Pazago Drive — RAG Assignment: Berkshire Hathaway Intelligence

🎯 Assignment Overview

Build a **complete RAG application** using the Mastra framework that can intelligently answer questions about Warren Buffett's investment philosophy using Berkshire Hathaway shareholder letters with real-time streaming responses.

Duration: 6-8 hours *(This may vary based on your familiarity with AI concepts and TypeScript. Focus on building a working system first, then add enhancements if time permits.)*

Difficulty: Advanced

Primary Resource: Mastra Documentation

LLM Model: OpenAI GPT-4o

📚 What You'll Build

A **production-ready RAG application** built using Mastra's comprehensive framework featuring:

- **Document Processing**: Using Mastra's MDocument class and ETL pipeline
- **Vector Storage**: Leveraging Mastra's unified vector database interfaces
- **Intelligent Agents**: AI agents with persistent memory and conversation management
- **Streaming Responses**: Real-time responses using Mastra's streaming capabilities
- **Web Interface**: Chat application using Mastra's frontend integration
- **Source Attribution**: Transparent citations using Mastra's metadata system

🏗️ System Architecture

Follow Mastra's architecture using the five core primitives:

```
Frontend ←→ Mastra Agents ←→ RAG System ←→ Vector Storage
   ↓           ↓               ↓            ↓
Chat UI    Memory & Tools   MDocument    Database
Streaming  Workflows        Processing   (PostgreSQL/Pinecone)
```

Core Components:

- **Agents**: Autonomous AI entities with GPT-4o integration
- **Workflows**: Document processing and retrieval pipelines
- **RAG**: Complete ETL for Berkshire Hathaway documents
- **Memory**: Persistent conversation management
- **Tools**: Vector search and document retrieval functions

📊 Data Source

Berkshire Hathaway Annual Shareholder Letters (2019-2024)

- **Download from:** Google Drive - Berkshire Hathaway Letters
- **Format:** PDF files containing Warren Buffett's annual letters
- **Content:** Investment philosophy, business analysis, economic insights, company performance

Important: Download all available letters from the provided Google Drive link before beginning the assignment. These documents will serve as the knowledge base for your RAG system.

Sample Questions Your System Should Handle:

- "What does Warren Buffett think about cryptocurrency?"
- "How has Berkshire's investment strategy evolved over the past 5 years?"
- "What companies did Berkshire acquire in 2023?"
- "What is Buffett's view on market volatility and timing?"

- "How does Buffett evaluate management quality in potential investments?"

🚀 Implementation Guide

Primary Resource: Use Mastra Documentation for all implementation details

Phase 1: Project Initialization with Mastra CLI

Task 1.1: Setup Mastra Project

- Use `npx create-mastra@latest` to initialize the project
- Select RAG, Agents, and Workflows components during setup
- Configure OpenAI GPT-4o as the primary LLM provider
- Choose your preferred vector database (PostgreSQL with pgvector recommended)
- Follow Mastra's installation guide for complete setup

Task 1.2: Environment Configuration

- Configure OpenAI API key for GPT-4o model access
- Set up vector database connection parameters per Mastra docs
- Configure development environment following Mastra's setup guide
- Test the playground at `http://localhost:4111` for development workflow

Phase 2: Document Processing with Mastra RAG

Task 2.1: Document Ingestion

- Download all Berkshire Hathaway letters from the provided Google Drive link
- Use PDF parser to parse document before using `MDocument`
- Use Mastra's `MDocument` class to process the downloaded PDFs
- Implement document loading following Mastra's RAG documentation
- Configure chunking strategy using Mastra's built-in options
- Set appropriate chunk sizes and overlap for financial documents

Task 2.2: Vector Database Setup

- Configure vector storage using Mastra's unified database interface
- Set up embedding generation with OpenAI's text-embedding models
- Create vector indexes following Mastra's vector database guide
- Test document storage and retrieval functionality

Phase 3: Agent Development

Task 3.1: Create RAG Agent

- Build intelligent agent using Mastra's agent architecture
- Configure GPT-4o model integration through Mastra's LLM setup
- Implement system prompts for financial document expertise
- Add persistent memory for conversation continuity

Sample Agent Instructions:

```
You are a knowledgeable financial analyst specializing in Warren Buffett's inve
stment philosophy and Berkshire Hathaway's business strategy. Your expertis
e comes from analyzing years of Berkshire Hathaway annual shareholder lette
rs.

Core Responsibilities:
- Answer questions about Warren Buffett's investment principles and philosop
hy
- Provide insights into Berkshire Hathaway's business strategies and decision
s
- Reference specific examples from the shareholder letters when appropriate
- Maintain context across conversations for follow-up questions

Guidelines:
- Always ground your responses in the provided shareholder letter content
- Quote directly from the letters when relevant, with proper citations
- If information isn't available in the documents, clearly state this limitation
- Provide year-specific context when discussing how views or strategies evol
ved
- For numerical data or specific acquisitions, cite the exact source letter and y
ear
- Explain complex financial concepts in accessible terms while maintaining ac
curacy

Response Format:
- Provide comprehensive, well-structured answers
- Include relevant quotes from the letters with year attribution
- List source documents used for your response
- For follow-up questions, reference previous conversation context appropriat
ely

Remember: Your authority comes from the shareholder letters. Stay grounded
in this source material and be transparent about the scope and limitations of y
our knowledge.
```

Task 3.2: Implement Tools

- Create vector search tools using Mastra's tool system
- Implement document retrieval functions with metadata filtering
- Configure tools for financial document-specific queries
- Test tool integration within agent workflows

Phase 4: Frontend Integration

Task 4.1: Chat Interface Development

- Build chat interface using Mastra's frontend integration guides
- Implement streaming response handling per Mastra documentation
- Create message components with source citation display
- Add conversation management features

Task 4.2: User Experience Features

- Implement real-time streaming using Mastra's streaming capabilities
- Add conversation memory visualization
- Create source document display with clickable citations
- Handle error states and loading indicators

Phase 5: Testing & Deployment

Task 5.1: Application Testing

- Test RAG pipeline using Mastra's development playground
- Validate conversation memory and context preservation
- Verify source attribution accuracy
- Test streaming response performance

Task 5.2: Production Preparation

- Configure deployment following Mastra's deployment guides
- Set up monitoring and observability using Mastra's built-in features
- Implement error handling and recovery mechanisms
- Document setup and configuration process

📝 Required Deliverables

1. Complete Mastra RAG Application

- [ ] Functional Mastra application initialized with `create-mastra`
- [ ] Document processing using MDocument class and Mastra's RAG pipeline
- [ ] RAG agent with GPT-4o integration and proper tool configuration
- [ ] Persistent memory system for conversation continuity
- [ ] Vector storage with appropriate database integration

2. Chat Interface

- [ ] Working chat interface with streaming response display
- [ ] Conversation management with memory persistence
- [ ] Source citation display using Mastra's metadata system
- [ ] Error handling and loading states following Mastra patterns
- [ ] Responsive design compatible with Mastra's frontend approach

3. RAG Implementation

- [ ] Document ingestion pipeline using Mastra's document processing
- [ ] Vector search with retrieval enhancement (re-ranking if implemented)
- [ ] Source attribution with proper document metadata
- [ ] Conversation context maintenance using Mastra's memory system
- [ ] Tool integration for document search and analysis

4. Documentation & Configuration

- [ ] Complete setup instructions using Mastra CLI commands
- [ ] Environment configuration following Mastra standards
- [ ] Agent configuration with sample instructions
- [ ] Testing guide using Mastra's development playground
- [ ] Deployment notes based on Mastra's deployment options

🧪 Testing Requirements

Functional Testing (Following Mastra Development Workflow)

1. **Document Processing**
   - [ ] Successfully download documents from provided Google Drive link
   - [ ] Process documents using Mastra's MDocument class
   - [ ] Generate and store embeddings using Mastra's vector integration

- [ ] Verify vector search returns relevant results using Mastra's search tools

2. **Agent & Memory System**
    - [ ] Agent responds appropriately using GPT-4o integration
    - [ ] Conversation memory persists across interactions
    - [ ] Sources are cited using Mastra's metadata system
    - [ ] Tool integration functions correctly within agent workflows
3. **User Interface**
    - [ ] Messages stream using Mastra's streaming implementation
    - [ ] Interface follows Mastra's frontend integration patterns
    - [ ] Error states handled per Mastra's error handling approach
    - [ ] Conversations managed using Mastra's memory features

Sample Test Cases

Query 1: "What is Warren Buffett's investment philosophy?"

- Should return comprehensive response with citations from shareholder letters

Query 2: "Can you elaborate on his views about diversification?" (follow-up)

- Should use conversation context and provide specific examples from documents

Query 3: "How has Berkshire's acquisition strategy evolved over time?"

- Should analyze information across multiple years with temporal context

Performance Benchmarks

- **Document Processing**: Complete ingestion following Mastra's performance guidelines
- **Response Time**: Meet Mastra's streaming response benchmarks
- **Memory System**: Efficient conversation context retrieval
- **Agent Performance**: Consistent GPT-4o response quality
- **Vector Search**: Quick similarity search execution per Mastra standards

📊 Evaluation Criteria

Criteria	Weight	Requirements
Mastra Implementation	50%	Proper use of Mastra framework, following documentation patterns
RAG Functionality	25%	Vector search, context retrieval, source attribution working correctly
User Experience	15%	Intuitive interface, streaming responses, error handling
Code Quality	10%	Clean implementation following Mastra's best practices

🎯 Success Metrics

Technical Requirements

- **Mastra Compliance**: Implementation follows Mastra documentation patterns
- **Functionality**: All core RAG features working as designed
- **Performance**: Meets Mastra's performance benchmarks
- **Integration**: Proper use of Mastra's tools and workflows
- **Documentation**: Clear references to Mastra documentation sections used

User Experience Requirements

- **Streaming**: Smooth real-time responses using Mastra's capabilities
- **Conversation**: Natural dialogue flow with context preservation
- **Sources**: Clear attribution using Mastra's citation system
- **Interface**: Intuitive design following Mastra's UI patterns

⏰ Recommended Timeline

Total Duration: 6-8 hours

- **Phase 1**: Mastra Project Setup

- **Phase 2**: Document Processing with Mastra
- **Phase 3**: Mastra Agents & RAG Implementation
- **Phase 4**: Frontend Development with Mastra
- **Phase 5**: Integration & Deployment

Break Schedule:

- Take regular breaks throughout the session
- Extended break at the midpoint
- Short break before final testing

🚨 Common Challenges

Mastra Implementation Challenges

- **CLI Setup**: Ensure proper component selection during `create-mastra` initialization
- **Configuration**: Follow Mastra's environment setup precisely for all integrations
- **Agent Design**: Implement proper tool integration within Mastra's agent framework
- **Memory System**: Configure persistent memory correctly using Mastra's memory adapters

Development Challenges

- **Document Processing**: Use MDocument class appropriately for PDF handling
- **Vector Integration**: Configure vector database following Mastra's database guides
- **Streaming Setup**: Implement real-time responses using Mastra's streaming patterns
- **Frontend Integration**: Connect UI components with Mastra's backend systems

Learning Challenges

- **Framework Mastery**: Understand Mastra's five core primitives and their interactions
- **Documentation Navigation**: Efficiently use Mastra documentation for implementation guidance
- **Best Practices**: Follow Mastra's recommended patterns for production applications
- **Troubleshooting**: Use Mastra's development playground for debugging and testing

🎓 Learning Outcomes

Upon completion, you will have demonstrated mastery of:

Mastra Framework Expertise

- **Framework Architecture**: Understanding Mastra's core concepts and patterns
- **RAG Implementation**: Building RAG systems using Mastra's tools and workflows
- **Agent Development**: Creating intelligent agents following Mastra's agent patterns
- **Integration Skills**: Connecting various Mastra components and features
- **Best Practices**: Following Mastra's recommended development practices

Technical Implementation

- **Document Processing**: Using Mastra's document handling capabilities
- **Vector Operations**: Implementing search using Mastra's vector features
- **Streaming Systems**: Building real-time features with Mastra's streaming
- **Conversation Management**: Managing dialogue using Mastra's memory systems
- **Frontend Development**: Creating interfaces with Mastra's UI components

Software Development

- **Documentation Usage**: Effectively learning from technical documentation
- **Framework Adoption**: Quickly adapting to new development frameworks
- **System Integration**: Connecting multiple components into cohesive applications
- **Problem Solving**: Debugging and troubleshooting using framework resources
- **Quality Implementation**: Building production-ready applications with proper patterns

---

Your Primary Resource: Mastra Documentation

Ready to master the Mastra framework while building a sophisticated RAG application? Everything you need to know is in the official Mastra documentation! 🚀📚

This assignment will test your ability to learn from documentation, follow framework patterns, and implement complex AI functionality using a modern development framework. Focus on understanding Mastra's concepts deeply and implementing them correctly.