RAGme.io: Personal RAG Agent for Web Content

A Comprehensive Overview

🎯 What is RAGme.io?

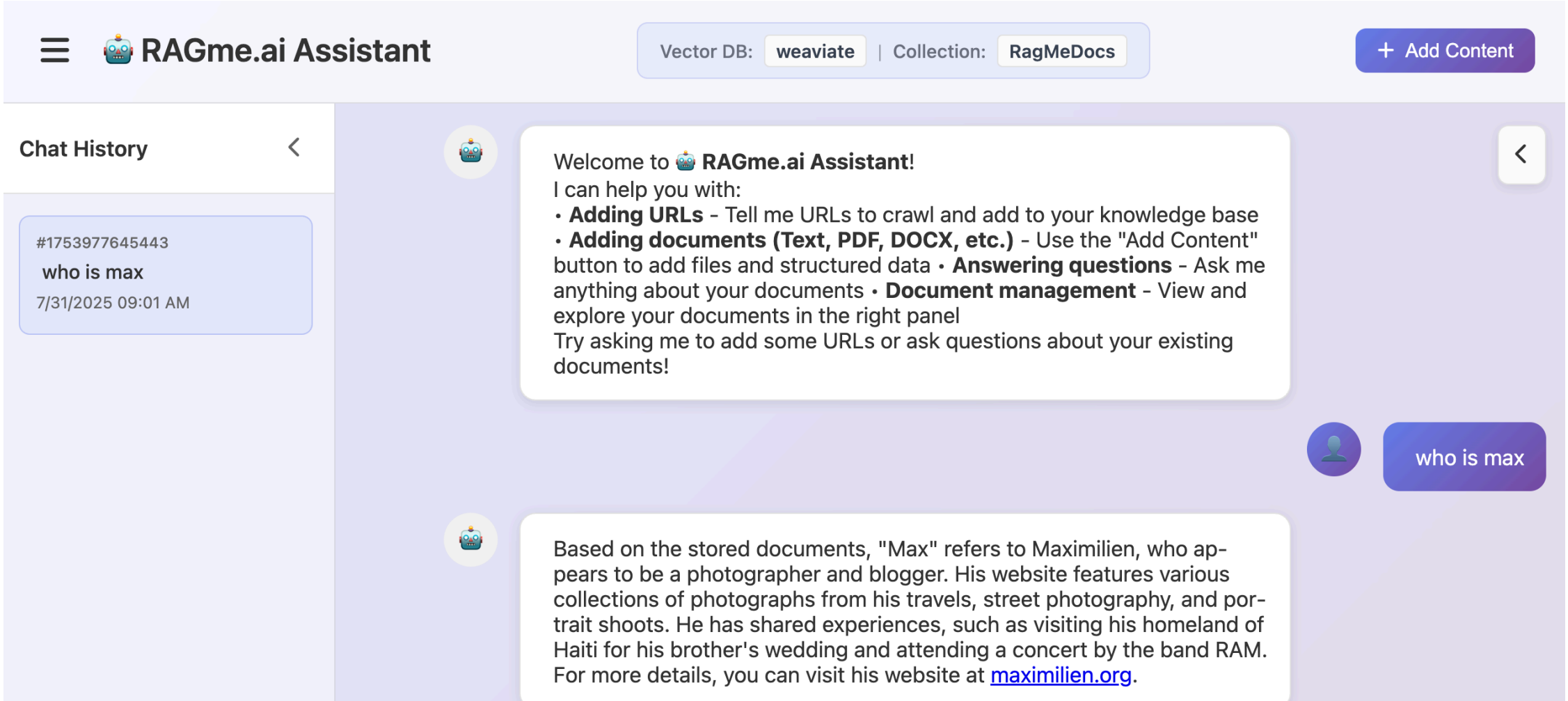
RAGme.io is a personalized agent that uses Retrieval-Augmented Generation (RAG) to process websites and documents you care about, enabling intelligent querying through an LLM agent.

Core Concept

- **RAG**: Combines document retrieval with AI generation
- **Personal**: Focuses on your specific content and interests
- **Agentic**: Uses LLM agents for intelligent interaction
- **Multi-modal**: Supports web pages, PDFs, and DOCX documents
- **Vector Database Agnostic**: Supports multiple vector databases (Milvus, Weaviate, etc.)

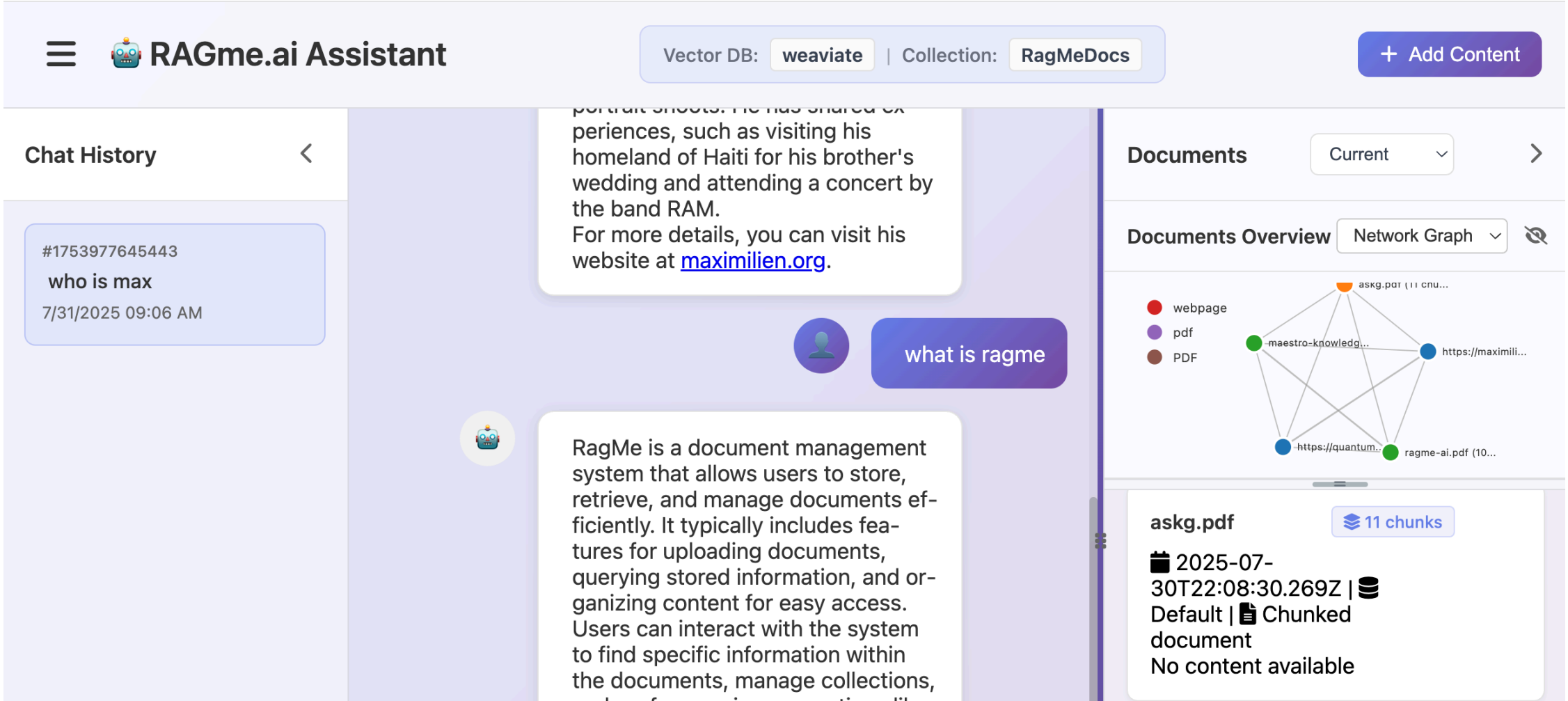
🖼️ New Frontend Interface

Modern Three-Pane Layout



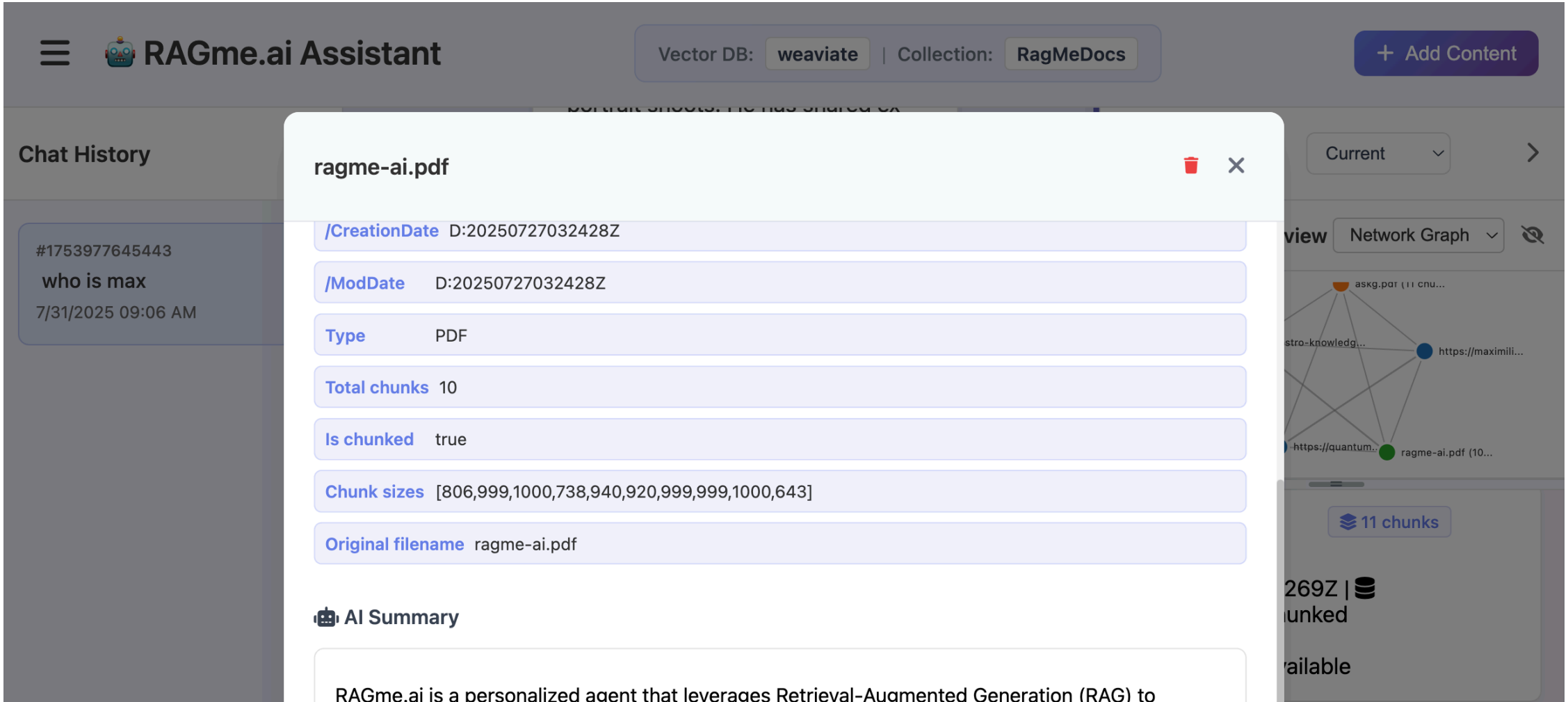
Document Management & Visualizations

Interactive Document Dashboard



📈 Data Visualization & Analytics

D3.js Powered Charts



🚀 Key Features & Use Cases

1. Interactive Personal RAG

- Add websites and documents (PDFs and DOCX)
- Query using natural language
- Get intelligent responses based on your content
- **Smart document chunking** for large files

2. Content Collection & Processing

- **Web Crawling**: Automatically discover and process web pages
- **Document Processing**: PDF and DOCX file ingestion with automatic chunking
- **Watch Directory**: Automatic processing of new files with consistent chunking
- **Chrome Extension**: One-click web page capture
- **Unified Processing**: Same chunking logic across all input methods

🏗️ Architecture Overview

Multi-Service Architecture with Three-Agent System

```
flowchart TB
    new-ui[New Frontend<br/>Port 3020] --> api[API Server<br/>Port 8021]
    new-ui[New Frontend<br/>Port 3020] --> ragme-agent[RagMeAgent<br/>Dispatcher]
    chrome[Chrome Extension] --> api
    subgraph "AI Agent Layer"
      agent-local[File Monitor Local Agent] --> mcp[MCP Server<br/>Port 8022]
      subgraph "Three-Agent System"
        ragme-agent --> functional-agent[FunctionalAgent<br/>Tool Operations]
        ragme-agent --> query-agent[QueryAgent<br/>Content Queries]
        functional-agent --> ragme-tools[RagMeTools<br/>Tool Collection]
        query-agent --> openai[OpenAI LLM]
      end
    end
    mcp --> api
    api --> ragme[RAGme Core]
    ragme --> vector-db[(Vector DB)]
    subgraph "Vector Database Layer"
        vector-db --> weaviate-local[(Local Weaviate<br/>Podman)]
        vector-db --> weaviate-cloud[(Weaviate Cloud)]
```

🎮 Usage Examples

1. Web Content Processing

```
# Add web pages to collection
"Crawl my https://example-blog.com up to 10 posts and add to my collection"

# Query the content
"What are the main topics discussed in the blog posts?"
```

2. Document Analysis

```
# Add PDF/DOCX to watch_directory/
# Automatically processed and indexed

# Query documents
"Summarize the key findings from the research papers"
```

3. Current Affairs

🔌 API Endpoints

Content Ingestion

```
# Add URLs
POST /add-urls
{
  "urls": ["https://example.com", "https://example.org"]
}

# Add JSON content
POST /add-json
{
  "data": {"content": "..."},
  "metadata": {"source": "..."}
}
```

Querying

```
# Query the collection
```

🛠️ Development Features

New Frontend UI ⭐ DEFAULT

```
// Real-time WebSocket communication
socket.emit('chat_message', { message: userInput });
socket.on('chat_response', (data) => {
    displayResponse(data.response);
});

// Document visualization with D3.js
const chart = d3.select('#document-chart')
    .append('svg')
    .attr('width', width)
    .attr('height', height);
```

Chrome Extension

```
// popup.js - Page capture functionality
async function captureCurrentPage() {
```

📊 Data Flow

1. Content Ingestion

```
flowchart LR
    A[Web Page/PDF/DOCX] --> B[Parser]
    B --> C[Text Extraction]
    C --> D[Chunking]
    D --> E[Embedding]
    E --> F[Vector DB]
```

2. Query Processing

```
flowchart LR
    A[User Query] --> B[Query Embedding]
    B --> C[Vector Search]
    C --> D[Retrieve Documents]
    D --> E[LLM Context]
    E --> F[Generate Response]
```

💡 Use Case Scenarios

Scenario 1: Research Assistant

```
User: "I'm researching quantum computing. Add these papers to my collection."
RAGme: "I've added 5 research papers. What specific aspects would you like to explore?"
User: "What are the main challenges in quantum error correction?"
RAGme: "Based on your papers, the main challenges are..."
```

Scenario 2: News Aggregator

```
User: "Add today's tech news articles about AI"
RAGme: "I've crawled and added 15 articles from tech news sites."
User: "What are the emerging AI trends this week?"
RAGme: "Based on the articles, the key trends are..."
```

Scenario 3: Document Manager

```
User: *drops PDF into watch_directory*
RAGme: "New document detected and processed: quarterly_report.pdf"
```

🎯 Key Benefits

For Individuals

- **Personalized Knowledge Base**: Your own curated content collection
- **Intelligent Search**: Natural language queries across all your content
- **Automated Processing**: Seamless ingestion of various content types
- **Insight Generation**: AI-powered analysis and summaries
- **Modern Interface**: Beautiful, responsive web interface with real-time features

For Organizations

- **Document Intelligence**: Extract insights from internal documents
- **Research Efficiency**: Rapid analysis of large document collections
- **Knowledge Discovery**: Find connections across different content sources
- **Scalable Architecture**: Multi-service design for enterprise deployment

🔧 Technical Highlights

Performance Optimizations

- **Batch Processing**: Efficient document ingestion
- **Vector Indexing**: Fast similarity search
- **Async Operations**: Non-blocking API responses
- **Memory Management**: Proper cleanup and resource handling
- **Real-time Updates**: WebSocket-based live communication

Extensibility

- **Modular Design**: Easy to add new content types
- **Plugin Architecture**: MCP server for document processing
- **API-First**: RESTful interfaces for integration
- **Vector Database Agnostic**: Support for multiple database backends

🚀 Future Roadmap

Phase 1: Infrastructure ✅ COMPLETED

- [x] ~~Decouple Weaviate dependency~~ **Completed!** Now supports Milvus, Weaviate, and extensible for others
- [x] ~~Add modern frontend UI~~ **Completed!** New three-pane interface with real-time features
- [x] ~~Add local Weaviate support~~ **Completed!** Podman-based local deployment
- [x] ~~Add debugging and monitoring tools~~ **Completed!** Comprehensive log monitoring
- [ ] Decouple LlamaIndex (docling integration)
- [ ] Add HTTPS security

Phase 2: Content Types

- [ ] Image and video processing

📈 Getting Started Guide

For detailed setup instructions, see the main README.md in the project root.

Quick Start

```
# Clone and setup
gh repo clone maximilien/ragme-io
cd ragme-io
uv venv
source .venv/bin/activate
uv sync --extra dev

# Configure environment
cp env.example .env
# Edit .env with your API keys

# Start all services (new frontend by default)
./start.sh
```

Access Points

🤝 Contributing

For detailed contribution guidelines, see CONTRIBUTING.md.

How to Help

- **Bug Reports**: Open issues for problems
- **Feature Requests**: Suggest new capabilities
- **Code Contributions**: Submit pull requests
- **Documentation**: Improve guides and examples

Development Setup

```
# Install development dependencies
uv sync --extra dev

# Run tests
./test.sh
```

📞 Support & Resources

Documentation

- **README.md**: Comprehensive setup guide
- **Vector Database Abstraction**: Database architecture guide
- **Process Management**: Service management guide
- **Troubleshooting**: Common issues and solutions
- **API Documentation**: Available at `/docs` when API server is running

Community

- **GitHub**: https://github.com/maximilien/ragme-io
- **Issues**: Bug reports and feature requests
- **Discussions**: Community support and ideas

🎉 Conclusion

RAGme.io represents a powerful approach to personal knowledge management:

- 🔍 **Intelligent Content Discovery**: Automatically process and index your content
- 🤖 **AI-Powered Insights**: Get intelligent responses from your personal knowledge base
- 🔄 **Seamless Integration**: Multiple ways to add and interact with content
- 📈 **Scalable Architecture**: Built for growth and customization
- 🎨 **Modern Interface**: Beautiful, responsive web interface with real-time features
- 🔧 **Flexible Deployment**: Support for multiple vector databases and deployment options

Ready to build your personal AI knowledge assistant?

Thank you for your attention! Questions and feedback welcome.