

Project Deployment Checklist

Pre-Deployment Setup

1. File Structure

- Create `agents/` directory
- Create `models/` directory
- Create `data/` directory
- Create `templates/` directory
- Verify all Python files in place
- Verify all HTML files in place

2. Dependencies

- Python 3.8 or higher installed
- Run `pip install -r requirements.txt`
- Verify all packages installed: `pip list`
- No ImportError when running test script

3. Model Files

- `models/activity_rf_ucihar.pkl` present
- `models/ecg_cnn_win10s_binary.pt` present
- `models/clinical_agent_model.joblib` present
- `models/clinical_agent_features.joblib` present
- All model files readable (check permissions)

4. Hardware Setup (if using Arduino)

- Arduino Nano connected via USB
 - AD8232 ECG sensor wired correctly
 - MPU6050 IMU sensor wired correctly
 - Arduino sketch uploaded
 - Serial port identified (COM3, /dev/ttyUSB0, etc.)
 - Update `SERIAL_PORT` in `app.py`
-

First Run

1. Validation Test

```
bash
```

```
python system_test.py
```

- All tests pass or show clear warnings
- Agent imports successful
- No critical errors

2. Start Application

```
bash
```

```
python app.py
```

- Server starts without errors
- Shows model loading status
- Shows agent initialization
- Displays web URLs

3. Access Dashboard

Open <http://localhost:5000>

- Page loads successfully
- Status indicator shows connection status
- 9 prediction cards visible
- Patient baseline section visible
- No JavaScript errors in console (F12)

4. Test Clinical Profile

Navigate to <http://localhost:5000/clinical>

- Form loads correctly
- Can enter all fields
- "Calculate Risk" button works
- Risk score displays after submission
- Risk factors shown

5. Test Alert System

Navigate to <http://localhost:5000/alerts>

- Page loads
 - Statistics show zeros initially
 - Empty state message shows
 - No errors in console
-

Functionality Tests

Real-Time Monitoring

- Sensor data updating (if Arduino connected)
- Charts animating smoothly
- Current values updating
- Activity prediction showing
- Heart rate calculating
- ECG quality indicator working

Patient Agent

- Baseline values initialize with defaults
- Days seen counter at 0 initially
- Confidence shows 0%
- Behavioral state shows STABLE
- (Wait for midnight) Daily update occurs

Clinical Agent

- Can save profile
- Risk calculates correctly
- Risk factors identified
- Profile persists after refresh

Decision Agent

- Global risk displays
 - Decision status shows NO_ALERT initially
 - (Simulate high risk) Alert triggers after persistence
 - Alert banner appears when ALERT
 - Cooldown activates after alert
 - Alert history logs correctly
-

Data Persistence

Check State Files

```
bash
```

```
ls -lh data/
```

- `patient_state.json` created after first data
- `clinical_profile.json` created after profile save
- `decision_state.json` created after first decision
- Files are valid JSON (can open in text editor)

After Restart

```
bash
```

```
# Stop server (Ctrl+C)  
# Restart server  
python app.py
```

- Patient days_seen preserved
- Clinical profile preserved
- Alert history preserved
- Baselines preserved

Performance Tests

Load Test

Leave running for 30 minutes:

- No memory leaks (check task manager)
- CPU usage stable (<20%)
- No error accumulation in logs
- Dashboard remains responsive
- Charts update smoothly

Data Volume Test

- Handles 3600 data points without lag
 - Alert history handles 100+ alerts
 - State files remain <1MB
 - No browser crashes
-

Security Checklist

Network

- Only accessible on localhost (not public)
- No external API calls (all local)
- No cloud data transmission
- Firewall allows localhost:5000

Data

- State files have appropriate permissions
 - No sensitive data in logs
 - Patient ID anonymized if needed
 - No PHI in error messages
-

Documentation Check

Code Documentation

- All agents have docstrings
- Key functions commented
- Configuration parameters documented
- API endpoints documented

User Documentation

- README.md complete
 - Setup guide available
 - Troubleshooting guide available
 - Agent-specific guides available
-

Edge Cases Tested

No Arduino Connected

- System starts without error
- Shows warning about serial connection
- Dashboard loads but shows no data
- Can still configure clinical profile
- Can view documentation

No Model Files

- System starts with warnings
- Uses default values
- Dashboard shows "model not loaded"
- Basic functionality works

Corrupted State Files

- Delete `data/*.json` and restart
- System reinitializes correctly
- No crashes or errors

Extreme Values

- Very high heart rate (>200 BPM) handled
 - Very low values handled
 - Negative values rejected
 - NaN/Inf values handled
-

User Acceptance Testing

First-Time User

- Can follow README to get started
- Understands what each metric means
- Can configure clinical profile
- Understands alert levels
- Knows when to seek help

Daily Usage

- Easy to check status
- Clear when action needed
- Alert explanations helpful
- Not overwhelming with data

Long-Term Usage

- Baseline adapts over weeks
 - Historical trends visible
 - Alert frequency acceptable
 - System remains stable
-

Production Readiness

Code Quality

- No `[TODO]` comments remaining
- No debug print statements in production code
- Error handling comprehensive
- Input validation present

Monitoring

- Logs are informative
- Errors clearly reported
- Status endpoints working
- Health checks possible

Maintenance

- State files can be backed up
 - System can be cleanly restarted
 - Updates won't break compatibility
 - Data migration path exists
-

Final Verification

Run complete test suite:

```
bash  
  
python system_test.py  
python app.py  
# Open all three web pages  
# Use system for 10 minutes  
# Check logs for errors
```

System is ready for deployment if:

- All critical tests pass
 - No critical errors in logs
 - All three pages load and function
 - Data persists across restarts
 - Alerts trigger appropriately
-

Post-Deployment

Day 1

- Monitor logs for errors
- Check data collection working
- Verify state files updating

Week 1

- Patient baseline learning
- No stability issues
- Alert system functioning

Month 1

- Full confidence reached
 - Personalization working
 - No long-term issues
-

Rollback Plan

If critical issues arise:

1. **Stop system:** `Ctrl+C`
 2. **Backup data:** `cp -r data/ backup/`
 3. **Check logs:** Review error messages
 4. **Restore previous version** if needed
 5. **Test in isolation:** Run `system_test.py`
 6. **Fix and redeploy**
-

Success Criteria

 System successfully:

- Collects and displays sensor data
- Learns patient baselines
- Incorporates clinical context
- Generates intelligent alerts
- Persists state across restarts
- Runs stably for extended periods
- Provides clear, actionable information

System Status: READY FOR DEPLOYMENT 