SPARTA: Sensor Platform for Analysis, Response, and Threat Awareness

Dan Dowlin · Chantelle Faria · Ishan Patel · James Weggel · Andrew Lin · Chara Kakraba-Ampeh

Project

Wearable devices worn by first responders and defense personnel collect critical health data, but it stays trapped in individual devices with no unified view for administrators.

Two Six Technologies' SIGMA detects CBNRE threats at scale, but has no visibility into the physiological state of the operators responding to those threats.

No commercial tool aggregates multi-device biometric streams, maps them geospatially, and exposes an API compatible with an existing threat-detection ecosystem like SIGMA.

Solution

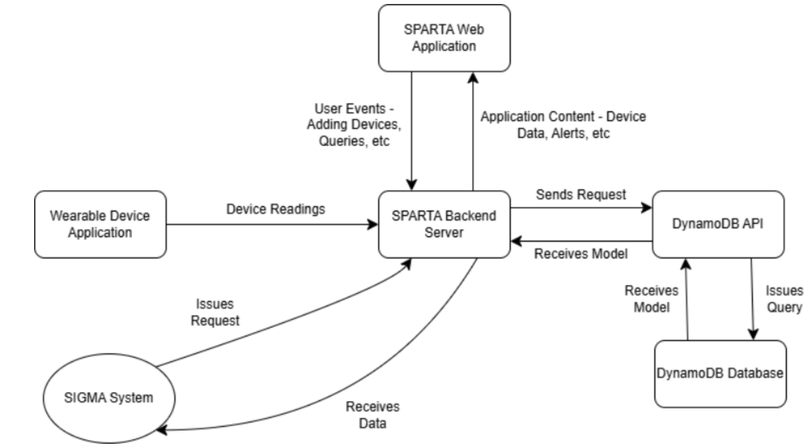
A cloud-based platform that aggregates biometric data from wearables into a centralized, real-time map-based dashboard with automated safety alerts for both administrators and field operators.

In the case of a critical threat, first responders' lives can be tracked and saved before it is too late.

A rollout of SPARTA could protect the lives of hundreds of first responders and defense personnel.

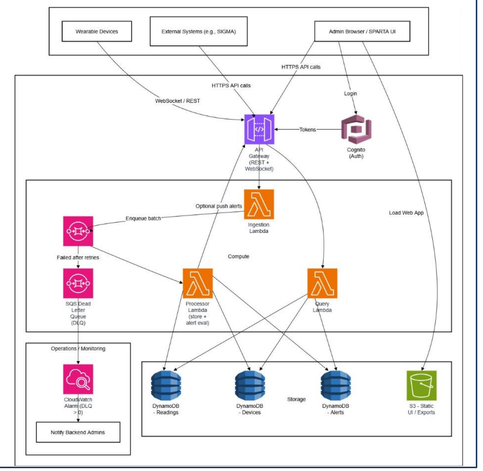
Metrics tracked and monitored include heart rate, respiration rate oxygen saturation, skin temperature, location, speed, stress level, and others.

System Overview Diagram



System Architecture Diagram

- Authentication & Routing: API Gateway uses Cognito to authenticate users and routes requests to the appropriate Lambda.
- Ingestion Lambda: Validates incoming sensor data and device registration payloads from various sources.
- Query Lambda: Handles frontend data requests
- SQS Queue: Decouples ingestion from processing, enabling independent scaling.
- Processor Lambda: Consumes SQS messages, checks thresholds, and writes to DynamoDB.
- DynamoDB Storage: Devices DDB stores thresholds, Readings and Alerts DDB stores sensor data.
- Resilience (DLQ): Failed messages route to a Dead Letter Queue, monitored to alert administrators.
- S3 Bucket: Hosts static web files.



Website Features

- **Map Page:** Shows latest location of each device
- **Device Page:** Manage and add device information and thresholds
- **Readings Page:** View and sort device readings and metrics
- **Alerts Page:** View and sort anomalous device readings
- **Alert Panel:** Notifies administrator when an anomalous device reading is detected

Tech Stack

- **Backend:** Java 17 · AWS Lambda (serverless) · Gradle
- **Frontend:** React + Vite · CSS / HTML
- **Cloud (AWS):** Lambda · API Gateway · DynamoDB · SQS · Cognito
- **Wearable:** Garmin smartwatches · Connect IQ · Monkey C · Bluetooth / Wi-Fi / Cellular

Tools Used

- **Cloud & Backend:** AWS (Lambda, API Gateway, DynamoDB, SQS) for scalable architecture.
- **Dev & Collab:** GitLab for version control · Jira for task tracking.
- **Frontend & Viz:** Web dashboard (HTML / CSS / JS) · Google Maps for real-time location.
- **Wearable:** Garmin SDK (Connect IQ) for smartwatch application.

Testing Strategies

- **Unit:** Individual components — data ingestion, APIs, wearable functions.
- **Integration:** End-to-end flow: Wearable → Backend → Dashboard.
- **Performance:** Multiple devices and continuous data streams.
- **Security:** Encryption in transit / at rest · HIPAA & GDPR compliance.
- **Edge cases:** Device offline, network delays, invalid / corrupted data.

Challenges

- **Wearable limits:** Garmin watches have limited memory and processing power; transmission depends on Wi-Fi / Bluetooth reliability.
- **Transmission:** Continuous real-time streaming was not viable — required batch readings instead.
- **Integration:** Manufacturer APIs limited raw sensor access; SIGMA-compatible API surface required.
- **Learning curve:** Ramp-up on AWS services (Lambda, API Gateway, DynamoDB, SQS) and cloud architecture.
- **System complexity:** Real-time ingestion, processing, and visualization handled simultaneously.