **Intelligence Engine (Component):** The core processing unit for analytics, machine learning, and generating personalized insights from the collected data. It consumes raw or preprocessed data and produces insights and models.
    - **Responsibilities:** Data analysis, machine learning model inference, model training (on capable devices), insight generation.
    - **Interfaces:** Consumes data from Data Collector and Caching, provides insights to UI, stores models/insights to Storage.

- **User Interface (Component):** Provides the user with a means to interact with the Shadow platform, visualize insights, manage data, and configure settings. This could be a mobile app, a desktop application, or a web interface.
    - **Responsibilities:** Data visualization, user input handling, configuration management, alerts display.
    - **Interfaces:** Receives insights from IE, sends commands/preferences to IE, receives alerts from Monitoring.
- **Peer Sync Module (Component):** Manages the decentralized, peer-to-peer communication and synchronization between different Shadow Nodes. It handles device discovery, secure connection establishment, and data/model/state synchronization.
    - **Responsibilities:** Device discovery, secure P2P connection management, data synchronization (delta detection), model synchronization, state synchronization.
    - **Interfaces:** Communicates with Data Collector, Intelligence Engine, Storage, and Monitoring for data exchange and status updates. Communicates directly with other PSM instances on other nodes.
- **Storage (Component):** Handles the persistent storage of raw data, processed insights, and machine learning models on the local device. This is typically a local database.
    - **Responsibilities:** Data persistence, data retrieval, data integrity.
    - **Interfaces:** Receives data from Data Collector and Intelligence Engine, provides data to Intelligence Engine.
- **Caching (Component):** Provides fast, temporary storage for frequently accessed data, optimizing the performance of the Intelligence Engine.
    - **Responsibilities:** Temporary data storage, quick data retrieval.
    - **Interfaces:** Receives data from Data Collector, provides data to Intelligence Engine.
- **Resource Manager (Component):** Orchestrates the dynamic distribution of computational load and resource utilization across the device ecosystem. It monitors local resources and coordinates with other RMs for task offloading.
    - **Responsibilities:** Resource monitoring (CPU, memory, battery), task scheduling, load balancing, task offloading negotiation.
    - **Interfaces:** Interacts with Data Collector, Intelligence Engine, PSM, Storage, and Caching to manage resource allocation. Receives resource metrics from Monitoring.
- **Monitoring (Component):** Collects system health metrics, performance data, and security events for internal diagnostics and user alerts.
    - **Responsibilities:** System health monitoring, performance metric collection, security event logging, anomaly detection.
    - **Interfaces:** Collects data from Data Collector, PSM, and Resource Manager. Provides alerts to UI and security events to the underlying security mechanisms.

## 5.2. Integrated Security (Cross-Cutting Concern):

Instead of a separate

Security Layer component, security is now treated as a pervasive, cross-cutting concern, integrated into the design and operation of every component and interaction. This aligns with the principle of "Security by Design" and avoids redundant explicit connections in diagrams.

**How Security is Integrated:**

- **Authentication & Authorization:** The Peer Sync Module (PSM) is responsible for secure device pairing and authentication (NFR-029). All communication channels (managed by PSM) enforce authorization policies, ensuring only authenticated and authorized devices participate in data synchronization (NFR-002). This includes secure key management and rotation (NFR-032).
- **Encryption (Data in Transit):** All data exchanged between components and across devices (via PSM) is encrypted using industry-standard protocols. Specifically, TLS 1.3 (NFR-031) is used for all network communications, and AES-256 (NFR-001) for data encryption.
- **Encryption (Data at Rest):** The Storage component ensures that all data stored locally on devices is encrypted using AES-256 (NFR-001).
- **Data Integrity:** Cryptographic hashing (e.g., SHA-256) is applied to data at rest and in transit to detect any unauthorized modifications or tampering. This is implicitly handled by the communication protocols and storage mechanisms.
- **Secure Boot & Runtime Integrity:** For embedded devices like the Raspberry Pi and T-display-S3, secure boot mechanisms ensure that only trusted software images are loaded and executed. Runtime integrity checks continuously monitor the system for unauthorized changes.
- **Auditing & Logging:** The Monitoring component collects security-relevant events (NFR-003), such as data access attempts, configuration changes, and failed authentication attempts. These logs are immutable and used for auditing and forensic analysis.
- **Privacy Controls:** The User Interface (UI) provides granular controls (FR-002, FR-003) for users to manage data collection, storage, sharing, and deletion, directly influencing the behavior of the Data Collector and Storage components.

# 6. C4 Model - Level 4: Code Diagram (Illustrative)

While a full code diagram is beyond the scope of this architectural overview, it's useful to illustrate how a specific component, such as the Data Collector, might be structured internally. This level focuses on the implementation details of individual components.

## 6.1. Data Collector (DC) Internal Structure (Example):

- **Sensor Adapters (Module):** A set of plugins or modules, each responsible for interfacing with a specific type of sensor or data source (e.g., HeartRateSensorAdapter,

AccelerometerAdapter, ScreenTimeAdapter). These modules abstract away the sensor-specific communication protocols and data formats.
- **Data Preprocessor (Module):** Handles initial processing of raw sensor data, including filtering, noise reduction, unit conversion, and aggregation. This prepares the data for the Intelligence Engine.
- **Data Queue (Internal Component):** A temporary buffer for incoming data before it is passed to the Intelligence Engine or Storage. This helps manage data flow and handle bursts of data.
- **Configuration Manager (Internal Component):** Reads and applies user-defined settings for data collection, including opt-in/opt-out preferences and collection frequencies.

# 7. Device-Specific Implementations of Shadow Node Components

While the logical components are consistent, their physical implementation and capabilities vary significantly across device types:

## 7.1. Linux Laptop (Raspberry Pi)

- **Data Collector:** Can integrate with system-level APIs for activity monitoring (screen time, keyboard/mouse usage) and potentially act as a gateway for other local network devices. Sensor Adapters would be implemented in Python.
- **Intelligence Engine:** Capable of running more complex ML models (e.g., full TensorFlow/PyTorch models) and performing extensive data analysis due to higher computational resources.
- **Storage:** Can host a more robust local database (e.g., PostgreSQL, SQLite for larger datasets) for long-term data retention and complex queries.
- **User Interface:** Could be a web-based interface served locally or a desktop application.
- **Peer Sync Module:** Primarily uses standard Wi-Fi for high-bandwidth P2P communication.

## 7.2. Android Phone

- **Data Collector:** Leverages Android APIs for accessing internal sensors (GPS, accelerometer, gyroscope, etc.) and integrates with BLE for wearable data. Sensor Adapters would be implemented in Kotlin/Java.
- **Intelligence Engine:** Uses lightweight ML frameworks (e.g., TensorFlow Lite) for on-device inference. More computationally intensive tasks can be offloaded to the Linux laptop via the Resource Manager.
- **Storage:** Typically uses SQLite for local data storage.
- **User Interface:** Native Android application.
- **Peer Sync Module:** Utilizes Wi-Fi Direct for high-bandwidth P2P with laptops and BLE for low-power communication with wearables.

### 7.3. T-display-S3 Dev Board + Sensors

- **Data Collector:** Highly optimized, low-level C/C++ code for direct sensor interfacing and efficient data acquisition. Minimal preprocessing occurs on-device.
- **Intelligence Engine:** Very limited processing capabilities; primarily responsible for basic data aggregation or simple thresholding. Most intelligence is offloaded.
- **Storage:** Minimal, often volatile storage for buffering data before transmission.
- **User Interface:** Limited to basic display on the T-display-S3 screen, primarily for status or simple notifications.
- **Peer Sync Module:** Focuses on efficient, low-power BLE communication, potentially participating in a Bluetooth Mesh network.

## 8. Peer-to-Peer Communication Refinements

The P2P communication is central to Shadow's decentralized nature. The Peer Sync Module (PSM) on each device orchestrates this.

### 8.1. Communication Channels:

- **Wi-Fi/Wi-Fi Direct:** For high-throughput data exchange between laptops and phones. Ideal for large data synchronizations and model transfers.
- **Bluetooth Low Energy (BLE):** The primary channel for communication with low-power wearables. Custom GATT profiles would be defined for specific sensor data.
- **Bluetooth Mesh:** For creating a robust, self-healing network among multiple BLE devices, allowing data to hop across devices to reach a more capable node (e.g., Android phone acting as a gateway).
- **Custom Application Protocol (over UDP/TCP):** A lightweight, secure protocol for control messages, metadata synchronization, and task negotiation between Resource Managers.

### 8.2. Synchronization Mechanisms:

- **Delta Detection:** The PSM employs intelligent delta detection using techniques like change data capture (CDC) and Merkle trees to minimize data transfer during synchronization. Only changed or new data blocks are transmitted.
- **Conflict Resolution:** A decentralized conflict resolution strategy (e.g., last-write-wins with versioning, or user-guided resolution for critical data) is implemented to handle concurrent modifications across devices.

## 9. Conclusion

This refined architecture, guided by C4-UML principles, provides a clearer, more structured view of the Shadow Wellness Platform. By treating security as an integrated, cross-cutting concern

and detailing component interactions within a hierarchical model, the design becomes more robust and easier to understand. The distinction between logical components and their device-specific implementations highlights the platform's adaptability across diverse hardware. This architecture serves as a comprehensive blueprint for the development of a truly privacy-first, decentralized wellness solution.

# 10. References

[1] Simon Brown. The C4 Model for Software Architecture. https://c4model.com/

# 11. Detailed Explanations of Architectural Decisions and Interactions

This section provides in-depth answers to specific questions regarding the Shadow Wellness Platform's software architecture, clarifying design choices and the intricate interactions between its components.

## 11.1. Rationale for Direct Feed of Encrypted Raw Data to Intelligence Engine

The question arises: why is "encrypted raw data" directly fed from the Data Collector (DC) to the Intelligence Engine (IE), especially when there's also a Caching (CCH) module for frequent data and a Storage (STG) module for persistent data? This design choice is driven by several key considerations, primarily focused on **privacy, performance, and real-time processing capabilities**.

**1. Privacy and Security (Edge-First Principle):**

- **Immediate Processing at Source:** The core principle of Shadow is

"Privacy by Design" and "Edge-First Processing." This means data should be processed as close to its source as possible, on the user's device, without unnecessary intermediaries or delays. Directly feeding encrypted raw data to the IE ensures that sensitive information is processed immediately upon collection, minimizing its exposure time in intermediate states or unencrypted forms. The encryption ensures that even during this direct transfer, the data remains protected.

- **Minimizing Data Exposure:** If the raw data were first written to Storage and then read by the IE, it would introduce an additional step where the data resides on disk before processing. While Storage is also encrypted, direct streaming from DC to IE reduces the attack surface and the window of vulnerability. It's a more

"just-in-time" approach to data processing.

**2. Performance and Real-time Processing:**

- **Low Latency for Real-time Insights:** Many wellness applications benefit from real-time or near real-time insights. For instance, stress detection or immediate feedback on activity levels requires rapid processing of incoming sensor data. A direct pipeline from DC to IE minimizes latency by avoiding disk I/O overhead that would be incurred if data always had to be written to Storage first. This allows the IE to perform immediate analysis and generate timely insights.
- **Optimized Data Flow for Stream Processing:** The Data Collector often deals with continuous streams of data (e.g., heart rate, accelerometer readings). The Intelligence Engine is designed to consume these streams efficiently. A direct connection facilitates a stream-processing paradigm, where data is processed as it arrives, rather than in batches after being written to disk. This is particularly efficient for machine learning models that operate on sequential or time-series data.

**3. Role of Caching and Storage in Conjunction with Direct Feed:**

While the direct feed is crucial for immediate processing, Caching (CCH) and Storage (STG) serve complementary, but distinct, purposes:

- **Caching (CCH):** The Caching module is used for **frequently accessed or recently processed data** that the IE might need to retrieve quickly for subsequent calculations or for historical context within a short timeframe. For example, if the IE needs to compare the current heart rate with the average heart rate over the last 5 minutes, that recent historical data might be pulled from the cache. The DC feeds data to CCH for this purpose, and the IE can access CCH for quick lookups. This avoids redundant processing or repeated disk reads for data that is still

"hot" in memory. It's a performance optimization for iterative or context-aware processing.

- **Storage (STG):** The Storage module is for **persistent, long-term storage** of all collected raw data and processed insights. This is essential for historical analysis, trend tracking over longer periods, data export/import, and recovery in case of system failures. While the IE might process data directly from the DC, the DC simultaneously writes this raw data to STG to ensure no data is lost. The IE also writes its generated models and insights to STG for persistence. The IE can also retrieve historical data from STG for batch processing, model retraining, or deeper, less time-sensitive analysis.

**In summary, the direct feed from Data Collector to Intelligence Engine is a design choice that prioritizes immediate, low-latency processing of incoming data for real-time insights, aligning with the privacy-first, edge-processing philosophy. Caching and Storage serve**

**as complementary mechanisms for performance optimization (short-term, frequently accessed data) and data persistence (long-term, historical data), respectively, ensuring both efficiency and data integrity within the decentralized system.**

## 11.2. Task Orchestration and Load Balancing in the Resource Manager (RM)

The Resource Manager (RM) is a critical component in the Shadow architecture, especially given its decentralized and resource-pooled nature. Its primary role is to intelligently manage and optimize the utilization of computational and storage resources across the connected devices (Shadow Nodes). The concepts of "task orchestration" and "load balancing" are central to this function.

**1. Task Orchestration:**

Task orchestration refers to the RM's ability to coordinate and manage the execution of various computational tasks within a Shadow Node and across the entire device ecosystem. It's about deciding *where* and *when* a task should be executed to achieve optimal performance, efficiency, and responsiveness.

- **Why it's needed:** In a multi-device, decentralized system like Shadow, different devices have varying computational capabilities (e.g., a powerful Linux laptop vs. a low-power wearable). Tasks also have different resource requirements (e.g., complex ML model training vs. simple sensor data aggregation). Without orchestration, tasks might be inefficiently executed on underpowered devices, or powerful devices might sit idle.
- **How it works (Interactions with Data Collector and Intelligence Engine):**
    - **From Data Collector (DC):** The DC collects raw data. Some initial processing or filtering might be simple enough to be handled directly by the DC. However, if the DC identifies a need for more complex preprocessing (e.g., advanced signal processing, feature extraction for ML) that is computationally intensive, it can inform the RM. The RM then decides whether this task should be executed locally by the IE or offloaded to another, more capable Shadow Node via the Peer Sync Module (PSM).
    - **From Intelligence Engine (IE):** The IE performs analytics and ML. Tasks like training new ML models, running complex simulations, or performing deep historical data analysis can be very resource-intensive. When the IE initiates such a task, it consults the RM. The RM, based on current resource availability (from Monitoring) and the task's requirements, orchestrates its execution. This might involve:
        - **Local Execution:** If the current device has sufficient resources.
        - **Offloading:** If the local device is resource-constrained (e.g., low battery on a phone), the RM can instruct the PSM to send the task to another available Shadow Node (e.g., the Linux laptop).

        - **Distributed Execution:** For very large tasks, the RM might break them down and distribute parts across multiple nodes.
- **Role of PSM:** The PSM is the communication backbone for orchestration. When the RM decides to offload a task, it uses the PSM to communicate with the RM on the target device, transfer the task data, and receive results.

**2. Load Balancing:**

Load balancing is a specific aspect of task orchestration, focusing on distributing computational or storage workloads evenly across available resources to prevent any single device from becoming a bottleneck and to maximize overall system throughput and responsiveness.

- **Why it's needed:** Without load balancing, one device might be overloaded while others are underutilized, leading to poor performance, increased battery drain, or even system crashes on the overloaded device.
- **How it works (Interactions with Storage and Caching):**
    - **Storage (STG) Load Balancing:** The RM manages how data is written to and read from the Storage module. In a decentralized system, data might be replicated across multiple nodes for redundancy and availability. The RM can direct write operations to the least busy storage instance or read operations from the closest available replica. It can also manage data migration between storage locations to balance disk usage.
    - **Caching (CCH) Optimization:** The RM optimizes cache utilization. It can decide which data should be cached on which device based on access patterns and available cache memory. For instance, if a particular type of data is frequently accessed by the IE on a specific device, the RM ensures that data is prioritized for caching on that device. It can also manage cache eviction policies to ensure the most relevant data remains in the cache.
- **Interactions with Monitoring (MON):** The Monitoring module is crucial for the RM's decision-making. MON continuously collects real-time metrics on CPU usage, memory availability, battery levels, network latency, and storage I/O from all components and devices. This

real-time resource utilization data is fed to the RM, enabling it to make informed decisions about task orchestration and load balancing. For example, if MON reports that a phone's battery is critically low, the RM will prioritize offloading tasks from that phone to a laptop.

**In essence, the Resource Manager acts as the distributed brain of the Shadow ecosystem, ensuring that computational tasks are executed efficiently and resources are optimally utilized across all connected devices, adapting dynamically to changing conditions and device capabilities.**

## 11.3. Detailed Interactions Between Modules

This section provides a comprehensive breakdown of the interactions between each module within a Shadow Node, elaborating on the data flows, control signals, and the purpose of each connection. It also implicitly highlights how security, as a cross-cutting concern, underpins these interactions.

### 11.3.1. Data Collector (DC) Interactions

The Data Collector is the entry point for all raw data into the Shadow system. It interfaces directly with physical sensors and device-specific APIs.

- **DC --> Intelligence Engine (IE):**
    - **Data Flow:** Encrypted raw data (e.g., sensor readings, activity logs).
    - **Purpose:** To provide immediate, low-latency data for real-time processing, analytics, and machine learning inference. This direct feed minimizes delay for time-sensitive insights.
    - **Security:** Data is encrypted at the source (DC) before transmission to the IE, ensuring confidentiality during processing.
- **DC --> Storage (STG):**
    - **Data Flow:** Raw collected data (encrypted).
    - **Purpose:** To persistently store all raw data for historical analysis, long-term trend tracking, data export, and system recovery. This ensures data integrity and availability over time.
    - **Security:** Data is encrypted before being written to persistent storage, protecting data at rest.
- **DC --> Caching (CCH):**
    - **Data Flow:** Frequently accessed or recently collected data (encrypted).
    - **Purpose:** To provide a fast, temporary repository for data that the IE or other modules might need to access quickly and repeatedly. This optimizes performance by reducing the need for repeated disk reads from STG.
    - **Security:** Data in the cache is also encrypted, maintaining confidentiality even in temporary storage.
- **DC --> Monitoring (MON):**
    - **Data Flow:** Metrics related to data collection (e.g., data volume, collection frequency, sensor status, errors).
    - **Purpose:** To provide the Monitoring module with insights into the health and performance of data acquisition processes. This allows for detection of issues like sensor failures or data bottlenecks.
    - **Security:** These are operational metrics, not sensitive user data, but their integrity is important for system health.
- **External Sensors --> DC:**
    - **Data Flow:** Raw sensor data (e.g., heart rate, steps, screen time).

    - **Purpose:** The primary input of raw data from the physical world into the digital system. The DC abstracts the complexities of different sensor interfaces.
    - **Security:** Data is collected from trusted sources; the DC immediately encrypts it upon ingestion.

### 11.3.2. Intelligence Engine (IE) Interactions

The Intelligence Engine is the analytical core, transforming raw data into meaningful insights.

- **IE <-- Data Collector (DC):** (See DC --> IE above)
- **IE --> User Interface (UI):**
    - **Data Flow:** Insights, wellness recommendations, analytical visualizations, processed data summaries.
    - **Purpose:** To present actionable information and personalized feedback to the user, enabling them to understand their wellness patterns and make informed decisions.
    - **Security:** Insights are derived from sensitive data; access is controlled by the UI's authentication and authorization mechanisms.
- **IE --> Storage (STG):**
    - **Data Flow:** Trained machine learning models, derived insights, aggregated data summaries.
    - **Purpose:** To persist the valuable outputs of the IE, allowing models to be reused, insights to be reviewed historically, and aggregated data to be available for future analysis or synchronization.
    - **Security:** Models and insights are encrypted before storage.
- **IE <-- User Interface (UI):**
    - **Data Flow:** User commands, preferences, configuration changes (e.g., setting wellness goals, requesting specific reports, adjusting privacy settings).
    - **Purpose:** To allow users to interactively control the IE's behavior, customize their experience, and trigger specific analytical processes.
    - **Security:** User commands are authenticated and authorized by the UI before being passed to the IE.
- **IE <-- Caching (CCH):**
    - **Data Flow:** Frequently accessed or recently processed data.
    - **Purpose:** To retrieve data quickly for iterative calculations, context-aware analysis, or to combine with new incoming data without incurring disk I/O overhead.
    - **Security:** Data retrieved from cache remains encrypted until used by the IE.
- **IE <-- Peer Sync Module (PSM):**
    - **Data Flow:** Synchronized machine learning models, model updates, configuration parameters from other Shadow Nodes.

    - **Purpose:** To ensure that all connected Shadow Nodes have consistent analytical capabilities and are using the latest versions of shared models, facilitating distributed intelligence.
    - **Security:** Data is securely synchronized via encrypted P2P channels, authenticated by PSM.
- **IE <-- Resource Manager (RM):**
    - **Control Signal:** Task orchestration commands (e.g.,

"start ML training," "process historical data"). * **Purpose:** The RM can initiate or direct the IE to perform specific tasks, especially those that are computationally intensive or require coordination with other nodes for resource pooling. * **Security:** Control signals are authenticated and authorized by the RM.

### 11.3.3. User Interface (UI) Interactions

The User Interface is the primary point of interaction for the human user.

- **UI <-- Intelligence Engine (IE):** (See IE --> UI above)
- **UI --> Intelligence Engine (IE):** (See IE <-- UI above)
- **UI <-- Monitoring (MON):**
    - **Data Flow:** Alerts, notifications (e.g., low battery on a connected device, data synchronization issues, security warnings, health anomalies).
    - **Purpose:** To inform the user about the system's operational status, potential issues, or important wellness-related events that require their attention.
    - **Security:** Alerts related to security events are prioritized and clearly communicated.

### 11.3.4. Peer Sync Module (PSM) Interactions

The PSM is the backbone of the decentralized, peer-to-peer communication.

- **PSM --> Data Collector (DC):**
    - **Data Flow:** Securely synchronized data from other Shadow Nodes.
    - **Purpose:** To ingest data collected by other devices into the local node's processing pipeline, ensuring a unified view of the user's wellness data across all connected devices.
    - **Security:** All data transferred is encrypted (AES-256) and authenticated (TLS 1.3), as per the SRS.
- **PSM --> Intelligence Engine (IE):** (See IE <-- PSM above)
- **PSM --> Storage (STG):**
    - **Data Flow:** Synchronized state information, metadata, and potentially raw data blocks from other Shadow Nodes.
    - **Purpose:** To maintain consistency of the distributed system's state across all nodes and to ensure data redundancy and availability. This includes

synchronization of configuration settings, user preferences, and data integrity checks.
    - **Security:** State and metadata are encrypted and authenticated during transfer.
- **PSM --> Monitoring (MON):**
    - **Data Flow:** Network health metrics (e.g., connection status, latency, bandwidth usage of P2P links), P2P communication status, synchronization progress.
    - **Purpose:** To provide the Monitoring module with real-time information about the health and performance of the decentralized network, enabling detection of connectivity issues or synchronization bottlenecks.
    - **Security:** Operational metrics, but important for diagnosing security-related network anomalies.
- **PSM <--> PSM (Other Shadow Nodes):**
    - **Data Flow:** Authenticated P2P communication (encrypted data, models, state, control messages).
    - **Purpose:** This represents the direct, secure communication channel between different Shadow Nodes. It's the core mechanism for data exchange, task offloading, and maintaining the mesh network.
    - **Security:** This interaction is heavily secured with TLS 1.3 for transport encryption, AES-256 for data encryption, and robust authentication mechanisms (secure device pairing, key management) to ensure only trusted devices can communicate.

#### 11.3.5. Storage (STG) Interactions

The Storage module is responsible for data persistence.

- **STG <-- Data Collector (DC):** (See DC --> STG above)
- **STG <-- Intelligence Engine (IE):** (See IE --> STG above)
- **STG <-- Peer Sync Module (PSM):** (See PSM --> STG above)
- **STG <-- Resource Manager (RM):**
    - **Control Signal:** Load balancing commands (e.g., "write data to this replica," "read from this partition," "migrate data").
    - **Purpose:** The RM directs storage operations to optimize performance, ensure data redundancy, and balance disk usage across available storage resources (potentially distributed across multiple nodes).
    - **Security:** Control signals are authenticated and authorized.
- **STG --> Intelligence Engine (IE):**
    - **Data Flow:** Historical raw data, historical insights, stored models.
    - **Purpose:** To provide the IE with access to long-term data for batch processing, model retraining, trend analysis, or to retrieve previously generated insights.
    - **Security:** Data is retrieved in encrypted form and decrypted by the IE for processing.

### 11.3.6. Caching (CCH) Interactions

The Caching module provides fast, temporary data access.

- **CCH <-- Data Collector (DC):** (See DC --> CCH above)
- **CCH --> Intelligence Engine (IE):** (See CCH --> IE above)
- **CCH <-- Resource Manager (RM):**
    - **Control Signal:** Optimization commands (e.g., "cache this data," "clear cache for this type," "adjust cache size").
    - **Purpose:** The RM optimizes cache utilization based on access patterns and system load, ensuring that frequently needed data is readily available to the IE.
    - **Security:** Control signals are authenticated and authorized.

### 11.3.7. Resource Manager (RM) Interactions

The RM is the orchestrator of resources and tasks.

- **RM --> Data Collector (DC):** (See RM --> DC in Task Orchestration explanation)
- **RM --> Intelligence Engine (IE):** (See RM --> IE in Task Orchestration explanation)
- **RM --> Peer Sync Module (PSM):** (See RM --> PSM in Task Orchestration explanation)
- **RM --> Storage (STG):** (See STG <-- RM above)
- **RM --> Caching (CCH):** (See CCH <-- RM above)
- **RM <-- Monitoring (MON):**
    - **Data Flow:** Resource utilization metrics (CPU, memory, battery, network bandwidth), system health data, performance bottlenecks.
    - **Purpose:** To provide the RM with real-time insights into the current state of local and remote resources, enabling it to make informed decisions about task orchestration and load balancing.
    - **Security:** Operational metrics, but their integrity is important for reliable resource management.

### 11.3.8. Monitoring (MON) Interactions

The Monitoring module keeps track of system health and events.

- **MON <-- Data Collector (DC):** (See DC --> MON above)
- **MON <-- Peer Sync Module (PSM):** (See PSM --> MON above)
- **MON --> Resource Manager (RM):** (See RM <-- MON above)
- **MON --> User Interface (UI):** (See UI <-- MON above)
- **MON --> Security Layer (Implicit):**
    - **Data Flow:** Security events, anomalies, potential threats.
    - **Purpose:** While not a direct explicit connection in the diagram (as security is cross-cutting), the Monitoring module is responsible for detecting and logging security-relevant events. These events are implicitly fed into the underlying

security mechanisms (e.g., logging to an immutable audit log, triggering alerts to the Security Layer for further action or reporting).
    - **Security:** This interaction is critical for maintaining the security posture of the system by providing visibility into potential breaches or vulnerabilities.

This detailed breakdown illustrates the complex yet cohesive interplay between the various modules, all operating under the umbrella of integrated security, to deliver the Shadow Wellness Platform's core functionalities.

# 12. Hardware Architecture

To complement the software architecture, understanding the underlying hardware is crucial. The Shadow Wellness Platform is designed to run on a distributed set of personal devices, leveraging their unique capabilities. This section outlines the typical hardware components for each device type.

## 12.1. Linux Laptop (Raspberry Pi)

- **Core Component:** Raspberry Pi (e.g., Raspberry Pi 4 Model B or newer)
    - **Processor:** Quad-core ARM Cortex-A72 (or similar)
    - **RAM:** 4GB or 8GB LPDDR4
    - **Storage:** MicroSD card (for OS and application) or external SSD (for larger data storage)
- **Connectivity:**
    - **Wi-Fi:** Dual-band (2.4 GHz and 5.0 GHz) IEEE 802.11ac wireless
    - **Bluetooth:** Bluetooth 5.0, BLE
    - **Ethernet:** Gigabit Ethernet (for stable network connection)
    - **USB:** USB 3.0 and USB 2.0 ports (for peripherals, external storage)
- **Power:** USB-C power supply
- **Optional Peripherals:**
    - **Display:** HDMI output for monitor
    - **Input:** Keyboard and mouse
    - **Case:** Protective enclosure

## 12.2. Android Phone

- **Core Component:** Standard Android Smartphone (e.g., Google Pixel, Samsung Galaxy, etc.)
    - **Processor:** ARM-based SoC (e.g., Qualcomm Snapdragon, Google Tensor, MediaTek Dimensity)
    - **RAM:** 6GB - 12GB LPDDR4X/LPDDR5
    - **Storage:** Internal UFS storage (128GB - 512GB)
- **Connectivity:**

    - **Wi-Fi:** Wi-Fi 6/6E (802.11ax) with Wi-Fi Direct support
    - **Bluetooth:** Bluetooth 5.0/5.1/5.2, BLE
    - **Cellular:** 4G LTE / 5G (for internet access, though not required for core Shadow functionality)
    - **USB:** USB-C (for charging and data transfer)
- **Sensors (Internal):**
    - **Accelerometer, Gyroscope, Magnetometer:** For motion tracking, orientation.
    - **GPS/GNSS:** For location tracking (opt-in).
    - **Ambient Light Sensor, Proximity Sensor:** For environmental context.
    - **Microphone:** For ambient sound analysis (opt-in).
    - **Camera:** For image/video analysis (opt-in, highly sensitive).
- **Power:** Integrated Li-ion battery

## 12.3. T-display-S3 Dev Board + Other Sensors

- **Core Component:** LilyGo T-Display-S3 (or similar ESP32-S3 based development board)
    - **Microcontroller:** ESP32-S3 (Dual-core Xtensa LX7 processor)
    - **Flash Memory:** 8MB - 16MB (for firmware and limited data storage)
    - **SRAM:** 512KB (internal), potentially PSRAM (external) for larger buffers
- **Display:** Integrated 1.9-inch LCD display
- **Connectivity:**
    - **Wi-Fi:** 2.4 GHz Wi-Fi (802.11 b/g/n)
    - **Bluetooth:** Bluetooth 5, BLE
- **Integrated Sensors (on-board or external via GPIO):**
    - **Accelerometer/Gyroscope:** (e.g., MPU6050, BMI160) for activity tracking.
    - **Heart Rate Sensor:** (e.g., MAX30102) for pulse oximetry and heart rate.
    - **Temperature/Humidity Sensor:** (e.g., DHT11, BME280) for environmental data.
    - **Galvanic Skin Response (GSR) Sensor:** For stress level indicators.
    - **Other custom sensors:** Connected via GPIO, I2C, SPI, UART.
- **Power:** USB-C (for programming and power), Li-Po battery connector (for portable use)

## 12.4. Hardware Architecture Diagram

This conceptual diagram illustrates the interaction between the different hardware components.