**System Components**

1. **Web Interface** (React/Vue.js)
2. **Mobile App** (React Native/Flutter)
3. **Backend API** (FastAPI/Flask)
4. **Local LLM** (Ollama/llama.cpp)
5. **Voice Processing** (Whisper for speech-to-text)
6. **Hardware Control** (GPIO interface for Raspberry Pi)

**Step-by-Step Development Plan**

**Phase 1: Development Environment Setup**

Docker Compose Development Setup

Code

*docker\_setup.txt*

**Phase 2: Project Structure**

Project Directory Structure

Code

*project\_structure.sh*

**Phase 3: Backend API Structure**

FastAPI Backend Main Application

Code

*backend\_main.py*

**Phase 4: Mobile App Structure**

React Native Mobile App Main Component

Code

*mobile\_app\_main.js*

**Development Roadmap**

**Phase 1: Initial Setup (Week 1-2)**

1. **Set up development environment**:

bash

git clone <your-repo>

cd ai-elephant-control

docker-compose up -d

1. **Install local LLM**:
   * Use Ollama with models like Llama 2 7B or Mistral 7B
   * Fine-tune for multilingual command understanding
2. **Basic web interface**:
   * Create React frontend with microphone integration
   * Implement command table with radio buttons

**Phase 2: Core Functionality (Week 3-4)**

1. **Voice processing pipeline**:
   * Integrate Whisper for speech-to-text
   * Implement LLM Supervisor agent
   * Create Language Understanding sub-agent
2. **Backend API development**:
   * FastAPI with WebSocket support
   * Database integration for feedback storage
   * Command execution simulation

**Phase 3: Mobile Development (Week 5-6)**

1. **React Native app**:
   * Voice recording functionality
   * Real-time communication with backend
   * Manual command interface
2. **Testing and optimization**:
   * Test multilingual voice recognition
   * Optimize for low-latency response

**Phase 4: Hardware Integration (Week 7-8)**

1. **Raspberry Pi setup**:
   * GPIO interface for servo/motor control
   * Docker deployment for ARM64
   * Hardware safety mechanisms
2. **Production deployment**:
   * System service configuration
   * Auto-startup on boot
   * Remote monitoring capabilities

**Key Technologies Stack**

**Backend**: FastAPI, PostgreSQL, Redis, Ollama **Frontend**: React, WebSocket, Web Speech API **Mobile**: React Native, Expo **AI/ML**: Whisper, Llama/Mistral, Custom fine-tuning **Hardware**: Raspberry Pi, GPIO, Servo controllers **Deployment**: Docker, Nginx, systemd

**Next Steps**

1. **Start with the Docker setup** I provided above
2. **Begin with voice recognition testing** using Whisper
3. **Implement the LLM pipeline** for command understanding
4. **Test multilingual capabilities** with your command dataset
5. **Gradually add hardware simulation** before actual GPIO integration