**LLM-Guided Reminiscence Therapy System**

**Comprehensive Documentation**

**Table of Contents**

1. [Introduction](#1-introduction)
2. [System Architecture](#2-system-architecture)
3. [Installation & Setup](#3-installation--setup)
4. [Core Components](#4-core-components)
5. [Processing Pipeline](#5-processing-pipeline)
6. [User Interaction Guide](#6-user-interaction-guide)
7. [Role-Playing Framework](#7-role-playing-framework)
8. [Azure OpenAI Integration](#8-azure-openai-integration)
9. [Local Mode Operation](#9-local-mode-operation)
10. [Performance Optimization](#10-performance-optimization)
11. [Troubleshooting](#11-troubleshooting)
12. [Future Development](#12-future-development)

**1. Introduction**

The LLM-Guided Reminiscence Therapy System is an innovative application of large language models to facilitate reminiscence therapy, a therapeutic approach that involves discussing past experiences and memories to improve psychological well-being. This system combines several advanced technologies:

* **Retrieval-Augmented Generation (RAG)** for personalized therapy based on user documents
* **Memory graph visualization** to track and connect life experiences
* **Emotional intelligence** to detect and respond appropriately to user feelings
* **Character role-playing** to enhance engagement, especially for younger users
* **Multimodal capabilities** for processing both text and images

The system is designed for therapeutic applications, research purposes, and personal use, though it is not intended to replace professional mental health care.

**Key Features**

* **Document processing**: Analysis of uploaded personal documents (letters, journals, etc.)
* **Memory extraction**: Identification of key people, places, events, and emotions
* **Memory graph**: Visual representation of connections between memories
* **Character personas**: Option to interact with different character-based therapists (e.g., Iron Man)
* **Multimodal input**: Support for both text conversations and image analysis
* **Session management**: Tracking progress across multiple therapy sessions

**2. System Architecture**

The system follows a modular architecture with six primary components:

Show Image

**High-Level Architecture**

1. **User Interface Layer** (Streamlit)
   * Chat interface
   * Document/image upload
   * Character selection
   * Memory graph visualization
   * Session controls
2. **Data Processing Layer**
   * Document analysis
   * Image processing
   * Memory extraction
   * Vector embedding
3. **Memory Management Layer**
   * Memory graph creation
   * Relationship mapping
   * Temporal organization
   * Graph visualization
4. **Emotional Intelligence Layer**
   * Emotion detection
   * Response strategy selection
   * Empathetic engagement
5. **Therapeutic Response Layer**
   * LLM integration
   * Context-aware prompting
   * Follow-up question generation
6. **Character Adaptation Layer**
   * Persona profiles
   * Language style adaptation
   * Response customization

**Data Flow**

1. User uploads documents/images and starts conversation
2. Documents are processed, analyzed, and converted to memory nodes
3. User messages are analyzed for emotional content
4. System retrieves relevant memories and context
5. LLM generates therapeutic responses
6. Character adaptation layer modifies responses if needed
7. User receives response through the interface

**3. Installation & Setup**

**Prerequisites**

* Python 3.8 or higher
* GPU support recommended for transformer models
* Azure OpenAI API access (optional but recommended)
* 4GB+ RAM

**Environment Setup**

1. **Create a dedicated virtual environment**
   * Using venv, conda, or another virtual environment tool
   * This isolates dependencies from other projects
2. **Install required packages**
   * Core packages: streamlit, langchain, transformers, torch, networkx
   * Document processing: pypdf, python-docx, pillow
   * Visualization: matplotlib, networkx
   * LLM integration: langchain-openai
3. **Configure Azure OpenAI (if using)**
   * Create Azure OpenAI resource
   * Deploy models (GPT-4 recommended)
   * Obtain API key, endpoint, and deployment details
4. **Environment Variables**
   * Set up a .env file with Azure OpenAI credentials
   * Configure system settings (paths, parameters)
5. **Windows Compatibility**
   * Add special handling for Unix modules on Windows
   * Set up compatibility with torch and other packages

**Folder Structure**

Copy

LLM-Guided-Reminiscence-Therapy/

├── Reminiscence\_Therapy\_System.py # Main system implementation

├── streamlit.py # Streamlit interface

├── Document\_and\_Image\_Processing.py # Document processing module

├── role\_playing.py # Character adaptation module

├── windows\_compatibility.py # Windows compatibility fixes

├── requirements.txt # Package dependencies

└── .env # Environment variables

**4. Core Components**

**AzureLLMProvider**

The AzureLLMProvider facilitates communication with Azure OpenAI services:

* **Functionality**: Manages connections to Azure's GPT models
* **Rate limiting**: Controls API request frequency to avoid throttling
* **Error handling**: Manages API errors and provides fallback mechanisms
* **Retry logic**: Implements exponential backoff for failed requests

**Memory Graph**

The Memory Graph creates and maintains a network of connected memories:

* **Node creation**: Creates memory nodes from documents and conversations
* **Relationship mapping**: Connects memories based on people, places, and time
* **Graph visualization**: Renders the memory network as an interactive visualization
* **Query capability**: Enables searching for related memories

**EmpatheticEngagement**

The EmpatheticEngagement module detects emotions and generates appropriate responses:

* **Emotion detection**: Identifies emotions in user messages
* **Strategy selection**: Chooses therapeutic approaches based on emotions
* **Response generation**: Creates empathetic, supportive responses
* **Therapeutic techniques**: Implements reflective listening, CBT, and other approaches

**DocumentProcessor**

The DocumentProcessor analyzes uploaded documents to extract personal information:

* **Text extraction**: Parses PDFs, Word documents, and text files
* **Chunking**: Divides documents into manageable pieces
* **Embedding**: Creates vector embeddings for semantic retrieval
* **Entity extraction**: Identifies people, places, dates, and events

**ImageAnalyzer**

The ImageAnalyzer processes uploaded images:

* **Image captioning**: Generates descriptions of image content
* **Memory prompting**: Creates questions related to images
* **Emotional analysis**: Detects potential emotional significance
* **Integration**: Connects images to the memory graph

**SessionManager**

The SessionManager maintains conversation state and progress:

* **History tracking**: Records conversation turns
* **Summarization**: Creates session summaries
* **Emotion tracking**: Monitors emotional trends
* **Progress assessment**: Evaluates therapeutic advancement

**PersonaAdapter**

The PersonaAdapter modifies responses to match character personas:

* **Character profiles**: Maintains personality traits and speaking styles
* **Language adaptation**: Transforms therapeutic responses to match persona
* **Style consistency**: Ensures consistent character portrayal
* **Therapeutic integrity**: Preserves therapeutic value while adapting style

**5. Processing Pipeline**

The system operates through a sequence of processing steps:

**1. Data Ingestion**

* **Document upload**: User uploads personal documents through the Streamlit interface
* **Format handling**: System identifies and processes various file formats
* **Initial parsing**: Text and metadata are extracted from documents
* **Image processing**: Photos are analyzed and described

**2. Memory Extraction**

* **Text chunking**: Documents are divided into processable segments
* **Entity recognition**: People, places, and dates are identified
* **Event detection**: Significant events are extracted from text
* **Semantic analysis**: Topics and themes are identified
* **Embedding creation**: Text chunks are converted to vector embeddings

**3. Memory Graph Construction**

* **Node creation**: Memory entities become nodes in the graph
* **Edge creation**: Relationships between memories are established
* **Temporal ordering**: Memories are organized chronologically
* **Significance weighting**: Memories are weighted by emotional significance
* **Visualization preparation**: Graph is prepared for rendering

**4. Conversation Processing**

* **Message reception**: User messages are received through the interface
* **Emotion detection**: Emotional content is analyzed
* **Context retrieval**: Relevant memories are retrieved from the graph
* **Strategy selection**: Appropriate therapeutic strategies are chosen

**5. Response Generation**

* **Context assembly**: User message, emotions, memories, and strategy are combined
* **Prompt creation**: A comprehensive prompt is created for the LLM
* **LLM generation**: Azure OpenAI generates a base response
* **Response verification**: Output is checked for appropriateness

**6. Character Adaptation**

* **Style matching**: Response is adapted to match character persona if selected
* **Language modification**: Vocabulary, tone, and references are adjusted
* **Metaphor inclusion**: Character-appropriate metaphors are incorporated
* **Consistency checking**: Final response is checked for therapeutic value

**7. Presentation & Feedback**

* **Response delivery**: Final response is shown to the user
* **Memory updating**: New information is added to the memory graph
* **Follow-up generation**: Potential follow-up questions are prepared
* **Session recording**: Interaction is recorded for future reference

**6. User Interaction Guide**

**Initial Setup**

1. **Launch the application**
   * Run the Streamlit interface
   * Connect to Azure OpenAI or initialize in minimal mode
2. **Configure settings**
   * Enter Azure OpenAI credentials if using
   * Select character persona (optional)
   * Adjust system parameters if needed
3. **Upload personal documents**
   * Select relevant documents for processing
   * Wait for analysis to complete
   * Verify successful processing

**Starting a Therapy Session**

1. **Initiate session**
   * Click "Start Session" button
   * View initial greeting from the therapist
   * Begin conversation
2. **Conversation flow**
   * Respond to therapist questions
   * Share personal memories and experiences
   * Upload images when relevant
   * View follow-up questions and suggestions
3. **Memory graph exploration**
   * Click "Visualize Memory Graph" to view connections
   * Explore relationships between memories
   * Use graph to guide further conversation

**Using Character Personas**

1. **Select character**
   * Choose from available character personas
   * Click "Set [Character] Persona"
   * Note change in conversation style
2. **Character interaction**
   * Engage with character-specific language and references
   * Experience therapeutic techniques through character lens
   * Request character change if desired

**Session Management**

1. **Session controls**
   * End session when finished
   * View session summary and emotional trends
   * Start new session when ready
2. **Document management**
   * Add new documents during ongoing therapy
   * Process additional materials as needed
   * Build comprehensive memory graph over time

**7. Role-Playing Framework**

**Character Persona Design**

The role-playing framework enhances engagement through character-based therapy:

1. **Character selection**
   * Iron Man (Tony Stark): Technological, witty, problem-solving approach
   * Captain America (Steve Rogers): Values-oriented, principled, empathetic
   * Spider-Man (Peter Parker): Relatable, humorous, community-focused
   * Wonder Woman (Diana Prince): Compassionate, direct, empowering
   * Batman (Bruce Wayne): Analytical, strategic, methodical
2. **Character attributes**
   * Personality traits: Key character qualities
   * Speaking style: Vocabulary, phrasing, idioms
   * Background: Character history and experiences
   * Therapeutic approach: How the character approaches helping others

**Response Adaptation Process**

1. **Base response generation**
   * Standard therapeutic response is created
   * Core therapeutic content is established
2. **Character-specific adaptation**
   * Language style is modified to match character
   * Vocabulary and metaphors are adjusted
   * Speech patterns are applied
   * Character catchphrases may be incorporated
3. **Therapeutic integrity verification**
   * Check that therapeutic value is maintained
   * Ensure emotional support is preserved
   * Verify questions and prompts remain effective

**Character Integration Benefits**

1. **Engagement enhancement**
   * Makes therapy more approachable for younger users
   * Creates novel, interesting interactions
   * Reduces stigma around mental health discussions
2. **Metaphor enrichment**
   * Characters provide unique perspectives
   * Character-specific metaphors offer new insights
   * Different worldviews expand therapeutic approaches
3. **Preference accommodation**
   * Users connect with familiar characters
   * Different personas suit different user preferences
   * Character switching allows for varied experiences

**8. Azure OpenAI Integration**

**Configuration Process**

1. **Azure OpenAI setup**
   * Create Azure OpenAI resource in Azure portal
   * Deploy GPT-4 model (recommended) or GPT-3.5
   * Note API key, endpoint URL, and deployment name
2. **System configuration**
   * Enter Azure credentials in .env file or UI
   * Set API version and deployment details
   * Configure temperature and other parameters
3. **Connection testing**
   * Verify successful connection
   * Test basic prompts
   * Confirm model capabilities

**Prompt Engineering**

1. **Therapeutic prompt structure**
   * Character persona information (if applicable)
   * Session context and history
   * User's emotional state
   * Relevant memories and documents
   * Therapeutic strategy instructions
   * Current user message
2. **Response strategies**
   * Reflective listening: For strong emotions
   * CBT: For negative thought patterns
   * Psychodynamic: For exploring connections
   * General therapy: For neutral exchanges
3. **Context management**
   * Efficient token usage
   * Key information prioritization
   * Memory summarization techniques

**Rate Limit Management**

1. **Tier limitations**
   * S0 tier: Lower rate limits
   * S1/S2 tiers: Higher capacity
   * Enterprise tiers: Maximum throughput
2. **Implementation strategies**
   * Request spacing with time delays
   * Exponential backoff for 429 errors
   * Batch processing for document analysis
   * Request prioritization for user interactions
3. **Quota increase options**
   * Azure quota increase request process
   * Upgrade paths for higher usage
   * Multi-region deployment considerations

**9. Local Mode Operation**

**Minimal API Mode**

The system can operate with reduced API dependency:

1. **Local emotion detection**
   * Transformer-based emotion classification
   * Rule-based fallback methods
   * Vocabulary-based emotion scoring
2. **Template-based responses**
   * Character-specific response templates
   * Emotion-appropriate response selection
   * Context-aware template filling
3. **Local memory processing**
   * Rule-based memory extraction
   * Simple graph construction
   * Basic visualization without API

**Offline Capabilities**

1. **Document processing**
   * Local text extraction
   * Basic entity recognition
   * Simple keyword matching
2. **Image analysis**
   * Local image captioning model
   * Basic visual content description
   * Filename-based context inference
3. **Response generation**
   * Pre-defined therapeutic responses
   * Template-based question generation
   * Character voice adaptation rules

**Hybrid Operation**

1. **Selective API usage**
   * Critical components use API
   * Non-critical use local processing
   * Balance between quality and rate limits
2. **Caching strategies**
   * Store common responses
   * Cache document analysis results
   * Reuse similar prompt outcomes
3. **Progressive enhancement**
   * Basic functionality without API
   * Enhanced features with API
   * Graceful degradation when rate limited

**10. Performance Optimization**

**Memory Efficiency**

1. **Document processing optimization**
   * Efficient chunking strategies
   * Metadata extraction before full processing
   * Progressive document loading
2. **Vector storage optimization**
   * Dimension reduction techniques
   * Index compression methods
   * Partial index loading
3. **Graph optimization**
   * Pruning less significant nodes
   * Edge reduction for visualization
   * Hierarchical memory organization

**Speed Improvements**

1. **Pipeline parallelization**
   * Concurrent document processing
   * Asynchronous API requests
   * Background memory graph updates
2. **Caching strategies**
   * LLM response caching
   * Document analysis caching
   * Emotion detection result storage
3. **Computation distribution**
   * GPU acceleration for transformers
   * CPU optimization for graph processing
   * Browser rendering for visualization

**Resource Management**

1. **API usage optimization**
   * Token count reduction
   * Prompt compression techniques
   * Response length management
2. **Local computation balancing**
   * Model size vs. performance tradeoffs
   * Selective feature enabling/disabling
   * Progressive loading of components
3. **Storage management**
   * Document archiving strategies
   * Session data compression
   * Memory graph serialization

**11. Troubleshooting**

**Common Issues**

1. **API connection problems**
   * Credential verification
   * Endpoint URL checking
   * API version compatibility
   * Rate limit handling
2. **Document processing errors**
   * File format compatibility
   * Character encoding issues
   * Size limitations
   * Parsing failures
3. **Memory graph issues**
   * Insufficient data for visualization
   * Relationship mapping errors
   * Temporal ordering problems
   * Visualization rendering issues
4. **UI/UX challenges**
   * Streamlit component compatibility
   * Browser compatibility
   * Interface responsiveness
   * State management

**Diagnostic Procedures**

1. **API diagnosis**
   * Test simple API requests
   * Check error codes and messages
   * Verify quota and rate limits
   * Test with postman or curl
2. **System verification**
   * Component isolation testing
   * Minimal configuration testing
   * Progressive feature enabling
   * Log analysis
3. **Performance diagnosis**
   * Memory profiling
   * CPU/GPU utilization monitoring
   * Request timing analysis
   * Component benchmarking

**Recovery Procedures**

1. **API recovery**
   * Implement exponential backoff
   * Switch to minimal mode
   * Reset connections
   * Clear cached credentials
2. **Document recovery**
   * Alternative format conversion
   * Chunking size adjustment
   * Manual text extraction
   * Progressive loading
3. **System recovery**
   * Component reinitialization
   * Clean session restart
   * Environment variable refresh
   * Model reloading

**12. Future Development**

**Enhanced Features**

1. **Advanced memory graph**
   * Hierarchical memory organization
   * Semantic relationship mapping
   * Memory importance scoring
   * User-editable graph
2. **Multimodal expansion**
   * Video processing capabilities
   * Audio memory recording
   * Drawing and art therapy integration
   * Virtual reality support
3. **Therapeutic methodology expansion**
   * Additional therapy approaches
   * Cultural adaptation capabilities
   * Age-specific intervention strategies
   * Goal-oriented therapy tracking

**Technical Improvements**

1. **Model enhancements**
   * Fine-tuned therapy-specific models
   * Smaller local models for offline use
   * Multi-language support
   * Domain-specific training
2. **Infrastructure updates**
   * Containerized deployment
   * Cloud-based operation
   * Mobile application version
   * Enterprise integration options
3. **Security enhancements**
   * End-to-end encryption
   * Anonymized data processing
   * HIPAA compliance capabilities
   * Secure document handling

**Research Opportunities**

1. **Effectiveness studies**
   * Comparative effectiveness research
   * User engagement metrics
   * Therapeutic outcome measurement
   * Long-term impact assessment
2. **Methodology development**
   * LLM-based therapy protocols
   * Digital mental health best practices
   * Character-based therapy frameworks
   * Reminiscence technique optimization
3. **Technical exploration**
   * Memory representation mechanisms
   * Emotion extraction improvements
   * Character adaptation algorithms
   * Multimodal therapy approaches

**Appendix A: Glossary**

* **RAG (Retrieval-Augmented Generation)**: Technology that enhances LLM responses with external information
* **Reminiscence Therapy**: Therapeutic approach using memory recall to improve mental wellbeing
* **Memory Graph**: Network representation of connected memories and relationships
* **Embedding**: Vector representation of text for semantic similarity
* **Character Persona**: Fictional personality adopted for therapeutic interaction
* **Azure OpenAI**: Microsoft's API for accessing OpenAI models
* **Multimodal**: Supporting multiple forms of input (text, images, etc.)
* **CBT (Cognitive Behavioral Therapy)**: Therapy focused on identifying negative thought patterns
* **Psychodynamic Therapy**: Therapy exploring past experiences and relationships
* **Reflective Listening**: Technique of restating and clarifying what someone has said