EmpowerNet System Design Document

System Architecture Overview

EmpowerNet implements a **Hierarchical Swarm Architecture** that combines the intelligence of specialized AI agents with robust cost controls and safety mechanisms. The system is built around a central "Supervisor Node" that acts as the orchestration brain, routing user requests to specialized agents while maintaining strict financial and operational guardrails.

Core Architectural Principles

1. **Hierarchical Swarm Design**: A supervisor-controlled multi-agent system where specialized "Digital Didis" handle domain-specific tasks
2. **Emergency Bypass System**: Critical safety monitoring that can override normal routing for immediate crisis response
3. **Financial Shield Architecture**: Built-in cost controls including recursion limits, token caps, and single-response patterns
4. **Voice-First Accessibility**: WhatsApp-based interface with Whisper transcription for users with varying literacy levels

System Flow:

1. WhatsApp User sends message to FastAPI Webhook
2. Anti-Retry Shield processes and validates message
3. Whisper Audio Processing transcribes voice messages
4. Supervisor Node performs emergency check and intent classification
5. If crisis detected, routes to SOS Trigger; otherwise routes to appropriate specialized agent
6. Specialized agents (Rights Guard, Skill Mentor, Dream Builder, Community Voice, Commute Partner, Financial Services) process requests
7. Response Refiner processes agent output
8. Final response sent back through WhatsApp

Core Components & Specialized Agents

The Supervisor Node (Central Brain)

The Supervisor serves as the entry point and orchestration layer for all user interactions:

Key Responsibilities:

- Message normalization and context loading
- User validation and session management
- Routing decisions based on intent classification
- State management across conversation turns

Implementation Details: The supervisor validates user sessions and manages conversation state across multiple interactions.

Emergency Bypass (SOS Trigger)

Agent Name: SOS Trigger (Emergency Response Node) **Purpose:** Immediate crisis detection and safety network activation

Critical Features:

- **Multilingual Crisis Detection**: Monitors for keywords like "bachao", "help", "sos" in Bengali, Hindi, and English
- **Pattern Recognition**: Detects ALL CAPS messages with exclamation marks as distress signals
- **Network Activation**: Automatically alerts nearby Safety Guardians within 1km radius
- **Location-Based Response**: Provides specific 3-step safety plans based on user's current location

Emergency Detection Logic: The system uses multi-layered crisis detection including direct keyword matching in multiple languages and stress pattern detection through message formatting analysis.

Specialized Agent Swarm

Rights Guard (Legal Protection Node) - Legal Protection

Domain: Labor law, wage auditing, workplace rights **Key Features:**

- Minimum wage auditing against 2026 standards for specific sectors (Agriculture, Bakery, Construction, etc.)
- Plain-language legal guidance on harassment, maternity leave, and worker safety
- RAG-based search through legal PDFs
- Referral system to local labor unions for complex cases

Skill Mentor (Education Node) - Education & Training

Domain: Government skill programs, learning opportunities **Key Features:**

- Integration with Utkarsh Bangla and other state training programs
- Skill gap analysis based on local market demand
- Personalized learning path recommendations
- Certification and credential tracking

Dream Builder (Opportunity Node) - Economic Opportunities

Domain: Job discovery, micro-business setup **Key Features:**

- Hyper-local job search within 5km radius
- Community-vetted workplace recommendations
- Micro-business setup guidance and mentorship
- Market opportunity analysis for informal sector work

Financial Services (Finance Node) - Micro-Finance

Domain: Self-Help Groups, micro-loans, financial inclusion **Key Features:**

- SHG matching based on location and financial needs

- Micro-loan application assistance and requirements explanation
- Basic financial literacy education
- Credit building guidance for informal workers

Commute Partner (Safety Node) - Safety Companionship

Domain: Finding walking companions, safe travel **Key Features:**

- "Safety Sister" matching for walking home after work
- Real-time location sharing for trusted companions
- Safe route recommendations based on community reports
- Group formation for regular commute patterns

Community Voice (Analytics Node) - Institutional Reporting

Domain: Data analytics, community insights, safety reporting **Key Features:**

- Anonymized heatmaps of service demand and safety concerns
- Skill gap analytics for NGOs and government planning
- Red zone reporting for unsafe areas or workplaces
- Community trend analysis and resource allocation insights

Data Flow Logic

Message Lifecycle Architecture

The system implements a carefully orchestrated 6-stage pipeline:

Message Processing Flow:

1. User sends Voice/Text Message via WhatsApp
2. Webhook receives message and performs ID Check
3. Anti-Retry Shield performs Duplicate Detection
4. Supervisor performs Emergency Check
5. If Emergency Detected, routes to Emergency Response (SOS); otherwise routes to appropriate Specialist
6. Specialized Agent processes with Domain Tools
7. Response Refiner processes Raw Agent Output using GPT-4o-mini Refinement
8. Simple response sent back to User

Stage-by-Stage Breakdown

1. **Ingress (WhatsApp → FastAPI)**
    - Webhook receives message with metadata (sender, timestamp, message_id)
    - Immediate 200 response to prevent Meta retries
    - Background task initiation for processing
2. **Ingestion & Anti-Retry Shield**
    - Message ID tracking in memory set (5-minute window)
    - Duplicate detection prevents expensive re-processing

- User acknowledgment sent immediately ("Sunte paachhi... let me check")

3. **Normalization (Whisper Processing)**
    - Voice messages transcribed using OpenAI Whisper
    - Text normalization for consistent processing
    - Language detection and context preservation
4. **Processing (LangGraph Swarm)**
    - Emergency bypass check using `is_emergency()` function
    - Supervisor intent classification using GPT-4o
    - Specialized agent invocation with state management
    - Tool execution (legal search, spatial queries, etc.)
5. **Refinement (Didi Refiner)**
    - GPT-4o-mini post-processing for cost optimization
    - Complex agent outputs simplified to "Benglish" or Hindi
    - 250-token response limit for cost control
6. **Egress (WhatsApp Response)**
    - Final message delivery through WhatsApp Business API
    - Conversation state persistence for follow-up interactions

Operational Safety & Cost Design

Anti-Retry Shield

Problem Solved: Meta's webhook system can send duplicate messages during network delays, causing expensive re-processing.

Solution Implementation: The system uses memory-based deduplication with a processed message ID tracking system. Messages are stored in memory for a 5-minute window to prevent duplicate processing while managing memory usage through periodic cleanup.

Financial Shields

One-Way Specialist Edges

Design Pattern: Each specialized agent connects directly to END after providing their response, preventing expensive agent-to-agent loops. All specialists (rights_guard, skill_mentor, dream_builder, etc.) terminate directly without chaining to other agents.

Recursion Limits & Token Caps

Multi-layered Cost Control:

- **Recursion Limit:** Maximum 8 agent transitions per conversation
- **Token Cap:** 400 tokens maximum per specialist response

- **Model Selection:** GPT-4o for routing, GPT-4o-mini for refinement (90% cost savings)

The system implements comprehensive cost controls including thread ID management, token limits for specialists, and recursion limits to prevent infinite loops.

The Response Refiner

Purpose: Post-swarm processing to make responses accessible and cost-effective

Key Features:

- Uses GPT-4o-mini (90% cheaper than GPT-4o) for final response generation
- Converts complex agent outputs into simple "Benglish" or Hindi
- 250-token response limit ensures concise, actionable guidance
- Maintains empathetic tone while reducing technical complexity

The refiner takes raw agent output and user queries, then uses cost-optimized models to create simple, accessible responses with strict token limits for cost and clarity control.

Infrastructure & Storage Strategy

Compute Architecture

Deployment Model: Containerized microservices with auto-scaling capabilities

Core Components:

- **FastAPI Application:** Main webhook handler and API gateway
- **LangGraph Runtime:** Multi-agent orchestration engine
- **Background Task Queue:** Asynchronous message processing
- **Load Balancer:** AWS Application Load Balancer for high availability

Scaling Strategy:

- Horizontal scaling via AWS EC2 Auto Scaling Groups
- Container orchestration using Docker and AWS ECS
- Redis-based session management for stateless scaling

Persistence Strategy

PostgreSQL with pgvector (Primary Database)

Use Cases:

- User profiles and conversation history
- Legal document storage and RAG-based search
- Spatial data for location-based services
- Community analytics and reporting data

Key Features:

- pgvector extension for similarity search and embeddings

- AWS RDS/Aurora for managed database services
- Automated backups and point-in-time recovery
- Read replicas for analytics workloads

Redis Streams (Event Bus)

Use Cases:

- Asynchronous message processing
- Real-time notifications and alerts
- Session state management
- Rate limiting and anti-spam controls

Observability Strategy

Runtime Monitoring

OpenTelemetry Integration:

- Distributed tracing across agent interactions
- Performance metrics for each specialized agent
- Error tracking and alerting for failed interactions
- Cost tracking per user interaction and agent type

Amazon CloudWatch Integration:

- System health dashboards
- Custom metrics for business KPIs (response time, user satisfaction, cost per interaction)
- Automated alerting for system anomalies
- Log aggregation and search capabilities

Security & Compliance Monitoring

- AWS Secrets Manager for API key rotation
- IAM-based access control with principle of least privilege
- Audit logging for all user interactions and data access
- Privacy-preserving analytics with automatic PII detection and masking

Agent State Management

Conversation Persistence

The system uses LangGraph's state management to maintain context across interactions with conversation memory architecture that includes full conversation history, user identification, routing decisions, and crisis state tracking.

Key Benefits:

- **Context Continuity:** Agents remember previous interactions and can build on past conversations
- **Multi-turn Support:** Complex queries can be resolved across multiple message exchanges

- **Emergency State Tracking:** Crisis mode persists across interactions until resolved
- **User Personalization:** Responses adapt based on user history and preferences

Tool Integration Architecture

Each specialized agent has access to domain-specific tools that provide real-world data and capabilities:

Spatial Tools: Location-based services for safety and job discovery **Legal Tools:** RAG-based search through labor law documents **Memory Tools:** User context retrieval and conversation history **Notification Tools:** WhatsApp message sending and emergency alerts **Community Tools:** Analytics and reporting for institutional users

This design ensures that EmpowerNet operates as a true "Digital Sisterhood" - combining the intelligence of AI with the warmth and practical support of a community network, while maintaining strict cost controls and safety mechanisms that make it sustainable and reliable for vulnerable users.