This comprehensive **Master Technical Documentation** preserves every architectural decision, code snippet, and bug fix implemented for **Phase 3: Inter-Hospital Transfer Coordination**.

EMsync Phase 3: Master Technical Documentation

Project: Emergency Coordination Hub (Kerala AI-IoT Framework)
Module: Secure Inter-Hospital Transfer & Real-Time Coordination

---

1. Executive Summary

Phase 3 successfully transitions the EMsync project from static data models to a **Live Event-Driven Network**. It establishes a secure "Command Center" where paramedics can request transfers, and hospitals can manage resources and receive clinical handoffs in real-time.

---

2. System Architecture & Tech Stack

The system uses a **Decoupled Full-Stack Architecture** to ensure sub-second latency for emergency alerts.

- **Backend:** Python 3.9 + Flask (REST API)
- **Database:** SQLite with SQLAlchemy (Path: Virtual_network_hospitals/instance/emsync_network.db)
- **Real-Time Bridge:** Flask-SocketIO (WebSockets) for bidirectional network broadcasting
- **Security Engine:** PyCryptodome (AES-256 Bit Encryption)
- **Frontend UI:** React 18.x with Lucide-React and Axios

---

3. Backend Implementation (backend_api_app.py)

3.1. Server Initialization

The hub integrates CORS for React connectivity and SocketIO for real-time alerts.

Python

```
import os
import json
from flask import Flask, request, jsonify
from flask_cors import CORS
from flask_socketio import SocketIO, emit
from hosital_model import db, Hospital, TransferSession
from security import encrypt_patient_data, decrypt_patient_data
from sbar_logic import generate_sbar_report

app = Flask(__name__)
CORS(app)
socketio = SocketIO(app, cors_allowed_origins="*")

# Database Configuration
base_dir = os.path.abspath(os.path.dirname(__file__))
db_path = os.path.join(base_dir, 'instance', 'emsync_network.db')
app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:///{db_path}'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db.init_app(app)
```

3.2. Core Logic Endpoints

Endpoint	Logic Detail
/api/request_transfer	Randomizes patient identity (Alan, Abhijith, etc.) $\rightarrow$ Encrypts PII $\rightarrow$ Generates SBAR $\rightarrow$ Broadcasts WebSocket Alert.
/api/accept_transfer	**The Official Commit:** Reduces hospital available_icu_beds and updates session status to ACCEPTED.
/api/decrypt	Safely parses the SBAR JSON using json.loads (to avoid null NameErrors) and returns the plain-text identity.

/api/reset_simulation	Wipes the TransferSession table and restores bed counts for Aster, Lourdes, and Rajagiri.

---

4. Clinical & Security Modules

4.1. Security Engine (security.py)

Implements **AES-256 (CBC Mode)** to ensure Zero-Knowledge transmission of patient demographics.

- **Tokenization:** Patient Name and DOB are encrypted into a Base64 string.
- **Handshake:** The token is visible to the hospital immediately, but the "Unlock" (Decryption) only occurs after a formal acceptance.

4.2. SBAR Clinical Handoff (sbar_logic.py)

Automates Situation, Background, Assessment, and Recommendation reports.

Critical Implementation Change: Type Safety
The following code fix ensures vitals sent as strings from the React frontend are properly analyzed:

Python

```
# Assessment logic with Float Casting fix
assessment = {
    "vitals_snapshot": vitals,
    "hypoxia": "Yes" if float(vitals.get("o2sat", 100)) < 90 else "No",
    "fever": "Yes" if float(vitals.get("temp", 37)) >= 38 else "No",
    "predicted_needs": resource_needs
}
```

---

5. React Command Center UI (App.js)

5.1. Randomized Simulation Engine

Ensures the demonstration is dynamic by selecting from a specific pool of identities.

JavaScript

```
const mockPatients = [
 { name: "Alan", dob: "1995-05-12", age: 30, complaint: "Respiratory Distress" },
 { name: "Abhijith", dob: "1998-11-20", age: 27, complaint: "Cardiac Symptoms" },
 { name: "Batman", dob: "1939-05-27", age: 86, complaint: "Blunt Force Trauma" },
 { name: "Spiderman", dob: "1962-08-10", age: 63, complaint: "Venom Exposure" },
 { name: "Luffy", dob: "1997-07-22", age: 28, complaint: "Extreme Exhaustion" }
];
```

5.2. Four-Column Command Grid

1. **Registry (Paramedic):** Live resource tracking across the Kerala network nodes.
2. **Incoming (Hospital):** Real-time alert feed displaying the **AES-256 Token ID** for secure identification.
3. **Handshake (Security):** Automated decryption visualization displaying patient identity and SBAR JSON upon commit.
4. **Admin (Override):** Direct database modification fields and a global **Reset Simulation** button for demo repeatability.

---

6. Operational Implementation Log

To restore this environment, follow these steps:

1. **Backend Setup:** Run python backend_api_app.py on port 5000.
2. **Frontend Setup:** Run npm start in the emsync-frontend folder.
3. **Simulation Lifecycle:**
   - Click **Request** (Paramedic)
   - Observe **AES Token** (Hospital Incoming)
   - Click **Accept** (Hospital) $\rightarrow$ Bed Count Decreases
   - View **Decrypted PII** (Security Handshake)

Phase 3 Status: FULLY COMPLETED & DOCUMENTED.

Would you like me to generate the Phase 4 Integration Plan, which will link your Phase 1 Severity ML Model (.pkl) directly into this real-time Command Center?