Legal Case Management AI System - Technical Challenge

Company Overview

We are a software company that builds case management systems for law firms. Our platform helps attorneys manage cases, documents, and client communications efficiently. We're looking for an AI Architect to build intelligent document processing and generation capabilities.

Position: AI Engineer

Challenge Overview

Build an AI system that can intelligently combine case documents (PDFs), database records, and generate legal documents such as demand letters. This challenge tests your understanding of RAG (Retrieval-Augmented Generation) architecture, MCP (Model Context Protocol) server implementation, and legal document generation workflows.

Use Case: Automated Legal Document Generation

Scenario

A law firm handles personal injury cases using our case management system. For each case, they maintain:

- **PDF Documents**: Medical records, police reports, insurance correspondence, wage statements
- **Database Records**: Case details, client information, opposing parties, timeline events, financial data
- **Goal**: Generate professional legal documents by intelligently combining structured database data with insights extracted from PDF documents

Example Workflow

1. Attorney requests: "*Generate a demand letter for Case #2024-PI-001 focusing on medical expenses and lost wages*"
2. Your system should:
   - Query the database for case details, plaintiff info, defendant info
   - Retrieve relevant medical records and wage documentation from PDFs using RAG
   - Extract key facts and amounts using semantic search
   - Generate a professional demand letter combining all information sources

Technical Requirements

1. RAG System for Legal Documents (Required)

Document Processing Pipeline

- Handle legal PDF documents (medical records, police reports, correspondence)
- Implement domain-specific chunking strategies for legal documents
- Create specialized embeddings that understand legal terminology and document structure
- Design retrieval that can find relevant information across document types

Key Features Required

- **Semantic Search**: Locate specific medical expenses, injury descriptions, timeline events
- **Document Classification**: Automatically categorize documents by type and relevance
- **Citation Tracking**: Maintain source document references with page numbers for legal compliance
- **Temporal Understanding**: Handle dates, deadlines, and chronological sequence of events
- **Amount Extraction**: Accurately identify and extract monetary amounts, percentages, and financial data

2. MCP Server for Case Management Database (Required)

Database Schema

You'll work with the following PostgreSQL schema:

sql

```sql
-- Cases table
CREATE TABLE cases (
    case_id VARCHAR(50) PRIMARY KEY,
    case_type VARCHAR(100),
    date_filed DATE,
    status VARCHAR(50),
    attorney_id INT,
    case_summary TEXT
);

-- Parties table
CREATE TABLE parties (
    party_id SERIAL PRIMARY KEY,
    case_id VARCHAR(50),
    party_type VARCHAR(50), -- 'plaintiff', 'defendant', 'witness', 'insurance_company'
    name VARCHAR(200),
    contact_info JSONB,
    insurance_info JSONB
);

-- Documents table
CREATE TABLE documents (
    doc_id SERIAL PRIMARY KEY,
    case_id VARCHAR(50),
    file_path VARCHAR(500),
    doc_category VARCHAR(100), -- 'medical', 'financial', 'correspondence', 'police_report'
    upload_date DATE,
    document_title VARCHAR(300),
    metadata JSONB
);

-- Case_events table
CREATE TABLE case_events (
    event_id SERIAL PRIMARY KEY,
    case_id VARCHAR(50),
    event_date DATE,
    event_type VARCHAR(100), -- 'accident', 'medical_treatment', 'expense', 'correspondence'
    description TEXT,
    amount DECIMAL(10,2) -- for expenses, damages, bills
);
```

Required MCP Tools

Implement the following MCP server functions:

1. `get_case_details(case_id: str)`: Retrieve complete case information including all parties
2. `get_case_documents(case_id: str, category: str = None)`: Get document file paths, optionally filtered by category
3. `get_case_timeline(case_id: str, event_type: str = None)`: Retrieve chronological events for the case
4. `get_financial_summary(case_id: str)`: Calculate total medical expenses, lost wages, and other damages
5. `search_similar_cases(case_type: str, keywords: list)`: Find precedent cases for reference
6. `get_party_details(case_id: str, party_type: str)`: Get specific party information (plaintiff, defendant, etc.)

3. Document Generation System (Required)

Build a system that can generate professional legal documents (specifically demand letters) by combining information from both the RAG system and the MCP database queries. The system should be able to create properly formatted legal documents with accurate citations and comprehensive information synthesis.

Test Case: Case #2024-PI-001

Case Information

- **Case ID**: 2024-PI-001
- **Type**: Personal Injury - Motor Vehicle Accident
- **Date Filed**: March 15, 2024
- **Status**: Active - Demand Phase
- **Plaintiff**: John Smith
- **Defendant**: ABC Insurance Company (representing driver Sarah Johnson)

Available Documents

The following PDF documents will be provided in the `/sample_docs/2024-PI-001/` directory:

1. `medical_records_dr_jones.pdf` - Primary care physician records showing injury diagnosis and treatment
2. `police_report_incident_789.pdf` - Official police report with accident details and fault determination
3. `wage_statements_2024.pdf` - Employment records showing income loss due to injury
4. `insurance_correspondence.pdf` - Communication with defendant's insurance company

Test Query

Your system must successfully handle this request:

"Generate a demand letter for Case 2024-PI-001. Include all medical expenses, lost wages, and pain and suffering damages. Reference specific medical findings from Dr. Jones and cite the police report for liability determination. The demand should be professional and include proper legal citations."

Expected System Flow

1. **Database Query**: Retrieve case details, party information, and recorded events via MCP
2. **Document Retrieval**: Use RAG to extract relevant information from all four PDF documents
3. **Information Synthesis**: Combine database records with document insights
4. **Document Generation**: Create a professional demand letter with proper legal structure
5. **Citation Management**: Include proper document references with page numbers

Submission Requirements

Please provide a GitHub repository containing your complete implementation with clear build and setup instructions so we can run the system locally. The repository should include:

- **Complete Working Code**: All components implemented and functional
- **Build Instructions**: Clear README with setup, installation, and usage instructions
- **Sample Output**: A generated demand letter for the provided test case
- **Documentation**: Architecture overview and key implementation decisions

Technical Constraints and Requirements

Technology Stack

- **Python**: Primary development language, should work on Windows or MAC
- **Vector Database**: Choose appropriate solution (Pinecone, Weaviate, Chroma, etc.)
- **MCP Implementation**: Follow Model Context Protocol specifications
- **LLM Integration**: OpenAI GPT-4 or equivalent for document generation
- **Database**: PostgreSQL/MySql/MSSQL for case management data
- **PDF Processing**: PyPDF2, pdfplumber, or similar library