ARCHITECTURAL DESIGN DOCUMENT FOR NEUROGUARD

1. INTRODUCTION

1.1 Overview

The Architectural Design Document (ADD) for NeuroGuard outlines the structural framework for an AI-powered wearable health monitoring system, designed as a handband or headband. This system continuously tracks vital signs, neural signals, and movements to predict and detect epilepsy seizures 1-5 minutes in advance, delivering real-time alerts via a mobile app, cloud dashboard, device alarms, or emergency SMS/calls. The architecture is derived from project requirements, emphasizing a modular, scalable, and secure design to support future expansions (e.g., stroke risk, heart arrhythmias, sleep apnea).
The design follows a hybrid edge-cloud model, integrating hardware sensors, edge processing, cloud-based AI, and user interfaces. It ensures low-latency performance, compliance with health regulations (e.g., HIPAA, GDPR), and adaptability for additional health monitoring features using the same hardware.

1.2 Architectural Goals

- Scalability to support 10,000+ users with cloud auto-scaling.
- Reliability with a Mean Time Between Failures (MTBF) exceeding 1000 hours.
- Security through end-to-end encryption and role-based access control (RBAC).
- Maintainability with modular components and 80% test coverage.
- Performance with data processing latency below 1 second.
- Extensibility for future health monitoring modules without hardware modifications.

1.3 Scope

This ADD addresses the architectural design of the NeuroGuard wearable, firmware, backend services, and frontend interfaces. It excludes detailed hardware manufacturing, third-party API integrations beyond specified dependencies, and immediate implementation of future expansions, though the design accommodates these enhancements.

1.4 References

- Project Requirements Document, Version 3.0, dated 2025-09-26.
- User Flow Diagrams: Visual representations of data flow and user interactions.
- Standards: IEEE Std 1471-2000 (Recommended Practice for Architectural Description), ISO/IEC 42010 (Systems and Software Engineering - Architecture Description), HIPAA, GDPR.
- Industry Practices: AWS Well-Architected Framework, Microsoft Azure IoT Architecture.

1.5 Definitions, Acronyms, and Abbreviations

- **AI/ML:** Artificial Intelligence/Machine Learning – Technologies for predictive analytics.
- **EEG/EMG:** Electroencephalography/Electromyography – Neural signal measurements.
- **HR/SpO2:** Heart Rate/Blood Oxygen Saturation.
- **IoT:** Internet of Things – Network of interconnected devices.
- **RBAC:** Role-Based Access Control – Security model for user permissions.
- **OTA:** Over-The-Air – Method for firmware updates.
- Additional terms in Appendix A: Glossary.

2. HIGH-LEVEL ARCHITECTURE

2.1 System Overview

NeuroGuard employs a hybrid edge-cloud architecture to optimize real-time processing and scalability. The wearable device acts as an edge node, collecting and preprocessing data from sensors, which is then transmitted to a mobile app via Bluetooth or to the cloud via Wi-Fi, with GSM fallback for emergency alerts in offline conditions. The cloud hosts advanced AI models, databases, and notification services, while the frontend provides user interfaces for monitoring and administration.

The architecture is organized into four layers:

- Hardware Layer: Physical sensors and microcontroller.
- Firmware Layer: Edge processing and connectivity.
- Backend Layer: Cloud-based AI, storage, and notifications.
- Frontend Layer: Mobile and web interfaces.

2.2 Architectural Style

- **Microservices Architecture:** Independent backend services (e.g., AI Engine, Notification Service) for modularity.
- **Edge Computing:** On-device preprocessing to reduce latency and bandwidth.
- **Event-Driven Architecture:** Real-time alerts via WebSockets and push notifications.
- **Layered Architecture:** Separation of concerns across hardware, firmware, backend, and frontend.

3. COMPONENT DESCRIPTIONS

3.1 Hardware Components

The wearable device incorporates:

- **Sensors:** BioAmp for EEG/EMG, MAX30102 for HR/SpO2, MPU6050 for accelerometer/gyroscope data.
- **Microcontroller:** Arduino UNO for prototyping, ESP32 for production (dual-core, low-power, Wi-Fi/Bluetooth).

- **Communication Module:** SIM800L for GSM-based SMS/calls in offline mode.
- **Actuators:** Buzzer and vibrator for local alarms.

3.2 Firmware Components

- **Data Acquisition Module:** Captures sensor data at configurable rates (e.g., 100Hz for EEG, 1Hz for HR).
- **Preprocessing Module:** Applies noise filtering (e.g., Kalman filter) and edge AI (TensorFlow Lite) for initial anomaly detection.
- **Connectivity Module:** Manages Bluetooth LE for app pairing, Wi-Fi for cloud sync, and GSM fallback.
- **Alert Module:** Triggers local sound/vibration based on edge-detected anomalies.

3.3 Backend Components

- **AI/ML Engine:** Deploys LSTM/CNN models for time series analysis and prediction, utilizing Firebase ML or edge/cloud hybrid.
- **Database Service:** Firebase Realtime Database for live data streams, Firestore for historical logs and user profiles.
- **API Gateway:** Provides RESTful and GraphQL endpoints for data sync, queries, and integrations.
- **Notification Service:** Utilizes Firebase Cloud Messaging for push notifications, integrated with SIM800L for SMS/calls.
- **Authentication Service:** Implements OAuth 2.0 with JSON Web Tokens (JWT) for RBAC (patient, caregiver, admin).

3.4 Frontend Components

- **Mobile Application:** Cross-platform app developed with Flutter (Dart) for real-time monitoring, alerts, and settings.
- **Cloud Dashboard:** Web application built with React.js, hosted on Firebase Hosting, for multi-patient management and analytics.

Component Table

Component	Layer	Description	Technologies
Sensors	Hardware	Collect neural, vital, and motion data.	BioAmp, MAX30102, MPU6050
Microcontroller	Hardware	Process and transmit data.	ESP32, Arduino UNO
Data Acquisition	Firmware	Captures sensor data at specified rates.	C++/MicroPython
Preprocessing	Firmware	Filters noise and performs edge AI.	TensorFlow Lite
Connectivity	Firmware	Manages Bluetooth, Wi-Fi, and GSM.	ESP-IDF
AI/ML Engine	Backend	Analyzes time series for predictions.	TensorFlow, PyTorch
Database	Backend	Stores real-time and historical data.	Firebase Realtime/Firestore
API Gateway	Backend	Handles requests and integrations.	Node.js, GraphQL
Notification	Backend	Sends alerts via push, SMS, or calls.	Firebase, SIM800L
Mobile App	Frontend	User interface for monitoring.	Flutter (Dart)
Cloud Dashboard	Frontend	Web-based admin interface.	React.js

4. DATA FLOW

4.1 Data Flow Description

- **Data Collection:** Sensors (BioAmp, MAX30102, MPU6050) capture raw time series data (e.g., EEG, HR, motion).
- **Edge Processing:** Firmware preprocesses data (noise filtering, edge AI) and buffers it locally.
- **Data Transmission:** Data is sent via Bluetooth to the mobile app or Wi-Fi to the cloud; GSM triggers emergency alerts if offline.
- **Cloud Processing:** Backend AI models analyze data, store results in the database, and generate notifications.

- **User Interaction:** Frontend (app/dashboard) queries the backend for real-time updates, reports, and settings adjustments.
- **Expansion Support:** Modular AI plugins process additional health metrics without altering core data flow.

4.2 Data Flow Diagram Description

(Note: Insert diagram in DOCX using tools like Visio or Lucidchart. The diagram should depict: Sensors -> Firmware (Preprocessing/Connectivity) -> Cloud (AI/Database/Notifications) -> Frontend (App/Dashboard), with arrows indicating bidirectional communication and offline fallback paths.)

5. DEPLOYMENT CONSIDERATIONS

5.1 Deployment Model

- **Wearable Device:** Firmware deployed Over-The-Air (OTA) via cloud updates, with rollback support.
- **Mobile Application:** Distributed via Google Play Store and Apple App Store.
- **Backend Services:** Serverless deployment on Firebase (Functions, Hosting, ML), auto-scaling based on demand.
- **Cloud Dashboard:** Hosted on Firebase Hosting with Content Delivery Network (CDN) for global access.

5.2 Hardware Deployment

- **Prototyping Phase:** Utilizes Arduino UNO with breadboard-mounted sensors.
- **Production Phase:** ESP32-based wearable device, designed for <50g weight and >8-hour battery life, with modular sensor slots.

5.3 Security and Scalability

- **Security:** AES-256 encryption for data at rest and in transit; RBAC enforced via JWT; firmware signing to prevent tampering.

- **Scalability:** Firebase auto-scales to support 10,000+ concurrent users; load balancing via Google Cloud infrastructure.
- **Monitoring:** Firebase Performance Monitoring and Analytics integrated for system health and usage tracking.

5.4 Backup and Recovery

- **Data Backup:** Daily snapshots of Firestore data stored in Google Cloud Storage.
- **Recovery:** Automated rollback for OTA firmware updates; database restoration within 24 hours.

6. ASSUMPTIONS AND DEPENDENCIES

6.1 Assumptions

- Sensor accuracy meets clinical standards post-calibration.
- AI models achieve >90% prediction accuracy with representative training data.
- Network availability supports non-emergency features; GSM coverage exists for emergencies.

6.2 Dependencies

- Firebase ecosystem (Realtime Database, Firestore, Cloud Functions, Hosting, ML).
- Third-party libraries: Flutter Blue for Bluetooth, Twilio for extended SMS capabilities (if integrated).
- External datasets for initial AI model training.

7. ARCHITECTURAL CONSTRAINTS

7.1 Hardware Constraints

- Wearable weight <50g; power consumption <100mW in idle mode.
- Battery life >8 hours under continuous use.

7.2 Software Constraints

- Compliance with HIPAA and GDPR for health data handling.
- Cross-platform compatibility (iOS 15+, Android 11+, modern browsers).

8. RISK MITIGATION

8.1 Identified Risks

- Latency in cloud processing impacting real-time alerts.
- Sensor failure leading to data loss.
- Security breaches compromising patient data.

8.2 Mitigation Strategies

- Edge AI for immediate anomaly detection to reduce latency.
- Local buffering and redundancy in sensor readings.
- Regular security audits and penetration testing.

9. SUPPORTING INFORMATION

9.1 Glossary

- **Anomaly:** Abnormal readings (e.g., EEG spikes indicating seizures).
- **RBAC:** Role-Based Access Control – Security framework for user permissions.
- **LSTM:** Long Short-Term Memory – AI model for time series prediction.

- **OTA:** Over-The-Air – Wireless firmware update mechanism.

9.2 Appendices

- Appendix A: Detailed Component Interface Specifications (e.g., API endpoints, sensor protocols).
- Appendix B: Performance Benchmarks (e.g., latency targets, throughput).
- Appendix C: Diagram Templates (for future updates to data flow visuals).