Project 6: Intelligent Chat System with File Analysis

Objective (Why?)

Transform the authentication dashboard from Project 5 into a fully functional intelligent chat system. This accelerated timeline leverages your established foundation and growing expertise. You will practice:

- Part 1: Chat thread management, real-time conversations, and message persistence
- Part 2: File upload integration, document processing, and context-aware AI responses
- Full Integration: Combining authentication, chat functionality, document intelligence

Core Requirements

Part 1: Basic Chat System

Component	Requirement
Chat Management	Create new chat threads, manage multiple conversations, and persistent chat history
LLM Integration	Real-time conversations with OpenAI/Gemini, message streaming, and conversation context
Database Design	PostgreSQL tables for chat threads, messages, and user associations
UI Enhancement	Activate the disabled components from Project 5 dashboard

Part 2: File-Based Intelligence

Component	Requirement
File Processing	Upload and extract text from PDF, CSV, TXT files with proper validation
Document Analysis	Convert documents to searchable embeddings and implement vector storage
Context-Aware Chat	Query documents conversationally and provide source-referenced responses
Multi-File Support	Handle multiple documents per chat thread with intelligent context switching
Orchestration	Use LangChain to build the end-to-end RAG pipeline, from document loading to response generation

Part 3: Role-Based Intelligence

Component	Requirement

Authorization	Integrate RBAC from Project 5, restricting document upload/management to admin roles
UI Adaptation	The file upload component should be visible only to users with the admin role
Secure Endpoints	All file processing and management endpoints must be protected by the admin role
Ownership	Chat history and threads remain tied to individual users, regardless of role

Development Approach: Milestone-Based Progression

Philosophy: Focus on deliverable quality and comprehensive review compliance rather than rigid timelines. Each milestone must pass all relevant review templates from our Templates folder before proceeding.

Milestone 1: Advanced Chat Foundation & File Processing

Deliverables:

- Extended chat system with thread management and persistence
- Multi-format file upload and text extraction pipeline
- Database schema for chats, messages, and documents
- Vector embedding generation and storage system
- Authentication integration with chat ownership

Review Requirements:

- Architecture Review: Scalable chat and document processing architecture
- Security Review: File upload security, access control, data protection
- Performance Review: Efficient document processing and storage

Milestone 2: RAG Implementation Development

Deliverables:

- RAG system with document context retrieval
- Context-aware AI responses with source attribution
- Basic LangChain implementation with function calling
- Tool integration for document analysis and external APIs
- Semantic search across uploaded documents
- Multi-document conversation support

Review Requirements:

- AI Integration Review: Sophisticated RAG implementation
- Architecture Review: Clean separation of concerns and scalable design
- Performance Review: Fast document retrieval and response generation

Milestone 3: Production Features & Optimization

Deliverables:

- Advanced chat management (thread organization, search, export)
- Document management interface with analysis tools
- Performance optimization and caching strategies
- Comprehensive testing and documentation
- Production deployment preparation

Review Requirements:

- AI Integration Review: Production-ready RAG system with advanced features
- Architecture Review: Complete scalable system architecture
- Security Review: Enterprise-level security and data protection
- Performance Review: Optimized performance under load

Milestone Progression Rules:

- Cannot advance to next milestone without passing all review requirements
- Flexible timing allows for learning at individual pace
- Quality gates ensure each milestone meets professional standards
- Mentor support available for concept clarification and review failures

Measurable Goals & Review Template Compliance

Primary Objectives

- AI Integration Mastery: Pass AI Integration Review with 9.0/10+ score (advanced RAG)
- Architecture Excellence: Pass Architecture Review with 8.5/10+ score
- Security Standards: Pass Security Review with 8.5/10+ score
- Performance Optimization: Sub-2s document analysis and response generation
- Code Quality Standards: Pass Code Quality Review with 8.5/10+ score

Performance Standards

- Document Processing: < 5 seconds for typical documents (PDF, DOCX, TXT)
- Response Generation: < 2 seconds for context-aware responses
- Vector Search: < 500ms for semantic similarity queries
- Chat Performance: Real-time messaging with < 100ms latency

Task Tracking & Project Management Integration

Epic: Project 6 - Intelligent Chat System with File Analysis

Epic ID: P6-RAG-CHAT
Priority: High

Milestone 1: Advanced Chat Foundation & File Processing

Feature 6.1: Enhanced Chat Infrastructure

Task ID: P6-M1-CHAT
Priority: Critical

Sub-tasks:

- P6-M1-CHAT-01: Chat database schema extension
    - Description: Tables for threads, messages, documents with relationships
    - Acceptance Criteria: Normalized schema with proper indexing
- P6-M1-CHAT-02: Thread management system
    - Description: Create, manage, and organize chat conversations
    - Acceptance Criteria: Full CRUD operations for chat threads
- P6-M1-CHAT-03: Message persistence and retrieval
    - Description: Store and retrieve conversation history efficiently
    - Acceptance Criteria: Paginated message history with search

Feature 6.2: Document Processing Pipeline

Task ID: P6-M1-DOC
Priority: Critical

Sub-tasks:

- P6-M1-DOC-01: File upload and validation
    - Description: Secure file upload with type and size validation
    - Acceptance Criteria: Support PDF, DOCX, TXT with security checks
- P6-M1-DOC-02: Text extraction engine
    - Description: Multi-format text extraction and preprocessing
    - Acceptance Criteria: Clean text extraction from all supported formats
- P6-M1-DOC-03: Vector embedding system
    - Description: Generate and store document embeddings
    - Acceptance Criteria: Efficient embedding generation and storage

Milestone 2: RAG Implementation & Intelligent Responses

Feature 6.3: RAG System Implementation

Task ID: P6-M2-RAG
Priority: High

Sub-tasks:

- P6-M2-RAG-01: Semantic search implementation
    - Description: Vector similarity search for document retrieval
    - Acceptance Criteria: Fast, relevant document chunk retrieval
- P6-M2-RAG-02: Context-aware response generation
    - Description: LLM integration with document context
    - Acceptance Criteria: Accurate responses with source attribution
- P6-M2-RAG-03: Multi-document conversation support
    - Description: Handle context from multiple uploaded documents
    - Acceptance Criteria: Coherent responses across document sources

Milestone 3: Production Features & Optimization

Feature 6.4: Advanced Features & Optimization

Task ID: P6-M3-PROD
Priority: Medium

Sub-tasks:

- P6-M3-PROD-01: Chat management interface
    - Description: Advanced chat organization and search capabilities
    - Acceptance Criteria: Full chat management with search and export
- P6-M3-PROD-02: Document analysis tools
    - Description: Document insights, summaries, and management
    - Acceptance Criteria: Document analytics and management interface
- P6-M3-PROD-03: Performance optimization
    - Description: Caching, indexing, and query optimization
    - Acceptance Criteria: Meet all performance standards

Technical Specifications

Part 1: Database Schema

- Design chat_threads table with user associations, timestamps, and metadata
- Create messages table with thread relationships, content, and role indicators

- Implement proper indexing for efficient message retrieval and pagination

Part 2: File Processing Pipeline

- Build document upload system with file type validation and size limits
- Implement text extraction using appropriate libraries for each file format
- Create vector embedding storage using PostgreSQL with pgvector extension

Authentication Integration

- Extend existing user authentication to include chat thread ownership
- Implement proper authorization for chat and document access control
- Maintain session management and secure API endpoints

Project Structure

- Extend the authentication system from Project 5 with chat and document modules
- Add new database models for chats, messages, and documents
- Create service layers for LLM integration, file processing, and vector operations
- Enhance React frontend with chat components and file upload interface

Part 1: Chat System Features

Chat Thread Management

- Create new conversations with automatic title generation
- List and organize chat threads in the left sidebar
- Implement thread deletion and archiving functionality

Real-time Messaging

- Send messages and receive LLM responses with proper loading states

- Support message editing and conversation regeneration
- Implement typing indicators and message status tracking

Conversation Persistence

- Store all messages with proper user and thread associations
- Implement efficient pagination for long conversations
- Support conversation export and sharing capabilities

Part 2: Document Intelligence Features

File Upload and Processing

- Drag-and-drop interface for multiple file types (PDF, CSV, TXT)
- Progress tracking and error handling for large files
- File validation, virus scanning, and size limit enforcement

Document Analysis

- Extract and chunk text content for optimal processing
- Generate embeddings and store in vector database
- Create document metadata and indexing for quick retrieval

Context-Aware Responses

- Implement RAG (Retrieval-Augmented Generation) for document queries
- Provide source attribution and page references in responses
- Support follow-up questions and document comparisons

Advanced Features (Stretch Goals)

Part 1 Enhancements

- Real-time Collaboration: Share chat threads with other users
- Voice Integration: Speech-to-text input and text-to-speech output
- Chat Templates: Pre-built conversation starters and prompts

- Advanced Search: Full-text search across all user conversations

Part 2 Enhancements

- Visual Document Processing: Handle images and charts within documents
- Multi-language Support: Process documents in various languages
- Document Comparison: Side-by-side analysis of multiple files
- Data Visualization: Generate charts and graphs from CSV data

Deliverables

Part 1 Deliverables

1. Working Chat System with multiple thread support
2. LLM Integration with real-time responses
3. CHAT_DEMO.md with conversation screenshots and feature walkthrough

Part 2 Deliverables

4. File Upload System supporting PDF, CSV, TXT formats
5. Document Intelligence with question-answering capabilities
6. DOCUMENT_ANALYSIS.md with sample file uploads and query examples

Combined Deliverables

7. GitHub Repository with complete source code
8. Live Demo showing both chat and file analysis features
9. INTEGRATION_GUIDE.md documenting the complete system architecture
10. Technical_Learnings.md documenting the technical learnings

Evaluation Rubric (100 Points)

Criterion	Points	Details
Part 1: Chat System	40 pts	• Multiple chat threads working • LLM integration and persistence • Real-time messaging interface
Part 2: File Intelligence	35 pts	• Document upload and processing • Vector search and RAG implementation • Context-aware responses

Integration & UX	15 pts	• Seamless user experience • Proper authentication integration • Professional UI/UX design
Code Quality	10 pts	• Clean architecture and documentation • Error handling and performance • Security best practices

Testing Scenarios

Part 1: Chat System Testing

- Create new chat thread and send messages
- Switch between multiple active conversations
- Verify message persistence across sessions
- Test LLM integration with various prompts
- Validate chat history and pagination

Part 2: File Analysis Testing

- Upload different file types (PDF, CSV, TXT)
- Ask questions about uploaded document content
- Verify source attribution in responses
- Test multi-file context switching
- Validate file size limits and error handling

Integration Testing

- Ensure proper user authentication throughout
- Test responsive design on mobile devices
- Verify data isolation between users
- Check performance with large files and long chats
- Validate security and access controls

Quick Start Resources

Chat System Development

- WebSocket Integration: FastAPI WebSockets Guide
- React Chat Components: Building Chat Interfaces
- Message Streaming: Server-Sent Events

File Processing

- Document Parsing: PyPDF2 Documentation
- Vector Databases: pgvector Setup Guide or similar
- RAG Implementation: LangChain RAG Tutorial

Performance Requirements

Part 1: Chat Performance

- Message sending: < 2 second response time
- Chat thread loading: < 3 seconds
- Real-time updates: < 2 seconds latency

Part 2: File Processing Performance

- File upload: Support files up to 5MB
- Document processing: < 30 seconds for typical documents
- Query response: < 3 seconds for document-based questions

Security Considerations

Chat Security

- Implement proper user authorization for chat access
- Sanitize all user inputs and LLM outputs
- Secure message storage with encryption at rest

File Security

- Validate file types and scan for malicious content
- Implement proper file storage with access controls
- Ensure document data isolation between users