Proximo Agentic AI Architecture

System Overview

```
User Input → Orchestrator → Specialized Agents → Response Synthesis → Output
```

Core Agents

1. Conversation Agent

- **Model**: Fine-tuned Mistral-7B per friend personality
- **Function**: Generate contextual responses
- **Memory**: Vector database of conversation history
- **Tools**: LangChain, MemGPT for long-term memory

2. Emotion Detection Agent

- **Model**: RoBERTa-emotion or BERT-emotion
- **Function**: Analyze user emotional state
- **Input**: Text, voice tone analysis
- **Output**: Emotion classification + intensity

3. Personality Matching Agent

- **Model**: Custom-trained BERT for Big Five traits
- **Function**: Match response style to friend personality
- **Data**: Friend's communication patterns, linguistic markers
- **Output**: Personality-adjusted response parameters

4. Avatar Generation Agent

- **Options**:
  - **Paid**: HeyGen API ($99/month)
  - **Open Source**: SadTalker + Wav2Lip + Real-Time-Voice-Cloning
- **Function**: Generate talking avatar videos
- **Input**: Text response + friend's appearance data
- **Output**: MP4 video with lip-sync

5. **Voice Synthesis Agent**

- **Options**:
  - **Paid**: ElevenLabs ($5-330/month)
  - **Open Source**: Coqui TTS + OpenVoice
- **Function**: Clone friend's voice patterns
- **Input**: Voice samples from friend
- **Output**: Synthesized audio in friend's voice

6. **Memory Management Agent**

- **Tools**: ChromaDB + custom retrieval system
- **Function**: Store and retrieve conversation context
- **Features**:
  - Long-term personality consistency
  - Context-aware responses
  - Relationship progression tracking

Agent Orchestration Flow

Input Processing

```python
```

```python
class ProximoOrchestrator:
    def process_user_input(self, message, user_id, friend_id):
        # 1. Emotion Detection
        emotion = self.emotion_agent.analyze(message)

        # 2. Retrieve Context
        context = self.memory_agent.get_context(user_id, friend_id)

        # 3. Personality Parameters
        personality = self.personality_agent.get_traits(friend_id)

        # 4. Generate Response
        response = self.conversation_agent.generate(
            message, emotion, context, personality
        )

        # 5. Create Avatar
        avatar_video = self.avatar_agent.create_video(
            response, friend_id
        )

        # 6. Update Memory
        self.memory_agent.store_interaction(
            user_id, friend_id, message, response, emotion
        )

        return avatar_video, response
```

Response Generation Pipeline

```
User Message → Emotion Analysis → Context Retrieval →
Personality Loading → Response Generation → Avatar Creation →
Voice Synthesis → Video Compilation → User Delivery
```

Technical Implementation

Infrastructure Stack

- **Orchestration**: FastAPI + Celery for async processing
- **Agents**: Individual Docker containers per agent
- **Communication**: Redis for inter-agent messaging

- **Storage**: PostgreSQL + ChromaDB for vectors
- **Caching**: Redis for frequently accessed data

Agent Deployment

yaml

```yaml
services:
  orchestrator:
    image: proximo/orchestrator
    ports: ["8000:8000"]

  conversation-agent:
    image: proximo/conversation
    environment:
      - MODEL_PATH=/models/mistral-friend-{friend_id}

  avatar-agent:
    image: proximo/avatar
    environment:
      - USE_HEYGEN=${HEYGEN_API_KEY:+true}
      - SADTALKER_PATH=/models/sadtalker

  voice-agent:
    image: proximo/voice
    environment:
      - ELEVENLABS_KEY=${ELEVENLABS_KEY}
      - COQUI_MODEL_PATH=/models/coqui
```

Cost Optimization Strategy

Hybrid Approach: Open Source + Paid Tiers

- **Free Tier**: Open source tools only
  - SadTalker for avatars
  - Coqui TTS for voice
  - Self-hosted Mistral-7B
  - Cost: ~$200/month for VPS hosting
- **Premium Tier**: Best-in-class paid tools
  - HeyGen for high-quality avatars
  - ElevenLabs for premium voice cloning

  - Cost: ~$500/month + compute costs

Compute Requirements (Self-Hosted)

- **GPU**: RTX 4090 or A6000 (24GB VRAM)
- **RAM**: 64GB system memory
- **Storage**: 2TB NVMe SSD
- **Estimated Cost**: $5,000-8,000 hardware + $300/month electricity

Development Phases

Phase 1: Core Agents (Months 1-3)

- Build conversation agent with Mistral fine-tuning
- Implement basic emotion detection
- Create memory management system
- Simple text-based responses

Phase 2: Avatar Integration (Months 4-5)

- Integrate SadTalker for video generation
- Add voice synthesis with Coqui TTS
- Build friend data processing pipeline
- Basic avatar customization

Phase 3: Premium Features (Months 6-7)

- HeyGen API integration for paid users
- ElevenLabs voice cloning
- Advanced personality modeling
- Real-time conversation capabilities

Phase 4: Optimization & Scale (Months 8-9)

- Multi-agent load balancing
- Response caching and optimization
- Mobile app integration
- Performance monitoring

Key Advantages of Agentic Approach

1. **Modularity**: Easy to upgrade individual components
2. **Scalability**: Each agent can scale independently
3. **Cost Flexibility**: Mix open source and paid tools
4. **Rapid Development**: Leverage existing specialized tools
5. **Quality**: Best-in-class tools for each function
6. **Maintainability**: Cleaner separation of concerns