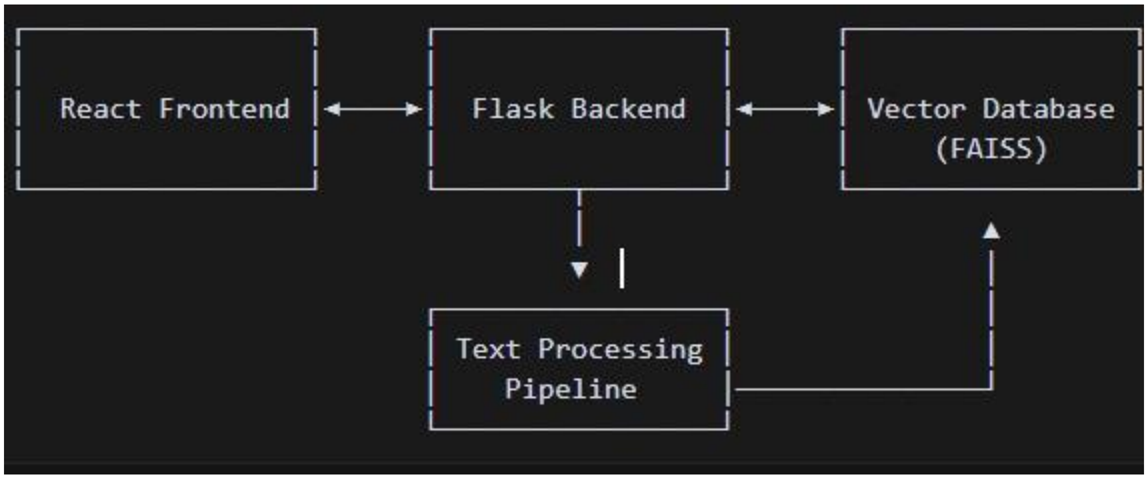
Research & Development Document

1. System Architecture

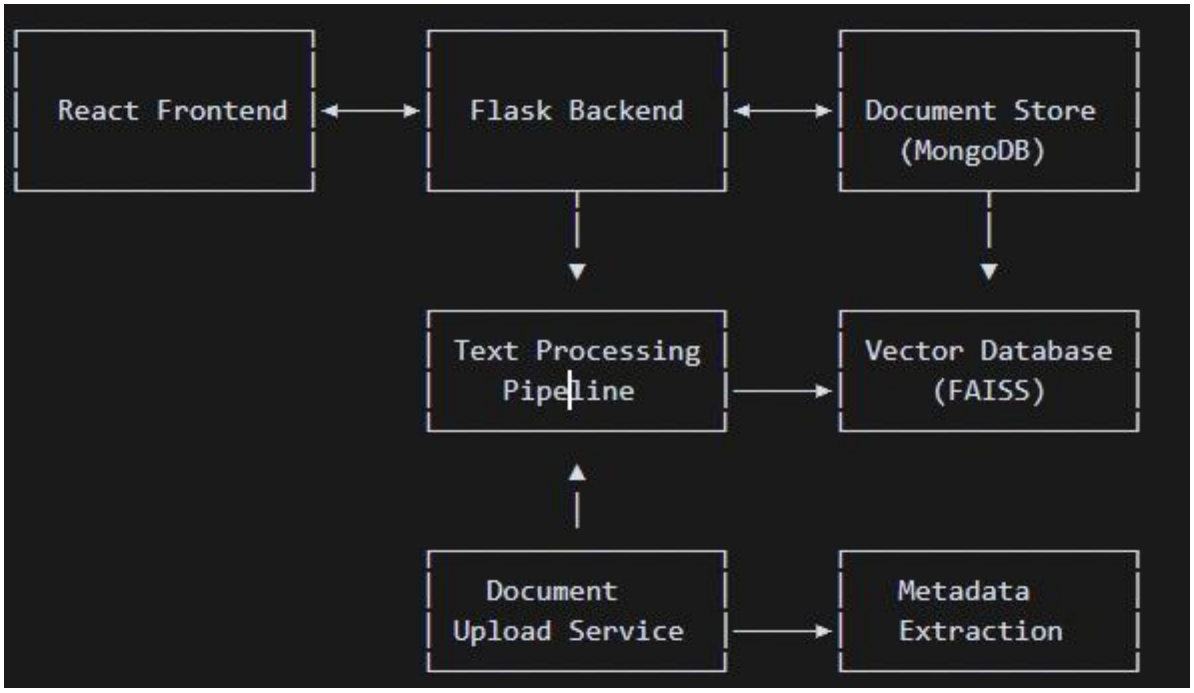
Current Architecture

Our system employs a modern, well-structured architecture with the following components:

- **Frontend:** React.js with Tailwind CSS
- **Backend:** Flask RESTful API
- **Embedding Model:** Sentence-Transformers (all-MiniLM-L6-v2)
- **Vector Database:** FAISS for similarity search
- **Text Processing:** Custom extraction and processing pipeline



Proposed Enhanced Architecture



2. Data Flow

Document Ingestion Process

Document Upload

- Documents are uploaded through the UI
- Initial validation checks format and readability
- Files stored temporarily for processing

Text Extraction

- PDF/Image documents processed with OCR
- Text cleaned and normalized
- Initial metadata extracted from document structure

Document Analysis

- Named Entity Recognition (NER) for identifying key entities

- Structure analysis for court, dates, parties, etc.
- Citation linking and reference extraction

Vector Embedding Generation

- Text chunked into semantic units
- Embeddings generated using Sentence-Transformers
- Vectors stored in FAISS index with document IDs

Document Storage

- Processed text and metadata stored in MongoDB
- Documents indexed for full-text search
- Reference links established between related documents

Search and Retrieval Process

Query Processing

- User query analyzed for intent and entities
- Query expansion for semantic variations
- Filter parameters extracted

Vector Search

- Query converted to embedding vector
- Top-k similar documents retrieved from FAISS
- Results ranked by relevance score

Metadata Filtering

- Results filtered based on user-selected criteria
- Date ranges, court types, and other metadata applied

Result Presentation

- Most relevant sections highlighted
- Context provided around matching segments

- Related documents suggested

Chat Interaction Flow

Query Understanding

- User question analyzed for legal context
- Intent classification (e.g., fact-finding, precedent search)
- Entity extraction for specific references

Document Retrieval

- Relevant documents retrieved based on query embedding
- Context window assembled from top document segments

Response Generation

- Retrieved context used to formulate response
- Citations included from source documents
- Response formatted for readability

3. Technology Stack

Current Technologies

- **Frontend:** React.js, Tailwind CSS
- **Backend:** Flask (Python)
- **Embedding Model:** Sentence-Transformers
- **Vector Search:** FAISS
- **OCR:** Not explicitly mentioned, likely using standard libraries

Recommended Enhancements

- **Document Database:** MongoDB for flexible schema storage
- **Advanced NLP:** Spacy or Stanza for legal entity recognition
- **Improved Embeddings:** Switch to higher-performance models like MPNet or legal-domain specific embeddings

- **Caching Layer:** Redis for query caching and session management
- **Search Enhancement:** Elasticsearch for text search alongside vector search
- **API Management:** API gateway for rate limiting and authentication
- **Containerization:** Docker and Kubernetes for scalable deployment

4. Scalability Considerations

Current Limitations

- In-memory FAISS index has size constraints
- Single-server Flask deployment limits throughput
- Document processing may become a bottleneck with large volumes

Scaling Strategies

Horizontal Scaling

- **API Layer:** Multiple Flask instances behind a load balancer
- **Processing Pipeline:** Worker pools for document processing
- **Vector Search:** Distributed FAISS or migration to specialized services

Data Management

- **Sharding:** Divide document collection by logical boundaries (court, year)
- **Caching Tiers:** Multi-level caching for frequently accessed documents
- **Async Processing:** Queue-based architecture for document ingestion

Infrastructure

- **Cloud Deployment:** Leverage cloud-native services for automatic scaling
- **Serverless Components:** Use serverless functions for bursty workloads
- **CDN Integration:** Distribute static content and common query results

5. Performance Optimization

Search Performance

- **Index Optimization:** Fine-tune FAISS parameters for recall vs. speed
- **Query Caching:** Cache common queries and their results
- **Precomputed Views:** Generate materialized views for common filters

Document Processing

- **Parallel Processing:** Multi-threaded document analysis
- **Incremental Updates:** Delta updates for document modifications
- **Batch Processing:** Group similar documents for efficient processing

Frontend Experience

- **Progressive Loading:** Load essential UI first, then enhancements
- **Request Batching:** Combine related API calls
- **Client-side Caching:** Store recent results in browser storage

6. Security Considerations

- **Document Access Control:** Role-based permissions system
- **Data Encryption:** Encrypt sensitive document data at rest
- **API Authentication:** JWT-based authentication for API access
- **Audit Logging:** Track document access and modifications
- **Input Validation:** Sanitize all user inputs to prevent injection attacks

7. Monitoring and Analytics

- **Performance Metrics:** Track response times and resource usage
- **User Behavior:** Analyze search patterns and document interactions
- **Error Tracking:** Monitor and alert on system errors
- **A/B Testing:** Test UI and algorithm improvements

8. Future Enhancements

Short-term Improvements

- **Document Upload UI:** Add drag-and-drop document upload interface
- **Advanced Filters:** Expand metadata filtering options
- **Saved Searches:** Allow users to save and share searches
- **Feedback System:** Collect user feedback on search results
- **Mobile Optimization:** Further improve mobile experience

Long-term Vision

- **Legal AI Assistant:** More sophisticated legal reasoning capabilities
- **Cross-document Analysis:** Automatically identify related cases and precedents
- **Precedent Mapping:** Visual network of case citations and influences
- **Predictive Analytics:** Suggest relevant documents based on user behavior
- **Multi-language Support:** Expand to handle documents in multiple languages