Initial Architecture Document

HUD-Link: AR Health Display for Meta Quest Pro
Team 10: Asa Maker, Brandon Dodge, Josh Dwoskin, John Gott, Zach Sevart
Project: HUD-Link
Due Date: February 8, 2026

Project Synopsis:
HUD-Link is a modular, privacy-first AR health display system designed to explore how real-time personal metrics could be presented in lightweight AR eyewear. The Meta Quest Pro is used as a high-fidelity prototyping and simulation platform to model interaction patterns, data pipelines, and comfort constraints that would apply to form factors such as Meta Ray-Ban smart glasses. This approach allows the system to be evaluated under controlled conditions while remaining forward-compatible with lower-power, glasses-based AR devices.

1. System Architecture

1.1 Architectural Overview

While HUD-Link is implemented using the Meta Quest Pro, the headset is not treated as the final target device. Instead, it functions as a **development surrogate** that enables rapid iteration, instrumentation, and visualization of concepts intended for lightweight AR glasses. Architectural decisions deliberately avoid headset-specific assumptions and favor constraints consistent with glasses-based systems, such as minimal rendering load, limited interaction surfaces, and strong privacy boundaries.

1.2 Mobile Data Core (Android Application)

The Android application is the primary execution environment and acts as the system's data engine. Its responsibilities include:

- Health and sensor data ingestion
- Explicit permission and consent enforcement
- Data normalization and serialization
- Local-only data storage and lifecycle management

Rather than transmitting raw sensor streams, the application derives simplified metrics (such as heart rate, speed, and heading) before transmission. This approach reduces bandwidth usage, improves predictability, and aligns with privacy-first requirements.

During Sprint 1 and Sprint 2, the mobile application operates independently of any headset UI. Validation occurs through logging, mock consumers, and debugging tools rather than XR rendering.

1.3 Transport Layer (Bluetooth Low Energy)

Communication between the smartphone and headset is abstracted behind a transport layer designed for low latency and low power consumption. Bluetooth Low Energy (BLE) is selected due to its suitability for short-range telemetry and minimal infrastructure requirements.

The transport layer follows an event-driven model in which the phone pushes updates only when data changes. Payloads are structured, versioned, and compact to ensure predictable delivery and efficient recovery from disconnections.

1.4 XR Client (Unity on Meta Quest Pro)

The XR client running on Meta Quest Pro serves as a **simulation environment** for glasses-style AR interaction rather than a fully immersive VR experience. Unity is used to prototype HUD placement, update cadence, and perceptual comfort in a way that approximates how information might appear in transparent or semi-transparent eyewear displays.

All rendering choices prioritize simplicity, peripheral placement, and low cognitive load, mirroring the constraints expected in smart glasses rather than exploiting the full graphical capabilities of a VR headset.

1.5 Widget System

The HUD interface is built around a modular widget framework designed for extensibility and incremental feature growth. Each widget adheres to shared design rules, enabling independent rendering, consistent layout behavior, and predictable interaction patterns.

Widgets are positioned using a grid-based layout that preserves a central “safe zone” in the user’s field of view. This ensures situational awareness while keeping information accessible in peripheral vision.

1.6 Performance & Comfort Strategy

Performance targets and comfort safeguards are intentionally aligned with **glasses-class hardware limitations**, ensuring that insights gained from the Quest Pro prototype remain transferable to future wearable AR form factors.

2. Required Illustrations

> **Note:** All figures listed below have already been created. The descriptions are retained for reference and alignment with the architecture.

2.1 Figure 1: High-Level System Context Diagram

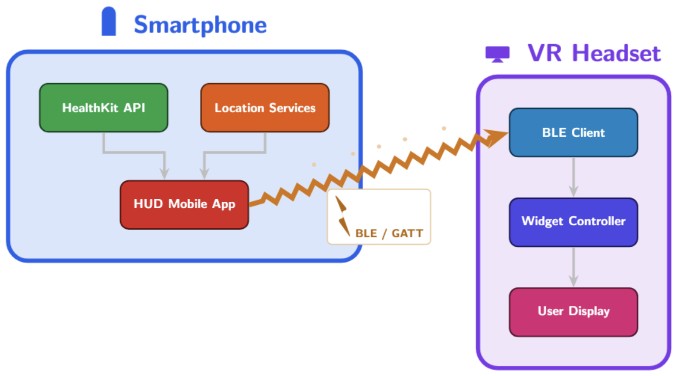
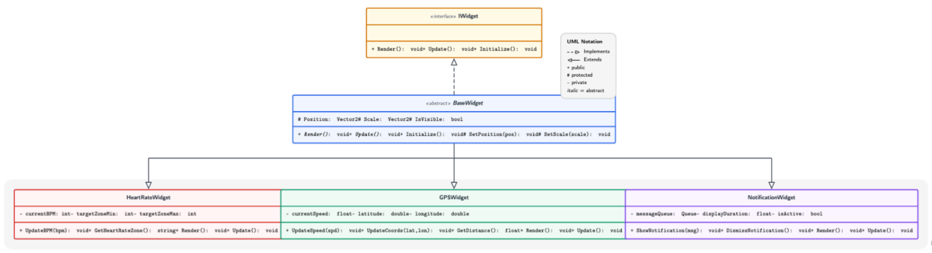


Figure 1: The Hub-and-Spoke architecture separates data aggregation (Phone) from visualization (Headset).

Description: A block diagram showing the Smartphone (HealthKit/Location) connected via BLE to the VR Headset (Client/Controller).
Caption: Figure 1: The hub-and-spoke architecture separates data aggregation (phone) from visualization (headset).

2.2 Figure 2: Widget Class Hierarchy (UML)



«interface» IWidget
+ Render(): void+ Update(): void+ Initialize(): void
UML Notation
Implements
Extends
+ public
# protected
- private
italic = abstract
«abstract» BaseWidget
# Position: Vector2# Scale: Vector2# IsVisible: bool
+ Render(): void+ Update(): void+ Initialize(): void# SetPosition(pos): void# SetScale(scale): void
HeartRateWidget
- currentBPM: int- targetZoneMin: int- targetZoneMax: int
+ UpdateBPM(bpm): void+ GetHeartRateZone(): string+ Render(): void+ Update(): void
GPSWidget
- currentSpeed: float- latitude: double- longitude: double
+ UpdateSpeed(spd): void+ UpdateCoords(lat,lon): void+ GetDistance(): float+ Render(): void+ Update(): void
NotificationWidget
- messageQueue: Queue- displayDuration: float- isActive: bool
+ ShowNotification(msg): void+ DismissNotification(): void+ Render(): void+ Update(): void

Figure 2: The polymorphic widget system allowing for modular extension and independent rendering logic.

Description: A UML diagram showing the IWidget interface and BaseWidget abstract class inherited by HeartRateWidget, GPSWidget, and NotificationWidget.
Caption: Figure 2: The polymorphic widget system enables modular extension and independent rendering logic.

2.3 Figure 3: Data Flow Sequence Diagram

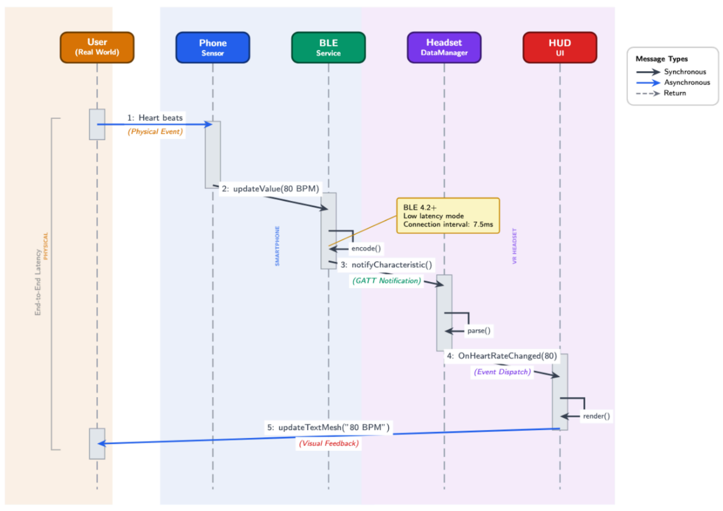


Figure 3: Event-driven data flow from physical sensor to virtual display, minimizing latency.

Description: Sequence showing the flow from User → Phone Sensor → BLE Service → Headset DataManager → HUD UI → User (Visual).
Caption: Figure 3: Event-driven data flow from physical sensors to virtual display minimizes latency.

2.4 Figure 4: HUD UI Layout (Wireframe)

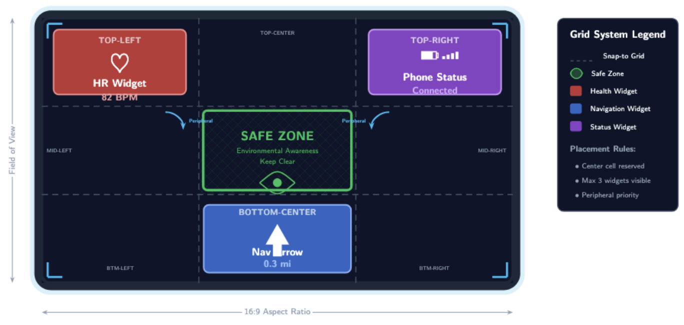


Figure 4: HUD Grid Layout Strategy. The 3x3 anchor system ensures widgets remain in peripheral vision, leaving the central "Safe Zone" clear for environmental awareness.

Description: A 16:9 viewport showing a 3×3 grid with a central clear “Safe Zone” and peripheral widgets such as heart rate, phone status, and navigation.
Caption: Figure 4: The grid-based HUD layout preserves central vision while supporting peripheral awareness.

2.5 Figure 5: Connection State Machine

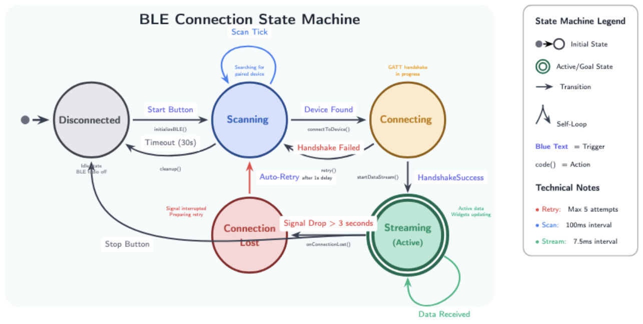


Figure 5: BLE Connection State Machine. The automated reconnection loop ensures robust data streaming without requiring user intervention during physical activity.

Description: State diagram showing transitions between Disconnected, Scanning, Connecting, and Streaming states with automated retry behavior.
Caption: Figure 5: The BLE connection state machine ensures resilient communication and automatic recovery.

3. Implementation Notes

3.1 Technology Stack

- **Headset:** Meta Quest Pro (Unity 2022.3 LTS, OpenXR)
- **Mobile:** Android (Kotlin, Android Studio)
- **Protocol:** Bluetooth Low Energy (GATT)
- **Data Format:** Binary, versioned serialization
- **Architecture Pattern:** Layered modular design

3.2 Key Design Decisions

- **Android-first execution:** Reduces early integration risk and allows isolated testing.
- **Deferred HUD rendering:** Prevents premature performance assumptions.
- **Local-only data handling:** Aligns with privacy and consent requirements.
- **Event-driven transport:** Minimizes battery and CPU usage.

3.3 Sprint Alignment

The HUD-Link project follows a staged sprint plan designed to reduce technical risk by validating foundational components before introducing XR rendering, user interface complexity, and experimental features. Each sprint incrementally builds on previously validated functionality.

Sprint 1 – Architecture & Feasibility Research

Focuses on reducing early uncertainty and selecting an optimal technical approach.

- Platform and architecture research
- SDK and toolchain validation for Android, Unity, and Meta Quest Pro
- End-to-end data flow design
- Security and privacy baseline analysis

Aligned Requirements: R1–R5

Sprint 2 – Android Application Foundation

Establishes the core mobile execution environment without headset rendering.

- Android project skeleton and module structure
- Permission and consent framework
- Health and sensor data ingestion
- Structured data modeling and serialization
- Local-only data handling

Aligned Requirements: R6–R10

Sprint 3 – Data Pipeline & Unity Bridge Validation

Introduces controlled headset integration without full HUD functionality.

- Unity project setup and Android-to-Unity data bridge
- Connection reliability and recovery handling

- Latency buffering and smoothing
- Telemetry and debug instrumentation
- Baseline performance profiling

Aligned Requirements: R11–R15

Sprint 4 – HUD Core & Widget Framework

Implements the first in-headset visual elements and interaction model.

- Minimal HUD overlay with toggle support
- Core widget framework and layout system
- Heart rate widget rendering
- Session persistence of layout and settings
- HUD performance optimization

Aligned Requirements: R16–R20

Sprint 5 – Feature Expansion & Stability

Builds additional user-facing functionality while improving robustness.

- Extended health metrics integration
- GPS-based widgets and motion-aware behavior
- Safety mode and distraction reduction
- Unified error handling and recovery flows
- Cross-device compatibility validation

Aligned Requirements: R21–R25

Sprint 6 – Experimental Features & Finalization

Explores optional social features and prepares the system for final delivery.

- Proximity discovery and consent gating (experimental)
- Shared stats rendering with privacy controls
- Ethical and privacy evaluation of social AR features
- System refactoring and architectural cleanup
- Final integration, documentation, and demonstration

Aligned Requirements: R26–R32

4. Conclusion

HUD-Link leverages the Meta Quest Pro as a **prototyping and simulation platform** to investigate how real-time health and contextual data could be safely and comfortably presented in smart glasses. By treating the Quest Pro as a stand-in rather than a final deployment target, the project avoids overfitting to VR-specific capabilities and instead emphasizes portability, privacy, and user comfort.

This architecture-first, staged approach allows the team to validate data pipelines, consent models, and HUD interaction patterns under realistic constraints while maintaining forward compatibility with lighter, glasses-based AR devices such as Meta Ray-Ban smart glasses. As a result, HUD-Link provides both a functional prototype and a transferable design framework for future wearable AR systems.