.NET Document Management System with ML Classification

Project Overview and Architecture Document

Version 1.0 - February 19, 2025

Table of Contents

1. Introduction

Project Purpose

Development of an on-premises document management system with intelligent document classification using ML.NET. The system will provide secure document storage, versioning, and automated classification capabilities within an organization's internal infrastructure.

Key Objectives

- Secure document management and storage
- Automated document classification using ML.NET
- Integration with internal Active Directory
- High performance document processing
- Scalable on-premises deployment

2. Architecture Overview

System Architecture

The system follows Clean Architecture principles with clear separation of concerns:

```
DocumentManagement.NET/
├── DocumentManagement.Domain/         # Core business logic and entities
├── DocumentManagement.Application/    # Application services and interfaces
├── DocumentManagement.Infrastructure/ # External concerns and implementations
├── DocumentManagement.API/           # Web API layer
├── DocumentManagement.ML/            # ML.NET integration
└── DocumentManagement.Web/           # Frontend application
```

Design Principles

- Dependency Inversion Principle
- Clear separation of concerns
- Domain-driven design
- SOLID principles
- Repository pattern for data access
- CQRS for complex operations

3. System Components

Core Components

1. Document Management Core
    - Document storage and retrieval
    - Version control
    - Metadata management
    - Search functionality
2. ML Classification Engine
    - ML.NET model training
    - Document classification
    - Model management
    - Feature extraction
3. Security Services
    - Authentication
    - Authorization
    - Audit logging
    - File encryption
4. Storage Management
    - File system operations
    - MinIO integration
    - Caching layer
    - Backup management

Supporting Components

1. Background Services
    - Document processing
    - ML model training
    - System maintenance
    - Backup operations

2. Monitoring Services
    - System health
    - Performance metrics
    - Security monitoring
    - ML model performance

4. Technology Stack

Core Technologies

- .NET 8.0
- ASP.NET Core Web API
- Entity Framework Core 8.0
- ML.NET 3.0
- SQL Server 2022

Frontend Options

1. Blazor WebAssembly
    - Blazor.WebAssembly
    - MudBlazor
    - Blazor.WebAssembly.Authentication
2. React with TypeScript (Alternative)
    - React 18+
    - TypeScript 5.0+
    - SignalR Core

Infrastructure

- MinIO Object Storage
- Elasticsearch (Search)
- Redis (Caching)
- NLog (Logging)

5. ML Integration Architecture

ML.NET Implementation

```
DocumentManagement.ML/
├── Models/                 # ML model definitions
├── Training/              # Training pipelines
├── Prediction/           # Prediction engines
├── DataPreprocessing/    # Data preparation
└── Evaluation/          # Model evaluation
```

Classification Pipeline

1. Document Intake
    - File validation
    - Text extraction
    - Feature preprocessing
2. Classification
    - Model selection
    - Feature extraction
    - Prediction
    - Confidence scoring
3. Model Management
    - Version control
    - Performance monitoring
    - Retraining triggers
    - Model deployment

6. Security Architecture

Authentication

- Windows Authentication
- JWT for API access
- Active Directory integration
- Role-based access control

Authorization

- Document-level permissions
- Feature-based access control
- Administrative boundaries
- Audit logging

Data Security

- At-rest encryption
- In-transit encryption
- Secure file storage
- Backup encryption

7. Deployment Architecture

On-Premises Infrastructure

```
Internal Network/
├── Application Servers/
│   ├── Web Servers
│   ├── API Servers
│   └── ML Processing Servers
├── Database Servers/
│   ├── Primary SQL Server
│   └── Replica Servers
├── Storage/
│   ├── Document Storage
│   ├── MinIO Cluster
│   └── Backup Storage
└── Monitoring/
    ├── Prometheus
    ├── Grafana
    └── ELK Stack
```

High Availability

- Load balancing
- Failover clustering
- Database mirroring
- Storage redundancy

8. Performance Considerations

Optimization Strategies

1. Caching

    - Document metadata
    - Search results
    - ML model predictions
    - User permissions

2. Database Optimization

    - Query optimization
    - Indexing strategy
    - Partitioning
    - Memory optimization

3. Document Processing

    - Parallel processing
    - Batch operations
    - Asynchronous processing
    - Queue management

Scalability

1. Vertical Scaling

    - CPU utilization
    - Memory management
    - Storage capacity
    - Network bandwidth

2. Horizontal Scaling

    - Application servers
    - Database read replicas
    - Storage nodes
    - Processing nodes

Next Steps

1. Infrastructure Setup
2. Database Migration
3. Core Development
4. ML Model Development
5. Security Implementation
6. Testing and Validation