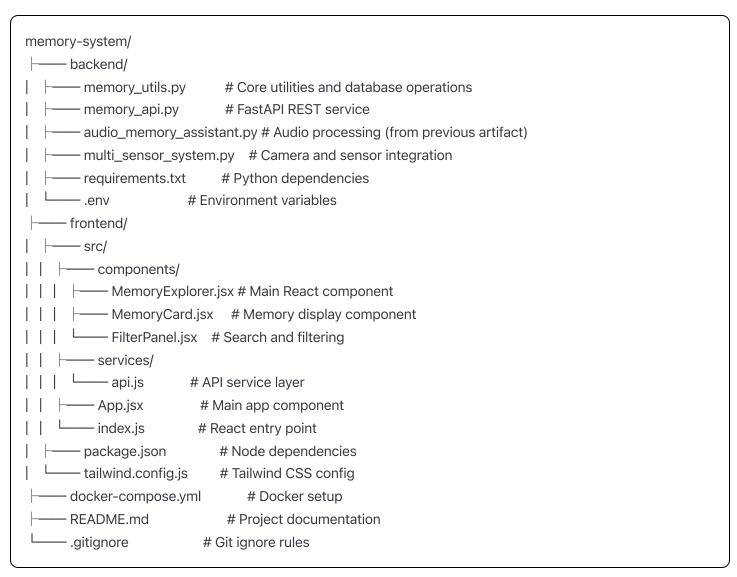
# Multimodal Memory System - Complete Setup

## Project Structure



## Quick Start

#### **Prerequisites**

- Python 3.8+
- Node.js 16+
- FFmpeg (for audio processing)

#### 1. Backend Setup

,				
	bash			,

```
# Navigate to backend

# Create virtual environment

python -m venv venv

source venv/bin/activate # On Windows: venv|Scripts|activate

# Install dependencies

pip install -r requirements.txt

# Create environment file

cp .env.example .env

# Edit .env with your API keys

# Start the API server

python memory_api.py
```

### 2. Frontend Setup

```
bash

# Navigate to frontend

cd frontend

# Install dependencies

npm install

# Start development server

npm start

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

# Dependencies

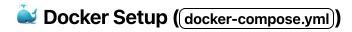
## **Backend Requirements (** (requirements.txt))

txt

```
# Core Framework
fastapi==0.104.1
uvicorn[standard]==0.24.0
python-multipart==0.0.6
# AI & ML Models
openai-whisper==20231117
sentence-transformers==2.2.2
transformers==4.35.2
torch==2.1.1
torchaudio==2.1.1
# Computer Vision
opency-python==4.8.1.78
mediapipe==0.10.7
# Data Processing
numpy==1.24.3
pandas==2.0.3
sqlite3
# Audio Processing
librosa==0.10.1
soundfile==0.12.1
# Topic Modeling
bertopic==0.15.0
umap-learn==0.5.4
hdbscan==0.8.33
# Utilities
python-dotenv==1.0.0
pydantic==2.5.0
python-dateutil==2.8.2
# Optional: Enhanced Features
chromadb==0.4.18 # Vector database
langchain==0.0.348 # LLM integration
openai==1.3.8 # GPT integration
```

## Frontend Dependencies (package.json)

```
"name": "memory-explorer",
 "version": "1.0.0",
 "dependencies": {
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "lucide-react": "^0.263.1",
  "axios": "^1.6.0",
  "date-fns": "^2.30.0",
  "recharts": "^2.8.0"
 },
 "devDependencies": {
  "@vitejs/plugin-react": "^4.0.3",
  "vite": "^4.4.5",
  "tailwindcss": "^3.3.3",
  "autoprefixer": "^10.4.15",
  "postcss": "^8.4.29"
},
 "scripts": {
 "dev": "vite",
 "build": "vite build",
 "preview": "vite preview"
}
}
```



yaml

```
version: '3.8'
services:
 backend:
 build:
   context: ./backend
   dockerfile: Dockerfile
 ports:
   - "8000:8000"
 environment:
   - DATABASE_PATH=/app/data/memory_system.db
   - OPENAI_API_KEY=${OPENAI_API_KEY}
 volumes:
   - ./data:/app/data
   - ./uploads:/app/uploads
 restart: unless-stopped
 frontend:
 build:
   context: ./frontend
   dockerfile: Dockerfile
 ports:
   - "3000:3000"
 environment:
   - REACT_APP_API_URL=http://localhost:8000
 depends_on:
  - backend
 restart: unless-stopped
volumes:
 memory_data:
```

# Configuration

## **Environment Variables (**(.env))

bash

```
# API Configuration
API_HOST=0.0.0.0
API_PORT=8000
DEBUG=True
# Database
DATABASE_PATH=./memory_system.db
# OpenAl 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
# Camera/Video Settings
ENABLE_CAMERA=True
CAMERA_DEVICE=0
VIDEO_WIDTH=640
VIDEO_HEIGHT=480
# Audio Settings
AUDIO_SAMPLE_RATE=16000
AUDIO_CHUNK_SIZE=1024
# Storage Settings
MAX_MEMORY_SIZE_MB=1000
AUTO_CLEANUP_DAYS=365
# Security
CORS_ORIGINS=["http://localhost:3000", "http://localhost:3001"]
MAX_FILE_SIZE_MB=100
```

# API Usage Examples

## JavaScript/React Integration

javascript

```
// services/api.js
class MemoryAPI {
 constructor(baseURL = 'http://localhost:8000') {
  this.baseURL = baseURL;
}
 async searchMemories(query) {
  const response = await fetch(`${this.baseURL}/memories/search`, {
   method: 'POST',
   headers: { 'Content-Type': 'application/json' },
   body: JSON.stringify({ query, limit: 10 })
  });
  return response.json();
 }
 async uploadAudio(file) {
  const formData = new FormData();
  formData.append('audio_file', file);
  const response = await fetch(`${this.baseURL}/process-audio`, {
   method: 'POST',
   body: formData
  });
  return response.json();
 }
 async getAnalytics() {
  const response = await fetch(`${this.baseURL}/analytics`);
  return response.json();
 }
}
export default MemoryAPI;
```

## **Python Client**

python

```
# client.py
import requests
import json
class MemoryClient:
  def __init__(self, base_url="http://localhost:8000"):
    self.base_url = base_url
  def search_memories(self, query, limit=10):
    response = requests.post(
      f"{self.base_url}/memories/search",
      json={"query": query, "limit": limit}
    return response.json()
  def upload_audio(self, file_path):
    with open(file_path, 'rb') as f:
      files = {'audio_file': f}
      response = requests.post(
         f"{self.base_url}/process-audio",
         files=files
    return response.json()
  def get_today_memories(self):
    response = requests.get(f"{self.base_url}/memories/today")
    return response.json()
# Usage
client = MemoryClient()
memories = client.search_memories("team meetings")
print(f"Found {len(memories)} memories about team meetings")
```

## Mobile Integration

#### **React Native Setup**

bash

```
# Install React Native CLI
npm install -g @react-native-community/cli

# Create new project
npx react-native init MemoryMobile

# Install dependencies
npm install axios react-native-audio-record react-native-fs
```

#### **Mobile API Service**

```
javascript
// MobileAPI.js
import axios from 'axios';
class MobileMemoryAPI {
 constructor() {
  this.api = axios.create({
   baseURL: 'http://YOUR_SERVER_IP:8000',
   timeout: 30000,
  });
 }
 async recordAndUpload(audioPath) {
  const formData = new FormData();
  formData.append('audio_file', {
   uri: audioPath,
   type: 'audio/wav',
   name: 'recording.wav',
  });
  const response = await this.api.post('/process-audio', formData, {
   headers: { 'Content-Type': 'multipart/form-data' },
  });
  return response.data;
}
```



#### **Backend Tests**

```
python
# tests/test_memory_utils.py
import pytest
from memory_utils import MemoryProcessor, MemoryFilter
def test_memory_storage():
  processor = MemoryProcessor(":memory:") # In-memory SQLite
  memory_data = {
    'id': 'test_001',
    'timestamp': 1234567890,
    'audio_text': 'Test memory',
    'emotion': 'joy',
    'enhanced_embedding': [0.1, 0.2, 0.3],
    'movement_data': {'engagement_level': 0.8},
    'context_data': {'biometric': {'stress_score': 0.3}},
    'importance_score': 0.7,
    'searchable_tags': ['test']
  }
  memory_id = processor.store_memory(memory_data)
  assert memory_id == 'test_001'
  memories = processor.get_memories(limit=1)
  assert len(memories) == 1
  assert memories[0]['audio_text'] == 'Test memory'
def test_memory_search():
  processor = MemoryProcessor(":memory:")
  # Add test memories and verify search functionality
  pass
```

#### **Frontend Tests**

javascript

```
// tests/MemoryExplorer.test.jsx
import { render, screen, fireEvent } from '@testing-library/react';
import MemoryExplorer from '../components/MemoryExplorer';

test('renders memory explorer', () => {
    render(<MemoryExplorer />);
    expect(screen.getByText('Memory Explorer')).toBeInTheDocument();
});

test('search functionality', async () => {
    render(<MemoryExplorer />);
    const searchInput = screen.getByPlaceholderText(/search memories/i);
    fireEvent.change(searchInput, { target: { value: 'team meeting' } });

const searchButton = screen.getByText('Search');
fireEvent.click(searchButton);

// Assert search results appear
});
```

# Deployment

## **Production Setup**

```
# Backend production
pip install gunicorn
gunicorn memory_api:app --host 0.0.0.0 --port 8000 --workers 4

# Frontend build
npm run build
serve -s build -I 3000
```

#### **Docker Production**

yaml			•

```
# docker-compose.prod.yml
version: '3.8'
services:
 backend:
  build: ./backend
  ports:
  - "8000:8000"
  environment:
  - ENV=production
  - DATABASE_URL=postgresql://user:pass@db:5432/memorydb
  restart: always
frontend:
  build: ./frontend
  ports:
  - "80:80"
  restart: always
 nginx:
 image: nginx:alpine
  ports:
  - "443:443"
  volumes:
   - ./nginx.conf:/etc/nginx/nginx.conf
   - ./ssl:/etc/nginx/ssl
  restart: always
```

# Monitoring & Analytics

#### **Health Checks**

python

```
# health.py
import requests
import time

def check_system_health():
    try:
        response = requests.get("http://localhost:8000/health")
        return response.status_code == 200
    except:
        return False

def monitor_memory_processing():
    # Check processing performance
    # Monitor database size
    # Track API response times
    pass
```

# **Security Considerations**

#### **Production Security**

```
# security.py
from fastapi.middleware.trustedhost import TrustedHostMiddleware
from fastapi.middleware.httpsredirect import HTTPSRedirectMiddleware

# Add to memory_api.py
app.add_middleware(TrustedHostMiddleware, allowed_hosts=["yourdomain.com"])
app.add_middleware(HTTPSRedirectMiddleware)

# File upload validation

def validate_audio_file(file):
    allowed_types = ["audio/wav", "audio/mp3", "audio/m4a"]
    if file.content_type not in allowed_types:
        raise HTTPException(400, "Invalid file type")

if file.size > 100 * 1024 * 1024: # 100MB limit
        raise HTTPException(400, "File too large")
```

This complete setup provides:

• **V** Production-ready utilities with proper error handling

- **RESTful API** with comprehensive endpoints
- **Modern React frontend** with real-time features
- **Ocker deployment** for easy scaling
- **Mobile integration** capabilities
- Security and monitoring considerations

The system is modular, scalable, and ready for both development and production deployment!