Multi-Agent Cardiac Risk Assessment System - Complete

System Architecture

```
Arduino Sensors (AD8232 + MPU6050)
  ↓
Sensor Agent (Deep Learning - Real-time)
  ├── ECG CNN Model → Arrhythmia Detection
  └── Random Forest → Activity Recognition
  ↓
Patient Agent (Statistical Learning - Daily)
  ├── Baseline Learning (EMA)
  └── Behavioral State Assessment
  ↓
Decision Agent (Future - Fusion)
  └── Alert Generation
```

✅ Implemented Components

1. Sensor Agent (COMPLETE)

Location: Integrated in `app.py`

ECG Processing:

- CNN model for arrhythmia detection (10-second windows)
- R-peak detection for HRV metrics (RMSSD, SDNN)
- Heart rate calculation
- ECG quality assessment
- Output: `physio_risk`, `physio_confidence`

Activity Recognition:

- Random Forest on IMU data (128-sample windows)
- 6 activity classes
- Output: `activity`, `activity_confidence`

Outputs (every 2 seconds):

```json
{
 "timestamp": "ISO-8601",
 "activity": "WALKING",
 "activity_confidence": 0.95,
 "heart_rate": 75.3,
 "hrv_rmssd": 38.2,
 "hrv_sdnn": 52.1,
 "arrhythmia_detected": false,
 "arrhythmia_probability": 0.12,
 "physio_risk": 0.18,
 "physio_confidence": 0.92
}
```

2. Patient Agent (COMPLETE)

Location: `agents/patient_agent.py`

Core Functions:

- Daily aggregation of sensor outputs
- Baseline learning (online EMA with α=0.05)
- Behavioral state detection (STABLE/DEGRADED/AT_RISK)
- Sensitivity factor calculation (1.0x - 1.3x)
- Persistent state storage in JSON

Tracks:

- Cardiac baselines (HR, HRV, arrhythmia rate, physio risk)
- Activity patterns
- Trend detection (degrading/stable/improving days)
- Confidence building over 30 days

Outputs (daily):

```json
{
  "day": "2025-12-27",
  "sensitivity_factor": 1.15,
  "behavioral_state": "DEGRADED",
  "confidence": 0.5
}
```

3. Clinical Agent (COMPLETE)

Location: `agents/clinical_agent.py`

Core Functions:

- Static medical risk assessment
- Logistic regression on 14 clinical features
- Risk factor identification
- Web interface for profile management

Input Features:

- Demographics: age, sex, BMI
- Vital signs: SBP, DBP
- Lab values: total cholesterol, HDL
- Conditions: diabetes, smoking, hypertension
- Medications: beta blocker, antihypertensive, statin, anticoagulant

Outputs (static until updated):

```json
{
  "day": "2025-12-27",
  "clinical_risk": 0.45,
  "confidence": 1.0
}
```

Risk Levels:

- LOW: <30%
- MODERATE: 30-59%
- HIGH: 60-79%
- VERY HIGH: ≥80%

4. Decision Agent (COMPLETE)

Location: `agents/decision_agent.py`

Core Functions:

- Risk fusion from all three agents
- Persistence checking (anti-false-alarm)
- Alert generation with explanations
- Cooldown management
- Alert history logging

Fusion Formula:

```
global_risk = 0.5 × physio_risk × sensor_conf
           + 0.3 × sensitivity × patient_conf
           + 0.2 × clinical_risk
```

Decision Thresholds:

- NO_ALERT: <50%
- MONITOR: ≥50% for 3+ readings
- ALERT: ≥70% for 5+ readings
- Cooldown: 30 minutes between alerts

Outputs (every 2 seconds):

```json
{
  "timestamp": "ISO-8601",
  "global_risk": 0.72,
  "decision": "ALERT",
  "explanation": "🚨 HIGH RISK DETECTED (72%) • Contributors: Physiological: arrhythmia...",
  "components": {
    "sensor_risk": 0.65,
    "patient_sensitivity": 1.15,
    "clinical_risk": 0.60
  }
}
```

5. Web Dashboard (COMPLETE)

Location: `templates/index.html` + `templates/clinical.html` + `templates/alerts.html`

Main Dashboard Displays:

- Real-time sensor data (ECG, accelerometer, gyroscope charts)
- AI predictions (9 cards):
  1. Current Activity + confidence
  2. Heart Rate (BPM)
  3. HRV (RMSSD)
  4. HRV (SDNN)
  5. Rhythm Status (HRV-based)
  6. ECG Quality
  7. Arrhythmia Detection (CNN-based)
  8. Clinical Risk (static)
  9. Global Risk Score + Decision
- Alert banner (red, prominent when ALERT triggered)
- Patient baseline profile
- Navigation to Clinical Profile and Alerts pages

Clinical Profile Page:

- Comprehensive medical history form
- Risk calculation on submission
- Identified risk factors display
- Protective factors display
- Risk level categorization

Alerts History Page:

- Alert statistics dashboard
- Full alert history with explanations
- Risk component breakdown
- Acknowledgment system
- Cooldown indicator
- Current system status

🔄 Data Flow

Real-time Loop (Every 2 seconds):

```
Arduino → Serial → Flask
 ↓
ECG Processing (CNN + HRV)
IMU Processing (Random Forest)
 ↓
Predictions Updated
 ↓
Decision Agent Evaluates:
  - Gets Sensor output (physio_risk, confidence)
  - Gets Patient state (sensitivity, behavioral_state)
  - Gets Clinical profile (clinical_risk)
 ↓
Calculates Global Risk
 ↓
Checks Persistence + Cooldown
 ↓
Makes Decision (NO_ALERT / MONITOR / ALERT)
 ↓
Generates Explanation
 ↓
Logged to daily_sensor_outputs[]
 ↓
Dashboard Updated (with Alert Banner if needed)
```

Daily Loop (Midnight):

```
Midnight Detected
 ↓
Patient Agent.daily_update(sensor_outputs)
 ↓
Aggregate Statistics Computed
 ↓
Baseline Updated (EMA)
 ↓
Behavioral State Assessed
 ↓
Sensitivity Factor Calculated
 ↓
State Persisted to JSON
 ↓
daily_sensor_outputs[] Cleared
```

API Endpoints

Sensor Data

- `GET /api/data` - Raw sensor timeseries
- `GET /api/predictions` - Current AI predictions
- `GET /api/latest` - Latest sensor reading
- `GET /api/status` - System status
- `GET /api/clear` - Clear data buffer

Patient Agent

- `GET /api/patient/baseline` - Learned baseline values
- `GET /api/patient/state` - Current behavioral state
- `GET /api/patient/risk_context` - Current vs baseline comparison
- `POST /api/patient/trigger_update` - Manual end-of-day update

Key Design Principles

Stability Over Reactivity

- EMA prevents single-day baseline shifts
- Requires 3 consecutive degrading days for AT_RISK state
- Confidence builds gradually over 30 days

Interpretability Over Complexity

- No black-box decisions
- Every metric traceable to source
- Statistical methods over deep learning for personalization

Personalization

- Each patient has unique baseline
- Detects deviations from THEIR normal, not population average
- Sensitivity adjusts based on patient state

Conservative Alerting

- High confidence required before sensitivity increases
- Multiple risk factors needed to escalate state
- Trend-based, not spike-based

Performance Characteristics

Sensor Agent

- **Latency**: ~2 seconds per prediction
- **Activity Accuracy**: ~95% (UCI HAR baseline)
- **ECG Processing**: 10-second windows for CNN
- **Update Rate**: Every 2 seconds

Patient Agent

- **Update Frequency**: Once daily (automatic at midnight)
- **Confidence Building**: 30 days to full confidence
- **Baseline Stability**: 95% weight on history after day 7
- **State Change Threshold**: 3 consecutive days

System Resources

- **Memory**: ~3600 ECG samples buffered (10 seconds)
- **Storage**: Patient state JSON (~10 KB)
- **GPU**: Optional for ECG CNN (auto-detects CUDA)

What's NOT Implemented (Future Work)

Medical Agent

- Static clinical risk based on:
  - Age, sex, hypertension, diabetes, smoking, etc.
- Simple logistic regression
- Rare updates

Decision Agent

- Fuses sensor + patient + medical outputs
- Alert decision logic
- Explanation generation
- Cooldown between alerts

Alert System

- Notification mechanism
- Alert history
- Explanation interface

File Dependencies

Python Packages

```
flask
pyserial
numpy
scipy
pandas
scikit-learn
joblib
neurokit2
wfdb
torch
```

Model Files

- `models/activity_rf_ucihar.pkl` - Trained Random Forest
- `models/ecg_cnn_win10s_binary.pt` - Trained PyTorch CNN
- `models/clinical_agent_model.joblib` - Trained Logistic Regression
- `models/clinical_agent_features.joblib` - Feature names/order

Data Files (Auto-created)

- `data/patient_state.json` - Persistent patient baseline
- `data/clinical_profile.json` - Clinical medical profile
- `data/decision_state.json` - Alert history and decision state

Running the System

```bash
# 1. Install dependencies
pip install -r requirements.txt

# 2. Create folder structure
mkdir -p agents models data templates

# 3. Place model files in models/
#    - activity_rf_ucihar.pkl
#    - ecg_cnn_win10s_binary.pt
#    - clinical_agent_model.joblib
#    - clinical_agent_features.joblib

# 4. Place agent code in agents/
# 5. Configure serial port in app.py

# 6. Run
python app.py

# 7. Open dashboard
http://localhost:5000

# 8. Configure clinical profile (first time)
http://localhost:5000/clinical
```

Success Metrics

✅ **Sensor Agent**: Detects physiological abnormalities in real-time

✅ **Patient Agent**: Learns individual baselines with stability

✅ **Clinical Agent**: Assesses static medical vulnerability

✅ **Decision Agent**: Fuses risks and generates intelligent alerts

✅ **Integration**: Clean message passing, no circular dependencies

✅ **Explainability**: Every decision has clear reasoning

✅ **Personalization**: Adapts to individual patient patterns

✅ **Anti-False-Alarm**: Persistence requirements prevent spikes

System Maturity

Component	Status	Completeness
Sensor Agent	✅ Done	100%
Patient Agent	✅ Done	100%
Clinical Agent	✅ Done	100%
Decision Agent	✅ Done	100%
Web Dashboard	✅ Done	100%
Alert System	✅ Done	100%

🎉 **SYSTEM COMPLETE**: All 4 agents operational with full alert management!

Next Steps for Enhancement

The complete system is operational! Optional enhancements:

1. **Mobile Notifications** (~2 hours)
   - Push notifications for alerts
   - SMS integration
   - Email alerts
2. **Data Export** (~1 hour)
   - CSV export of sensor data
   - PDF report generation
   - Share with healthcare provider
3. **Advanced Analytics** (~3 hours)
   - Weekly/monthly trend reports
   - Pattern recognition
   - Correlation analysis
4. **Multi-Patient Support** (~2 hours)
   - User authentication
   - Multiple patient profiles
   - Healthcare provider dashboard

Current State: Fully functional single-patient monitoring system with AI-powered risk assessment and intelligent alerting!