This is the master documentation log for **EMsync Phase 3: Inter-Hospital Transfer Coordination** as of December 22, 2025 11:26 AM . This document is designed to be a standalone technical guide for the project's current state[1].

EMsync Phase 3: Master Documentation Log

I. Technical Architecture & Logic

The goal of Phase 3 is to transition EMsync into a transactional coordination platform using a "Virtual Hospital Network"[2222222].

1. Virtual Node Logic

- **Concept**: Since real-world hospital APIs are unavailable, the system uses a centralized **Virtual Registry**[3].
- **Logic**: A database serves as the "Source of Truth," representing multiple hospital nodes (e.g., Aster Medicity, Lourdes Hospital)[444444444].
- **Data Points**: Each node tracks static data (GPS coordinates) and dynamic data (ICU bed availability, specialist status)[555555555].

2. Transactional State Machine

- **Purpose**: To manage the lifecycle of a patient transfer and prevent "double-booking" of critical resources[66666].
- **States**:
  - PENDING: Transfer requested by Originating Hospital.
  - ACCEPTED: Receiving Hospital confirms availability; resource is "locked."
  - *Upcoming*: EN_ROUTE and COMPLETED states for live tracking.

3. Security & Theory Concepts

- **AES-256 Encryption**: A symmetric encryption standard used to generate **Transfer**

IDs[777777777]. It protects Patient Identifiable Information (PII) by only releasing decryption keys to authorized receiving nodes[888888888].

- **SBAR Protocol**: A standardized clinical communication framework: **S**ituation, **B**ackground, ****Assessment, **R**ecommendation[999999999]. EMsync automates this by mapping ML severity predictions (Phase 1) and IoT vitals (Phase 2) into a digital report[10101010].

- **RBAC (Role-Based Access Control)**: Logic to differentiate dashboard views for **Ambulance Crews** (navigation/vitals) and **Hospital Staff** (resource/bed management)[1111111111111111].

II. Command & Implementation Log

1. Environment Initialization

Commands executed to set up the dedicated Phase 3 virtual environment:

Bash

```
# Create the environment
conda create --name virtual_network_emsync_env

# Activate the environment
conda activate virtual_network_emsync_env

# Install core dependencies for Phase 3
pip install Flask-SQLAlchemy pycryptodome flask-socketio
```

2. Project Directory Structure

Plaintext

```
G:\Final Year PROJECTTTTTTTTTTTTTT\p3 interhospital Transfer\
└── Virtual_network_hospitals\
```

```
├── __pycache__\
├── instance\
│   └── emsync_network.db    # SQLite Database file
├── hosital_model.py         # Database Schema (Renamed per user)
├── init_db.py               # Database Initialization Script
└── check_db.py              # Data Verification Script
```

III. Code Snippets & Logic

1. Data Model (hosital_model.py)

Defines the structure for virtual hospitals and transfer sessions.

Python

```python
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class Hospital(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)

    # Coordinates for Phase 2 Routing integration
    lat = db.Column(db.Float, nullable=False)
    lon = db.Column(db.Float, nullable=False)

    # Resource tracking
    total_icu_beds = db.Column(db.Integer, default=10)
    available_icu_beds = db.Column(db.Integer, default=5)
    has_ventilator = db.Column(db.Boolean, default=True)
    has_specialist = db.Column(db.Boolean, default=True)

class TransferSession(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    transfer_id_encrypted = db.Column(db.String(255), unique=True, nullable=False)
    origin_hospital_id = db.Column(db.Integer, db.ForeignKey('hospital.id'))
    destination_hospital_id = db.Column(db.Integer, db.ForeignKey('hospital.id'))
```

```
    status = db.Column(db.String(50), default='PENDING')
    sbar_json = db.Column(db.Text, nullable=True)
```

2. Database Population (init_db.py)

Seeds the network with real-world hospital names used in Phase 2.

Python

```
import os
from flask import Flask
from hosital_model import db, Hospital

app = Flask(__name__)
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)

def seed_data():
    with app.app_context():
        db.create_all()
        db.session.query(Hospital).delete() # Reset for fresh data

        hospitals = [
            Hospital(name="Aster Medicity", lat=10.0526, lon=76.2694, total_icu_beds=30,
available_icu_beds=12, has_ventilator=True, has_specialist=True),
            Hospital(name="Lourdes Hospital", lat=9.9961, lon=76.2843, total_icu_beds=25,
available_icu_beds=5, has_ventilator=True, has_specialist=True),
            Hospital(name="Rajagiri Hospital", lat=10.1082, lon=76.3507, total_icu_beds=40,
available_icu_beds=18, has_ventilator=True, has_specialist=True),
            Hospital(name="General Hospital Ernakulam", lat=9.9774, lon=76.2807, total_icu_beds=20,
available_icu_beds=0, has_ventilator=False, has_specialist=True)
        ]
        db.session.bulk_save_objects(hospitals)
        db.session.commit()
```

3. Verification Utility (check_db.py)

Ensures database integrity and absolute path resolution.

Python

```python
import os
from flask import Flask
from hosital_model import db, Hospital

app = Flask(__name__)
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}'
db.init_app(app)

def verify_hospitals():
    with app.app_context():
        hospitals = Hospital.query.all()
        for h in hospitals:
            print(f"{h.name} | Beds: {h.available_icu_beds}/{h.total_icu_beds}")

if __name__ == '__main__':
    verify_hospitals()
```

IV. Current Project Roadmap Status

- **Phase 1 & 2**: Complete (XGBoost models at 71% accuracy, Smart Routing prototype)[12121212].

- **Phase 3 (Ongoing)**:
  - [X] Virtual Registry Data Modeling.
  - [X] Environment & Database Seeding.
  - [ ] AES-256 Encryption Utility.
  - [ ] Bed Booking API Implementation.
  - [ ] SBAR Digital Handoff Automation.

End of Current Documentation Log.