🧠 Multimodal Memory System

A sophisticated AI-powered memory system that captures, analyzes, and retrieves audio conversations with body language, emotional context, and environmental data.

🚀 Quick Start

Prerequisites

- Python 3.8+
- Node.js 16+
- FFmpeg (for audio processing)
- 8GB+ RAM recommended

Option 1: Docker Setup (Recommended)

```bash
# Clone the repository
git clone <your-repo-url>
cd memory-system

# Create environment file
cp backend/.env.example backend/.env
# Edit backend/.env with your API keys

# Start with Docker
docker-compose up --build

# Access the application
# Frontend: http://localhost:3000
# Backend API: http://localhost:8000
# API Docs: http://localhost:8000/docs
```

Option 2: Manual Setup

Backend Setup

```bash
```

```
cd backend

# Create virtual environment
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Set up environment
cp .env.example .env
# Edit .env with your configuration

# Start the backend
python memory_api.py
```

Frontend Setup

```bash
cd frontend

# Install dependencies
npm install

# Start development server
npm run dev

# Open http://localhost:3000
```

📁 Project Structure

```
memory-system/
├── backend/
│   ├── memory_utils.py         # Core utilities
│   ├── memory_api.py           # FastAPI service
│   ├── audio_memory_assistant.py # Audio processing
│   ├── requirements.txt        # Python dependencies
│   ├── .env                    # Environment variables
│   └── Dockerfile              # Backend container
├── frontend/
│   ├── src/
│   │   ├── components/
│   │   │   └── MemoryExplorer.jsx # Main React component
│   │   ├── App.jsx             # App entry point
│   │   └── App.css             # Styles
│   ├── package.json            # Node dependencies
│   ├── vite.config.js          # Vite configuration
│   ├── tailwind.config.js      # Tailwind CSS config
│   └── Dockerfile              # Frontend container
├── docker-compose.yml          # Docker setup
└── README.md                   # This file
```

🎯 Features

Core Capabilities

- **Audio Processing**: Speech-to-text with Whisper
- **Emotion Analysis**: Real-time sentiment detection
- **Memory Storage**: Vector database with semantic search
- **Analytics**: Comprehensive memory insights
- **Export**: JSON and CSV export formats

Multimodal Analysis

- **Body Language**: Gesture and posture recognition
- **Environmental Context:** Location and weather data
- **Biometric Integration**: Stress and engagement levels
- **Temporal Patterns**: Time-based memory analysis

User Interface

- **Memory Explorer**: Browse and search memories

- **Real-time Upload**: Drag-and-drop audio processing
- **Analytics Dashboard**: Visual insights and trends
- **Responsive Design**: Works on desktop and mobile

🔧 Configuration

Environment Variables (.env)

```bash
# API Configuration
API_HOST=0.0.0.0
API_PORT=8000
DEBUG=True

# Database
DATABASE_PATH=./memory_system.db

# OpenAI Integration (Optional)
OPENAI_API_KEY=your_openai_api_key_here

# Model Configuration
WHISPER_MODEL=base
EMBEDDING_MODEL=all-MiniLM-L6-v2
EMOTION_MODEL=j-hartmann/emotion-english-distilroberta-base
```

📝 Usage Examples

Python API Client

```python
```

```
import requests

# Upload audio file
with open('meeting.wav', 'rb') as f:
    files = {'audio_file': f}
    response = requests.post('http://localhost:8000/process-audio', files=files)
    result = response.json()
    print(f"Memory created: {result['memory_id']}")

# Search memories
search_data = {"query": "team meeting", "limit": 5}
response = requests.post('http://localhost:8000/memories/search', json=search_data)
memories = response.json()
print(f"Found {len(memories)} memories")

# Get analytics
response = requests.get('http://localhost:8000/analytics/summary')
stats = response.json()
print(f"Total memories: {stats['total_memories']}")
```

JavaScript Client

```
javascript
```

```
// Upload audio
const formData = new FormData();
formData.append('audio_file', audioFile);

const response = await fetch('http://localhost:8000/process-audio', {
  method: 'POST',
  body: formData
});

const result = await response.json();
console.log('Memory created:', result.memory_id);

// Search memories
const searchResponse = await fetch('http://localhost:8000/memories/search', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ query: 'project discussion', limit: 10 })
});

const memories = await searchResponse.json();
console.log('Found memories:', memories.length);
```

🧪 Testing

Test the API

```bash
# Health check
curl http://localhost:8000/health

# Upload test audio
curl -X POST "http://localhost:8000/process-audio" \
  -F "audio_file=@test.wav"

# Search memories
curl -X POST "http://localhost:8000/memories/search" \
  -H "Content-Type: application/json" \
  -d '{"query": "meeting", "limit": 5}'
```

Frontend Testing

```bash
```

```
cd frontend
npm test
```

🚀 Deployment

Production Docker

```bash
# Build for production
docker-compose -f docker-compose.prod.yml up --build

# With custom environment
OPENAI_API_KEY=your_key docker-compose up -d
```

Cloud Deployment

- **Heroku**: Use provided Procfile
- **AWS**: Deploy with ECS or Lambda
- **Google Cloud**: Use Cloud Run
- **Azure**: Container Instances

🔐 Security Notes

- Set strong API keys in production
- Use HTTPS in production environments
- Implement authentication for multi-user deployments
- Regular backup of memory database
- Consider data encryption for sensitive audio

🛠️ Development

Adding New Features

1. **Backend**: Add endpoints to `memory_api.py`
2. **Frontend**: Create components in `src/components/`
3. **Models**: Extend `memory_utils.py` for new data types

Custom Models

Replace existing models in `audio_memory_assistant.py`:

```python
# Use custom emotion model
self.emotion_analyzer = pipeline(
    "text-classification",
    model="your-custom-emotion-model"
)
```

📊 Performance

Typical Performance

- **Audio Processing**: 2-5 seconds per minute of audio
- **Search**: <100ms for semantic queries
- **Memory Storage**: 1MB per hour of conversation
- **Concurrent Users**: 10-20 (single instance)

Optimization Tips

- Use GPU for faster model inference
- Implement Redis caching for frequent queries
- Use PostgreSQL for larger deployments
- Enable audio compression for storage

🤝 Contributing

1. Fork the repository
2. Create feature branch: `git checkout -b feature-name`
3. Commit changes: `git commit -am 'Add feature'`
4. Push to branch: `git push origin feature-name`
5. Submit pull request

📄 License

MIT License - see LICENSE file for details

🆘 Support

- **Documentation**: Check the `/docs` endpoint

- **Issues**: GitHub Issues
- **Community**: Discussions tab

🎉 What's Next

- **Mobile App**: React Native implementation
- **Video Analysis**: Computer vision integration
- **Team Features**: Multi-user memory sharing
- **AI Insights**: Predictive analytics
- **Integrations**: Slack, Zoom, Teams connectors

Built with ❤️ using FastAPI, React, and cutting-edge AI models