

# Document Management System with ML Classification

## Complete Project Plan and Documentation

### Table of Contents

- 1. Project Overview
- 2. Phase 1 - Database Foundation
- 3. Technology Stack
- 4. Phase 2 - Implementation Plan
- 5. Technical Specifications
- 6. Development Guidelines
- 7. Risk Management
- 8. Success Criteria

## 1. Project Overview

### Project Description

AI-powered document management system with automated topic classification capabilities.

### Project Objectives

- Create a scalable document management system
- Implement automated topic classification
- Provide efficient document search and retrieval
- Enable secure document storage and version control

## 2. Phase 1 - Database Foundation

### Completed Components

- 1. Database Schema Implementation
  - DocumentManagement database created
  - Core tables implemented
  - Constraints and relationships established
  - Performance indexes created
  - Initial reference data inserted

### Database Structure

Tables:

- Documents
- DocumentVersions
- DocumentMetadata
- DocumentRelationships
- Users

- └─ Topics
- └─ Tags
- └─ DocumentTypes

## Schema Features

- Version control support
- Flexible metadata storage
- Document relationships
- Security and audit capabilities
- Topic classification structure

# 3. Technology Stack

## Backend Technologies

### 1. Primary Framework

- FastAPI
- Async support
- Built-in API documentation
- Strong type validation

### 2. Database

- SQL Server
- SQLAlchemy ORM
- Alembic migrations

### 3. Document Processing

- PyPDF2
- python-docx
- Tesseract OCR

## Frontend Technologies

### 1. React + TypeScript

- Component-based architecture
- Strong typing
- Modern development practices

### 2. Key Libraries

- react-dropzone
- react-query
- tailwindcss
- react-pdf

## Supporting Technologies

### 1. Authentication

- Python-JOSE
- Passlib

### 2. File Handling

- python-multipart
- aiofiles

### 3. Storage

- MinIO
- Redis caching

## 4. Phase 2 - Implementation Plan

### Week 1-2: Project Setup & File Storage

#### 1. Development Environment

- Python setup
- FastAPI configuration
- Database integration

#### 2. File Storage Architecture

- Storage structure implementation
- Security configuration
- Backup procedures

### Week 3-4: Core Backend Development

#### 1. CRUD Operations

- Document management
- Version control
- Metadata handling

#### 2. Service Layer

- Document services
- User services
- Search services

### Week 5-6: Frontend Development

#### 1. React Application

- Project structure
- Core components

- API integration

2. User Interface

- Document management
- Search interface
- User management

Week 7-8: Features & Refinement

1. Document Processing

- Upload pipeline
- Preview generation
- Version control

2. Search Implementation

- Full-text search
- Metadata search
- Advanced filtering

5. Technical Specifications

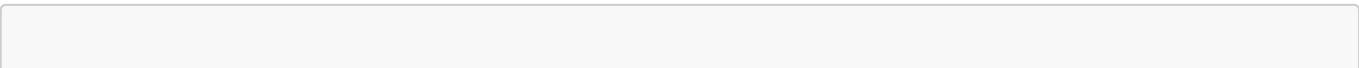
File Storage Structure

```
document_storage/  
├─ uploads/  
├─ documents/  
│   ├─ {year}/  
│   │   └─ {month}/  
│   │       └─ {docId}/  
├─ backups/  
└─ cache/
```

API Structure

```
app/  
├─ api/  
│   └─ v1/  
├─ core/  
├─ db/  
├─ services/  
└─ schemas/
```

Key Service Interfaces



```
class DocumentService:
    async def create_document(...)
    async def retrieve_document(...)
    async def update_document(...)
    async def delete_document(...)

class FileManager:
    async def store_file(...)
    async def retrieve_file(...)
    async def generate_preview(...)
```

## 6. Development Guidelines

### Code Standards

- PEP 8 compliance
- Type hints
- Comprehensive documentation
- Test coverage requirements

### Security Practices

- Authentication required
- Role-based access control
- Secure file storage
- Input validation

### Performance Requirements

- Response time < 2 seconds
- Efficient file handling
- Caching strategy
- Scalable architecture

## 7. Risk Management

### Technical Risks

#### 1. Data Loss Prevention

- Regular backups
- Version control
- File integrity checks

#### 2. Performance Issues

- Monitoring
- Optimization strategy
- Scalability planning

## Project Risks

### 1. Timeline Management

- Weekly reviews
- Milestone tracking
- Resource allocation

### 2. Quality Assurance

- Testing strategy
- Code review process
- Security audits

## 8. Success Criteria

### Functional Requirements

- Document upload/download
- Search functionality
- User authentication
- Document processing
- Topic classification

### Non-functional Requirements

- Performance metrics
- Security compliance
- Scalability
- Maintainability

## Appendix

### Database Scripts Location

```
DocumentManagement/  
├── Scripts/  
│   ├── 01_CreateDatabase.sql  
│   ├── 02_CreateTables.sql  
│   ├── 03_CreateConstraints.sql  
│   ├── 04_CreateIndexes.sql  
│   └── 05_InitialData.sql
```

### Development Environment Setup

#### 1. Required Software

- Python 3.9+
- Node.js 16+

- VS Code
- SQL Server

## 2. Extensions

- Python
- SQL Server
- React
- Git

## 3. Configuration

- Virtual environment
- Database connection
- API settings
- Storage location