System Troubleshooting Guide

Quick Diagnostics

Run the validation script first:

```bash
python system_test.py
```

This will automatically check:

- Directory structure
- Model files
- Agent functionality
- Dependencies
- Serial ports

Common Issues & Solutions

1. Import Errors

Symptom: `ModuleNotFoundError` or `ImportError`

Solutions:

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

# If specific package fails
pip install flask pyserial numpy scipy pandas scikit-learn joblib neurokit2 wfdb torch

# For PyTorch issues
pip install torch torchvision --index-url https://download.pytorch.org/whl/cpu
```

2. Serial Connection Issues

Symptom: "Could not establish serial connection"

Diagnose:

```python
import serial.tools.list_ports
ports = list(serial.tools.list_ports.comports())
for port in ports:
    print(f"{port.device}: {port.description}")
```

Solutions:

- **Windows**: Check Device Manager → Ports (COM & LPT)
- **Linux**: Add user to dialout group: `sudo usermod -a -G dialout $USER`
- **Mac**: Check `/dev/cu.usbserial*` or `/dev/cu.usbmodem*`
- Update `SERIAL_PORT` in `app.py` to match your Arduino port
- Check Arduino is plugged in and recognized
- Try different USB ports/cables
- Ensure Arduino IDE isn't using the port

3. Model Loading Failures

Symptom: "Could not load model" warnings

Check:

```bash
ls -lh models/
```

Expected files:

- `activity_rf_ucihar.pkl` (scikit-learn RandomForest)
- `ecg_cnn_win10s_binary.pt` (PyTorch checkpoint)
- `clinical_agent_model.joblib` (scikit-learn LogisticRegression)
- `clinical_agent_features.joblib` (Python list)

Solutions:

- Ensure files are in correct location
- Check file permissions (readable)
- Verify model format matches expected type
- For PyTorch: check version compatibility
- System will use defaults if models missing (degraded mode)

4. Division by Zero / Math Errors

Symptom: `ZeroDivisionError` or `RuntimeWarning: divide by zero`

Fixed in latest code:

- Added `max()` guards in Patient Agent baseline calculations
- Clamped sensitivity normalization in Decision Agent
- Protected against zero std deviation

If still occurring:

- Check `data/*.json` files for corruption
- Delete state files and restart (system will reinitialize)

5. Dashboard Not Updating

Symptom: Charts/values frozen or not changing

Check:

1. Browser console (F12) for JavaScript errors
2. Flask server logs for Python errors
3. Network tab to see if API calls failing

Solutions:

- Hard refresh browser (Ctrl+F5 / Cmd+Shift+R)
- Check Flask server is running without errors
- Verify serial connection is active
- Check if data collection paused (Pause button)
- Try different browser (Chrome/Firefox recommended)

6. Alert System Issues

Problem: Alerts not triggering when expected

Check:

- Global risk must be ≥70% for 5 consecutive readings
- System may be in cooldown (30 min after last alert)
- Check consecutive counters in `/api/decision/current`

Problem: Too many false alerts

Solutions:

- Adjust thresholds in `decision_agent.py`:

```python
ALERT_THRESHOLD = 0.8  # Increase from 0.7
ALERT_PERSISTENCE = 7  # Increase from 5
```

Problem: Alert banner stuck on screen

- Check browser console for errors
- Refresh page
- Verify decision API returning correct status

7. Clinical Profile Not Saving

Symptom: Profile resets or doesn't update risk

Check:

```bash
cat data/clinical_profile.json
```

Solutions:

- Ensure `data/` directory exists and is writable
- Check file permissions
- Look for JSON syntax errors
- Clear browser cache
- Check browser console for POST errors

8. Patient Baseline Not Learning

Symptom: Baseline values not changing over days

Check:

- `days_seen` in patient state
- Need at least 1 day for initial updates
- EMA is slow by design (95% weight on history)

Verify:

```bash
cat data/patient_state.json | grep days_seen
```

Expected behavior:

- First 7 days: faster learning (α=0.2)
- After 7 days: slow/stable (α=0.05)
- Takes ~30 days for full confidence

9. Memory / Performance Issues

Symptom: System slowing down or using too much RAM

Solutions:

- Reduce `MAX_DATA_POINTS` in `app.py` (currently 3600)
- Clear alert history periodically
- Restart Flask server daily
- Check for memory leaks in browser (reload page)

10. State File Corruption

Symptom: JSON decode errors on startup

Fix:

```bash
# Backup first
cp data/patient_state.json data/patient_state.json.backup

# Delete corrupted files
rm data/patient_state.json
rm data/clinical_profile.json
rm data/decision_state.json

# System will recreate on next run
python app.py
```

Debug Mode

Enable detailed logging:

In app.py:

```python
import logging
logging.basicConfig(level=logging.DEBUG)
```

In agents: Add print statements in critical functions:

```python
def make_decision(self, ...):
    print(f"DEBUG: global_risk={global_risk}, decision={decision}")
    ...
```

Validation Checklist

Before reporting issues, verify:

- [ ] Python 3.8+ installed
- [ ] All dependencies installed (`pip install -r requirements.txt`)
- [ ] Directory structure correct (`agents/`, `models/`, `data/`, `templates/`)
- [ ] Model files present (4 files in `models/`)
- [ ] Template files present (3 HTML files)
- [ ] Serial port configured correctly (or running without Arduino)
- [ ] Browser is modern (Chrome/Firefox/Edge)
- [ ] No firewall blocking localhost:5000
- [ ] Flask server running without errors

Getting Logs

System logs:

```bash
python app.py 2>&1 | tee system.log
```

Browser logs:

1. Open Developer Tools (F12)
2. Go to Console tab
3. Right-click → Save as...

State dumps:

```bash
# Patient state
cat data/patient_state.json | python -m json.tool

# Clinical profile
cat data/clinical_profile.json | python -m json.tool

# Decision state
cat data/decision_state.json | python -m json.tool
```

Emergency Reset

If system is completely broken:

```bash
# 1. Stop Flask server (Ctrl+C)

# 2. Backup data
mkdir backup_$(date +%Y%m%d)
cp -r data/*.json backup_$(date +%Y%m%d)/

# 3. Clear all state
rm data/*.json

# 4. Restart fresh
python app.py
```

System will reinitialize with default values.

Known Limitations

1. **Single patient only** - No multi-user support yet
2. **No cloud sync** - All data stored locally
3. **Browser-only interface** - No mobile app
4. **English only** - No internationalization
5. **Simulation mode limited** - No Arduino = no real sensor data

---

Performance Benchmarks

Expected behavior:

Metric	Expected	Action if Different
Sensor update rate	2 seconds	Check serial connection
Dashboard update	100ms	Check browser performance
Patient update	Daily (midnight)	Check system clock
Alert latency	10 seconds	Expected (5 readings × 2s)
Memory usage	<500 MB	Reduce MAX_DATA_POINTS
CPU usage	<10% idle	Check for infinite loops

Support Resources

1. **Run validation**: `python system_test.py`
2. **Check logs**: Server output + browser console
3. **Review state files**: `data/*.json`
4. **Test individual agents**: See test script examples
5. **Read documentation**: Agent-specific guides

System Health Check

Quick commands to verify system health:

bash

```bash
# Check processes
ps aux | grep python

# Check files
ls -lR agents/ models/ data/ templates/

# Check ports
netstat -an | grep 5000

# Check Python
python --version
pip list | grep -E "flask|torch|sklearn|neurokit"

# Test imports
python -c "from agents.patient_agent import PatientAgent; print('OK')"
python -c "from agents.clinical_agent import ClinicalAgent; print('OK')"
python -c "from agents.decision_agent import DecisionAgent; print('OK')"
```

All should complete without errors for healthy system.