CuraAi API Documentation

Version: 1.0

Status: Production Ready

Base URL: https://curaaiteam-curaai.hf.space

Overview

CuraAi is a conversational AI system with sophisticated memory management capabilities. The API implements a dual-layer memory architecture that combines in-session contextual awareness with long-term emotional memory storage. This design enables the system to maintain continuity across conversations while preserving important user information based on emotional significance.

Performance Characteristics

The API exhibits different response times depending on system state. Initial requests to the service typically require approximately 10 seconds due to model initialization, memory retrieval operations, and embedding generation. Once the system is warmed up, subsequent requests complete in 5 to 8 seconds. This latency pattern reflects the combined overhead of model inference, memory embedding, and Pinecone vector database queries.

Authentication

CuraAi implements server-side authentication for all external service integrations, including the Grok language model and Pinecone vector database. Client applications do not need to provide or manage API keys. All authentication is handled transparently by the server.

Architecture

Memory System

The memory architecture utilizes a single Pinecone index with namespace isolation per user. Each user is assigned a unique namespace identified by their `user_id`, ensuring complete separation of memory data between users. The system maintains two distinct memory layers: an in-session RAM buffer that holds recent conversation context, and a long-term Pinecone storage layer for persistent memories.

Memory Intelligence

Memory storage is selective and emotion-weighted. The system analyzes each conversational exchange to determine memory worthiness based on several factors including personal identity information, emotional intensity, and thematic persistence. Memories are scored for importance, with higher scores given to emotionally significant exchanges. This weighting influences retrieval patterns, ensuring that meaningful memories surface more frequently in future conversations.

Persona Management

The system maintains a stable persona through controlled memory updates. Persona-shaping memories are updated gradually to prevent drift while allowing natural evolution over time. Session summarization prevents memory fragmentation by consolidating extended conversations into coherent summaries before storing them in long-term memory.

API Endpoints

Health Check Endpoint

GET /

This endpoint provides a simple health check to verify server availability and responsiveness.

Description

The health check endpoint returns a basic status confirmation. It can be used for monitoring, load balancer health checks, or verifying that the API is operational before making substantive requests.

Response

json:

```json
{
  "status": "ok"
}
```

Integration Example

python:

```python
import requests

response = requests.get("https://curaaiteam-curaai.hf.space/")
if response.json()["status"] == "ok":
    print("CuraAi API is operational")
```

javascript:

```javascript
fetch('https://curaaiteam-curaai.hf.space/')
  .then(response => response.json())
  .then(data => {
    if (data.status === 'ok') {
      console.log('CuraAi API is operational');
    }
  });
```

Conversational AI Endpoint

POST /ai-chat

This is the primary conversational endpoint for text-based interactions with CuraAi.

Description

The AI chat endpoint processes user messages with full memory awareness. When a message is received, the system appends it to the in-session buffer, retrieves relevant long-term memories based on semantic similarity and emotional weighting, and constructs a comprehensive system prompt that includes both recent context and historical memories. After generating a response through the Grok language model, the system analyzes the exchange to determine if new memories should be stored. Emotional intensity is scored, and qualifying interactions are persisted to Pinecone with appropriate importance weights. When the session buffer reaches capacity, the system automatically summarizes the conversation and stores the summary as a long-term memory, preventing context window overflow while preserving conversational continuity.

Request Body

json:
```
{
  "user_id": "string",
  "session_id": "string",
  "query": "string"
}
```

Field Specifications:

- user_id (required): A unique identifier for the user. This value determines the Pinecone

namespace for memory isolation. The same user_id should be used across all sessions for a given user to maintain memory continuity.
- session_id (required): An identifier for the current conversation session. Generate a new session_id for each distinct conversation. This allows the system to maintain separate in-session memory buffers for concurrent or sequential conversations.
- query (required): The user’s message text. This is the conversational input that will be processed by the system.

Response

json:
```json
{
  "reply": "string"
}
```

The response contains the AI-generated reply that incorporates relevant memories and maintains persona consistency.

Integration Example

python:
```python
import requests

url = "https://curaaiteam-curaai.hf.space/ai-chat"
payload = {
    "user_id": "user_12345",
    "session_id": "session_abc_001",
    "query": "How are you doing today?"
}

response = requests.post(url, json=payload)
result = response.json()
print(result["reply"])
```

javascript:

```javascript
const url = 'https://curaaiteam-curaai.hf.space/ai-chat';
const payload = {
  user_id: 'user_12345',
  session_id: 'session_abc_001',
  query: 'How are you doing today?'
};

fetch(url, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(payload)
})
  .then(response => response.json())
  .then(data => {
    console.log(data.reply);
  });
```

bash:

```bash
curl -X POST "https://curaaiteam-curaai.hf.space/ai-chat" \
  -H "Content-Type: application/json" \
  -d '{
    "user_id": "user_12345",
    "session_id": "session_abc_001",
    "query": "How are you doing today?"
  }'
```

Multimodal Input Endpoint

POST /multimodal

This endpoint handles image and audio inputs, optionally accompanied by text messages.

Description

The multimodal endpoint extends CuraAi's capabilities beyond text to support visual and audio inputs. When a file is submitted, the system processes it according to its type. Images undergo optical character recognition to extract text content, while audio files are transcribed to text. The extracted or transcribed content is then combined with any optional text message provided by the user. This combined input is processed through the same conversational pipeline as text-only messages, with full access to memory retrieval, persona shaping, and memory storage capabilities. This approach ensures that multimodal interactions benefit from the same contextual awareness and continuity as text-based conversations.

Request Format

This endpoint accepts `multipart/form-data` submissions with the following fields:

- user_id (text, required): The unique user identifier for memory namespace selection.
- session_id (text, required): The current session identifier for in-session memory management.
- file (file, required): An image or audio file. Supported formats include common image types for OCR processing and audio formats for transcription.
- text (text, optional): An accompanying text message that provides context or additional information related to the file.

Response

json:
```
{
  "reply": "string"
}
```

The response contains the AI-generated reply based on the processed file content and optional text input.

Integration Example

python:

```python
import requests

url = "https://curaaiteam-curaai.hf.space/multimodal"
files = {
    'file': open('document.jpg', 'rb')
}
data = {
    'user_id': 'user_12345',
    'session_id': 'session_abc_001',
    'text': 'What does this document say?'
}

response = requests.post(url, files=files, data=data)
result = response.json()
print(result["reply"])
```

javascript:

```javascript
const url = 'https://curaaiteam-curaai.hf.space/multimodal';
const formData = new FormData();
formData.append('user_id', 'user_12345');
formData.append('session_id', 'session_abc_001');
formData.append('file', fileInput.files[0]);
formData.append('text', 'What does this document say?');

fetch(url, {
  method: 'POST',
  body: formData
})
```

```
  .then(response => response.json())
  .then(data => {
    console.log(data.reply);
  });
```

bash:

```
curl -X POST "https://curaaiteam-curaai.hf.space/multimodal" \
  -F "user_id=user_12345" \
  -F "session_id=session_abc_001" \
  -F "file=@document.jpg" \
  -F "text=What does this document say?"
```

Memory System Details

In-Session Memory

The in-session memory layer maintains the most recent six messages in RAM. This buffer is automatically included in every prompt construction, providing immediate conversational context. The buffer is session-specific and clears when the session terminates. This design ensures that the system can reference very recent exchanges without requiring vector database queries, reducing latency for contextual responses.

Long-Term Memory Storage

Long-term memories are stored in Pinecone when specific criteria are met. The system persists memories when users express identity information or personal facts, when emotional intensity exceeds a defined threshold, or when persistent themes emerge across multiple conversational turns. This selective storage approach prevents memory pollution while capturing genuinely significant information.

Emotional Weighting

Memory importance is calculated dynamically based on emotional intensity detected in the conversation. Low-intensity exchanges receive an importance score around 0.3, medium-intensity interactions score approximately 0.6, and high-intensity exchanges approach a maximum importance of 1.0. These importance scores directly influence retrieval patterns, causing emotionally significant memories to surface more frequently in relevant contexts.

Persona Memory Evolution

Persona-defining memories represent the highest tier of importance. These memories shape the AI’s understanding of the user’s preferences, communication style, and relationship dynamics. The system updates persona memories gradually to maintain stability while allowing natural evolution over extended interactions. This controlled update mechanism prevents sudden persona shifts that could disrupt user experience.

Session Summarization

When an in-session buffer reaches maximum capacity, the system triggers an automatic summarization process. Recent conversational context is condensed into a coherent summary that captures key themes, decisions, and emotional highlights. This summary is then stored as a long-term memory, effectively compressing multiple exchanges into a single retrievable unit. This approach prevents memory fragmentation and maintains retrieval efficiency even across very long conversations.

Error Responses

Invalid Input Error

HTTP 400 Bad Request

Json:
```
{
  "detail": "Unsupported file type"
```

}

This error occurs when the multimodal endpoint receives a file type that cannot be processed, or when required fields are missing from a request.

Internal Server Error

HTTP 500 Internal Server Error

json:
```
{
  "detail": "Internal error"
}
```

This error indicates an unexpected server-side failure during request processing. Such errors may occur due to downstream service unavailability, model inference failures, or database connection issues.

Security and Safety Guarantees

The CuraAi system implements several critical safety mechanisms. The system prompt is immutable and cannot be overridden through user input, ensuring consistent behavior and preventing prompt injection attacks. Persona definitions are protected through controlled update mechanisms that prevent sudden drift or manipulation. Memory isolation is strictly enforced through Pinecone namespaces, guaranteeing that no user can access or influence another user's memory data. All responses are grounded in emotional intelligence principles, maintaining appropriate boundaries and therapeutic value even under adversarial pressure.

Best Practices for Integration

When integrating with CuraAi, generate a new `session_id` for each distinct conversation while reusing the same `user_id` across all sessions for a given user. This practice ensures proper memory continuity while maintaining clear session boundaries. Do not attempt to store or manage memory data client-side, as the server manages all memory operations with sophisticated weighting and retrieval algorithms. Allow CuraAi's internal systems to handle conversational continuity rather than implementing custom context management, as the emotion-weighted memory system provides superior results compared to simple message buffering.

Scalability Considerations

The architecture is designed for efficient scaling within Pinecone's free tier constraints. Using a single index with namespace-based isolation minimizes costs while maintaining strict user separation. Emotion-weighted retrieval reduces noise in memory queries by surfacing only the most relevant historical context. Session summarization controls memory growth by preventing unbounded accumulation of conversational fragments. These design decisions collectively enable the system to support substantial user bases without requiring premium vector database tiers.

System Status

The CuraAi API is production-ready and suitable for deployment in applications requiring emotionally intelligent conversational AI with persistent memory. The system operates successfully within Pinecone's free tier limitations, implements comprehensive emotional safety mechanisms, and provides scalable memory management capable of supporting growing user populations.