This is the **Master Project Documentation** for **EMsync: Emergency Coordination Hub**. It compiles all technical progress from Phase 1 to the current status of Phase 3, structured for absolute clarity and reproducibility.

EMsync: Master Project Documentation

1. Project Overview

EMsync is an integrated AI and IoT platform designed to synchronize ambulance crews and hospitals in real-time[1][1].

- **Phase 1:** AI-driven severity prediction using clinical datasets[2].
- **Phase 2:** Dynamic ambulance routing with real-time traffic integration[3].
- **Phase 3:** Inter-hospital transfer coordination and secure resource booking (Current Phase).

---

2. Environment & Infrastructure

Active Development Environment

- **Env Name:** virtual_network_emsync_env
- **Python Version:** 3.9
- **Core Libraries:** pandas, scikit-learn, xgboost, Flask-SQLAlchemy, pycryptodome, flask-socketio, flask-cors.

Project Directory Structure

Plaintext

```
G:\Final Year PROJECTTTTTTTTTTTTTTT\p3 interhospital Transfer\Virtual_network_hospitals\
├── instance/
│   └── emsync_network.db      # Virtual Registry Database
├── hosital_model.py           # Database Schema (Hospitals & Sessions)
├── security.py                # AES-256 Encryption Engine
```

```
├── backend_api_app.py        # Main Coordination API
├── init_db.py                # Database Initialization Script
└── check_db.py               # Verification Utility
```

---

3. Phase 1: Severity Prediction (ML Logic)

- **Dataset:** MIMIC-IV-ED clinical data[4].
- **Model:** XGBoost Classifier[555].
- **Input:** 20 features including core vitals (Temp, HR, SpO2, SBP, DBP) and engineered flags (Fever, Hypoxia)[6].
- **Performance:** 71% Balanced Accuracy for 3 urgency levels (Critical, Moderate, Low Urgency)[777].
- **Logic:** Standardizes patient acuity to allow hospitals to prepare resources (e.g., ICU beds, ventilators) before arrival[8].

---

4. Phase 2: Smart Routing (Navigation Logic)

- **Tech:** Flask (Backend) + React (Frontend)[9].
- **API:** OpenRouteService (ORS) for dynamic pathfinding[10].
- **Logic:** Instead of static distance, it uses real-time traffic data to calculate the fastest path to the selected hospital[11].
- **Animation:** Client-side React state updates ensure smooth marker movement without page refreshes[12].

---

5. Phase 3: Inter-Hospital Coordination (Current)

A. Virtual Hospital Registry (hosital_model.py)

Logic: Creates a "Source of Truth" for hospital resources.

- **Hospital Model:** Tracks name, coordinates (Lat/Lon), available_icu_beds, and hardware

availability (has_ventilator, has_specialist).
- **TransferSession Model:** Manages the lifecycle of a patient move (States: PENDING, ACCEPTED, EN_ROUTE, COMPLETED).

B. AES-256 Encryption (security.py)

Logic: Protects patient privacy (PII) during transfer.

- **Standard:** AES-256 in CBC mode with a unique IV (Initialization Vector) per session.
- **Process:** Patient names and conditions are encrypted into a "Blob" string. This string acts as the **Transfer ID**.
- **Handshake:** Only the authorized receiving hospital can decrypt this string to view patient details.

C. Coordination API (backend_api_app.py)

Logic: The central hub for resource allocation.

- **GET /api/hospitals**: Queries the database to show paramedics which hospitals have open beds and the right specialists.
- **POST /api/request_transfer**:
  1. Encrypts patient data via security.py.
  2. Atomically decrements (-1) the bed count in the database to "hold" the resource.
  3. Initializes a PENDING session.

---

6. Command Log

Initialize Virtual Network

Bash

```
# Populate the database with Aster Medicity, Lourdes, and Rajagiri
python init_db.py
```

Run the Backend API

Bash

```
# Starts the server on port 5000
python backend_api_app.py
```

---

7. State Machine Logic (The Transfer "Pulse")

1. **Search:** Paramedic searches for hospitals with available_icu_beds > 0.
2. **Request:** API encrypts PII $\rightarrow$ Bed is reserved $\rightarrow$ Status = PENDING.
3. **Accept:** Hospital confirms $\rightarrow$ Bed is occupied $\rightarrow$ Status = ACCEPTED.
4. **Handoff:** Arrival $\rightarrow$ SBAR report finalized $\rightarrow$ Status = COMPLETED.

Current Progress: The Virtual Registry and Secure API are fully operational. The system is ready to automate the **SBAR Report generation** using the Phase 1 severity scores.