Optimized Task List - .NET Document Management System

1. Development Environment Setup
    - Install Visual Studio 2022 (Enterprise/Professional)
    - Install .NET 8.0 SDK
    - Set up SQL Server Management Studio
    - Install required ML.NET Model Builder extension
    - Configure Git repository
    - Set up local development environment variables

2. Infrastructure Setup
    - Install and configure Jenkins or GitLab CI
    - Set up local Docker registry
    - Configure self-hosted container orchestration
    - Implement Prometheus for metrics
    - Set up Grafana for monitoring dashboards
    - Configure ELK Stack for logging
    - Create local backup solutions
    - Set up disaster recovery procedures
    - Implement high availability configuration
    - Configure system monitoring
    - Set up performance optimization
    - Create database maintenance jobs

3. Database Foundation
    - Port SQL Server schema from existing scripts
    - Implement Entity Framework Core models
    - Create database migrations
    - Set up stored procedures for complex queries
    - Implement database indexing strategy
    - Create database maintenance procedures
    - Set up backup and recovery processes

4. Core Application Architecture
    - Create .NET Core Web API project
    - Implement Clean Architecture pattern with following layers:
        - Domain Layer (Entities, Interfaces)
        - Application Layer (Services, DTOs)
        - Infrastructure Layer (Repositories, External Services)
        - API Layer (Controllers, Middleware)
    - Set up Dependency Injection container
    - Implement self-hosted identity solution

- Configure Windows Authentication integration
- Implement local configuration management

5. Document Processing Services

- Implement document parsing using iText7
- Create OCR service using Tesseract.NET
- Implement document metadata extraction
- Set up local file system storage with hierarchical structure
- Configure MinIO as self-hosted object storage
- Implement NAS integration (if available)
- Create document versioning system
- Implement local caching strategy
- Set up file system maintenance procedures

6. ML.NET Integration and Model Development

- Set up ML.NET infrastructure
- Create document classification model using ML.NET Model Builder
- Implement text feature extraction pipeline
- Develop training data preparation utilities
- Create model training and evaluation workflows
- Implement local model storage and versioning
- Create local model registry system
- Implement file-based model persistence
- Create training data pipeline
- Implement document feature extraction
- Create model evaluation framework
- Implement local model deployment pipeline
- Set up model retraining workflow
- Create local training data management system
- Implement model performance monitoring

7. API Development

- Implement RESTful controllers for:
  - Document management
  - User management
  - Classification services
  - Search services
- Implement proper validation using FluentValidation
- Set up API documentation using Swagger/OpenAPI
- Create rate limiting and request throttling

8. Security Implementation

- Set up JWT authentication
- Implement role-based authorization
- Configure Active Directory integration

- Implement local certificate management
- Configure internal network security
- Set up local audit logging
- Implement file system security
- Create security monitoring system
- Set up intrusion detection
- Configure backup and recovery procedures

9. Frontend Development (Choose one)

- Blazor WebAssembly Option:
  - Set up Blazor WASM project
  - Implement component library
  - Create document upload/preview components
  - Implement real-time classification feedback
  - Configure local asset management
- React with TypeScript Option:
  - Create API client using OpenAPI generator
  - Port existing React components
  - Implement SignalR for real-time updates
  - Set up local asset delivery
  - Configure internal network optimizations

10. Testing Infrastructure

- Set up unit testing with xUnit
- Implement integration tests
- Create API testing suite
- Set up automated UI testing
- Implement ML model testing framework
- Create performance testing suite
- Set up load testing environment
- Implement security testing procedures

11. System Administration

- Perform hardware capacity planning
- Configure network infrastructure
- Set up storage management
- Implement system monitoring
- Create maintenance schedules
- Configure backup systems
- Set up disaster recovery
- Implement performance monitoring
- Create system health checks
- Set up alerting systems

12. Documentation

    - Create API documentation
    - Write technical documentation
    - Create user guides
    - Document ML model architecture
    - Create deployment guides
    - Develop system architecture diagrams
    - Create network topology documentation
    - Write hardware requirements
    - Create installation guides
    - Document backup procedures
    - Write security policies
    - Create system administration guides
    - Document maintenance procedures

Key Changes Made:

1. Moved Infrastructure Setup to position #2
2. Positioned Document Processing Services before ML.NET Integration
3. Combined ML.NET Integration and Model Development into a single phase
4. Moved Security Implementation before Frontend Development
5. Positioned Frontend Development after security is in place
6. Kept System Administration and Documentation as final phases
7. Removed redundancies between phases

Notes:

- Each phase builds upon the previous phases
- Infrastructure and security are established early
- Core document processing is in place before ML features
- Frontend development occurs after backend stability
- Testing and documentation are continuous but formalized in final phases