Distribution Management System with WSO2 IS 7.1 and .NET 9

Complete Implementation Guide

Table of Contents

Overview

This guide provides a complete implementation of a Distribution Management System integrated with WSO2 Identity Server 7.1 for authentication and authorization using OAuth2 Resource Owner Password Credentials (ROPC) flow.

Key Features:

- Custom login page (not using WSO2 hosted login)
- OAuth2 Password Grant Type (ROPC flow)
- JWT Bearer token authentication
- Role-based access control (RBAC)
- Three user roles: yks_admin, yks_user, yks_test
- Clean Architecture pattern
- .NET 9 with latest best practices
- Entity Framework Core for data persistence
- Environment-based configuration

Access Control:

- **yks_admin**: Full access (view users, edit products, view products)
- **yks_user**: Edit and view products
- **yks_test**: View-only access to products

Prerequisites

Software Requirements

1. **.NET 9 SDK**

```
# Check installation
dotnet --version
```

2. **macOS Setup**

```
# Install .NET 9 SDK
brew install --cask dotnet-sdk

# Install EF Core tools
dotnet tool install --global dotnet-ef
```

3. **IDE Options**
   - Visual Studio for Mac
   - JetBrains Rider
   - Visual Studio Code with C# extension
4. **Database**
   - PostgreSQL, MySQL, or SQL Server
   - SQLite (for development)

WSO2 Identity Server Configuration

Your WSO2 IS 7.1 is already configured with:

- Application Name: yks_api
- Client ID: 7eWli_Xh2SdqbNQHfex0nZfC1mUa
- Client Secret: oyvSQgpwsvJWrbncCsqTR9lbVjPr4Sq8abdUhfN11R4a
- Token Endpoint: https://iam.bimats.com/oauth2/token
- UserInfo Endpoint: https://iam.bimats.com/oauth2/userinfo
- Roles: yks_admin, yks_test, yks_user

Project Setup

Step 1: Create Solution and Projects

```
# Create solution directory
mkdir DistributionManagementSystem
cd DistributionManagementSystem

# Create solution file
dotnet new sln -n DistributionManagementSystem

# Create src directory
mkdir src
cd src

# Create Domain Layer (Class Library)
dotnet new classlib -n DistributionManagement.Domain
dotnet sln ../DistributionManagementSystem.sln add DistributionManagement.Domain

# Create Application Layer (Class Library)
dotnet new classlib -n DistributionManagement.Application
dotnet sln ../DistributionManagementSystem.sln add DistributionManagement.Application

# Create Infrastructure Layer (Class Library)
dotnet new classlib -n DistributionManagement.Infrastructure
dotnet sln ../DistributionManagementSystem.sln add DistributionManagement.Infrastructure

# Create API Layer (Web API)
dotnet new webapi -n DistributionManagement.API
dotnet sln ../DistributionManagementSystem.sln add DistributionManagement.API

# Add project references
cd DistributionManagement.Application
dotnet add reference ../DistributionManagement.Domain

cd ../DistributionManagement.Infrastructure
dotnet add reference ../DistributionManagement.Domain
dotnet add reference ../DistributionManagement.Application

cd ../DistributionManagement.API
dotnet add reference ../DistributionManagement.Application
dotnet add reference ../DistributionManagement.Infrastructure

cd ../..
```

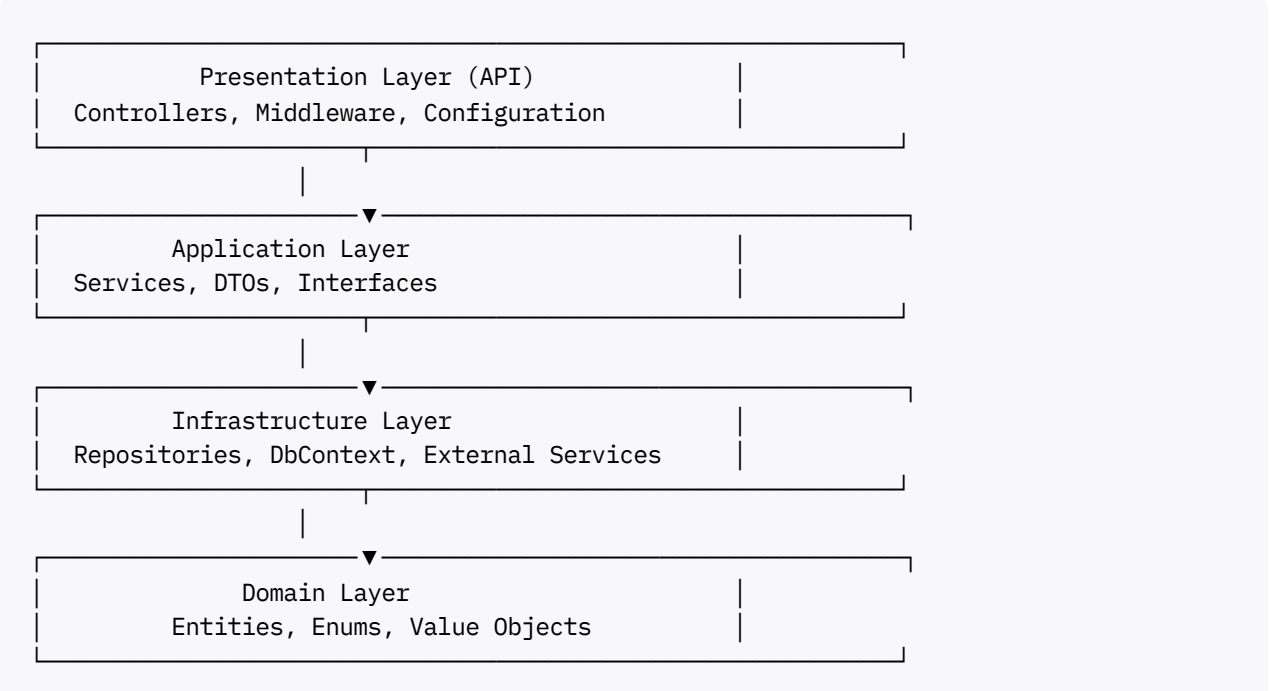
Step 2: Install Required NuGet Packages

```
# Domain Layer - minimal dependencies
cd src/DistributionManagement.Domain
# No additional packages needed

# Application Layer
cd ../DistributionManagement.Application
```

```
dotnet add package Microsoft.Extensions.DependencyInjection.Abstractions

# Infrastructure Layer
cd ../DistributionManagement.Infrastructure
dotnet add package Microsoft.EntityFrameworkCore
dotnet add package Microsoft.EntityFrameworkCore.Sqlite
dotnet add package Microsoft.EntityFrameworkCore.Design
dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL

# API Layer
cd ../DistributionManagement.API
dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer
dotnet add package Microsoft.AspNetCore.OpenApi
dotnet add package Swashbuckle.AspNetCore
dotnet add package Microsoft.EntityFrameworkCore.Tools
dotnet add package DotNetEnv
dotnet add package System.IdentityModel.Tokens.Jwt

cd ../..
```

Architecture

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



Dependency Flow: API → Infrastructure → Application → Domain

Domain Layer

File: `src/DistributionManagement.Domain/Enums/UserRole.cs`

```
namespace DistributionManagement.Domain.Enums;

/// &lt;summary&gt;
/// Defines the user roles in the system matching WSO2 IS roles
/// &lt;/summary&gt;
public enum UserRole
{
    /// &lt;summary&gt;
    /// Admin role with full access
    /// &lt;/summary&gt;
    YksAdmin = 1,

    /// &lt;summary&gt;
    /// User role with edit and view access
    /// &lt;/summary&gt;
    YksUser = 2,

    /// &lt;summary&gt;
    /// Test role with view-only access
    /// &lt;/summary&gt;
    YksTest = 3
}
```

File: `src/DistributionManagement.Domain/Entities/Product.cs`

```
using System.ComponentModel.DataAnnotations;

namespace DistributionManagement.Domain.Entities;

/// &lt;summary&gt;
/// Represents a product in the distribution management system
/// &lt;/summary&gt;
public class Product
{
    [Key]
    public Guid Id { get; set; }

    [Required]
    [MaxLength(200)]
    public string Name { get; set; } = string.Empty;

    [MaxLength(1000)]
    public string? Description { get; set; }

    [Required]
    public decimal Price { get; set; }

    [Required]
    public int StockQuantity { get; set; }
```

```
    [MaxLength(50)]
    public string? Category { get; set; }

    [MaxLength(100)]
    public string? Sku { get; set; }

    public bool IsActive { get; set; } = true;

    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

    public DateTime? UpdatedAt { get; set; }

    [MaxLength(100)]
    public string? CreatedBy { get; set; }

    [MaxLength(100)]
    public string? UpdatedBy { get; set; }
}
```

File: `src/DistributionManagement.Domain/Entities/User.cs`

```
using System.ComponentModel.DataAnnotations;
using DistributionManagement.Domain.Enums;

namespace DistributionManagement.Domain.Entities;

/// &lt;summary&gt;
/// Represents a user in the system (cached from WSO2 IS)
/// &lt;/summary&gt;
public class User
{
    [Key]
    public Guid Id { get; set; }

    [Required]
    [MaxLength(100)]
    public string Username { get; set; } = string.Empty;

    [Required]
    [EmailAddress]
    [MaxLength(200)]
    public string Email { get; set; } = string.Empty;

    [MaxLength(100)]
    public string? FirstName { get; set; }

    [MaxLength(100)]
    public string? LastName { get; set; }

    [Required]
    public UserRole Role { get; set; }

    public bool IsActive { get; set; } = true;
```

```
    public DateTime? LastLoginAt { get; set; }

    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

    public DateTime? UpdatedAt { get; set; }
}
```

File: `src/DistributionManagement.Domain/DistributionManagement.Domain.csproj`

```
&lt;Project Sdk="Microsoft.NET.Sdk"&gt;

  &lt;PropertyGroup&gt;
    &lt;TargetFramework&gt;net9.0&lt;/TargetFramework&gt;
    &lt;ImplicitUsings&gt;enable&lt;/ImplicitUsings&gt;
    &lt;Nullable&gt;enable&lt;/Nullable&gt;
  &lt;/PropertyGroup&gt;

  &lt;ItemGroup&gt;
    &lt;PackageReference Include="System.ComponentModel.Annotations" Version="9.0.0" /&gt
  &lt;/ItemGroup&gt;

&lt;/Project&gt;
```

Application Layer

File: `src/DistributionManagement.Application/DTOs/LoginRequest.cs`

```
using System.ComponentModel.DataAnnotations;

namespace DistributionManagement.Application.DTOs;

/// &lt;summary&gt;
/// Login request model for user authentication
/// &lt;/summary&gt;
public class LoginRequest
{
    [Required(ErrorMessage = "Username is required")]
    public string Username { get; set; } = string.Empty;

    [Required(ErrorMessage = "Password is required")]
    public string Password { get; set; } = string.Empty;
}
```

File: `src/DistributionManagement.Application/DTOs/LoginResponse.cs`

```
namespace DistributionManagement.Application.DTOs;

/// &lt;summary&gt;
/// Login response containing tokens and user information
```

```
/// &lt;/summary&gt;
public class LoginResponse
{
    public string AccessToken { get; set; } = string.Empty;
    public string TokenType { get; set; } = "Bearer";
    public int ExpiresIn { get; set; }
    public string? RefreshToken { get; set; }
    public UserInfoDto UserInfo { get; set; } = new();
}
```

File: `src/DistributionManagement.Application/DTOs/UserInfoDto.cs`

```
namespace DistributionManagement.Application.DTOs;

/// &lt;summary&gt;
/// User information data transfer object
/// &lt;/summary&gt;
public class UserInfoDto
{
    public string Username { get; set; } = string.Empty;
    public string Email { get; set; } = string.Empty;
    public string? FirstName { get; set; }
    public string? LastName { get; set; }
    public string Role { get; set; } = string.Empty;
    public List&lt;string&gt; Roles { get; set; } = new();
    public List&lt;string&gt; Groups { get; set; } = new();
}
```

File: `src/DistributionManagement.Application/DTOs/ProductDto.cs`

```
namespace DistributionManagement.Application.DTOs;

/// &lt;summary&gt;
/// Product data transfer object
/// &lt;/summary&gt;
public class ProductDto
{
    public Guid Id { get; set; }
    public string Name { get; set; } = string.Empty;
    public string? Description { get; set; }
    public decimal Price { get; set; }
    public int StockQuantity { get; set; }
    public string? Category { get; set; }
    public string? Sku { get; set; }
    public bool IsActive { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }
}
```

File: `src/DistributionManagement.Application/Interfaces/IAuthenticationService.cs`

```
using DistributionManagement.Application.DTOs;

namespace DistributionManagement.Application.Interfaces;

/// &lt;summary&gt;
/// Authentication service interface for WSO2 IS integration
/// &lt;/summary&gt;
public interface IAuthenticationService
{
    /// &lt;summary&gt;
    /// Authenticates user with username and password against WSO2 IS
    /// &lt;/summary&gt;
    Task&lt;LoginResponse&gt; LoginAsync(LoginRequest request);

    /// &lt;summary&gt;
    /// Gets user information from WSO2 IS using access token
    /// &lt;/summary&gt;
    Task&lt;UserInfoDto&gt; GetUserInfoAsync(string accessToken);

    /// &lt;summary&gt;
    /// Validates JWT token and extracts user information
    /// &lt;/summary&gt;
    Task&lt;bool&gt; ValidateTokenAsync(string token);
}
```

File: `src/DistributionManagement.Application/Interfaces/IProductService.cs`

```
using DistributionManagement.Application.DTOs;

namespace DistributionManagement.Application.Interfaces;

/// &lt;summary&gt;
/// Product service interface for business logic
/// &lt;/summary&gt;
public interface IProductService
{
    Task&lt;IEnumerable&lt;ProductDto&gt;&gt; GetAllProductsAsync();
    Task&lt;ProductDto?&gt; GetProductByIdAsync(Guid id);
    Task&lt;ProductDto&gt; CreateProductAsync(ProductDto productDto, string username);
    Task&lt;ProductDto?&gt; UpdateProductAsync(Guid id, ProductDto productDto, string use
    Task&lt;bool&gt; DeleteProductAsync(Guid id);
}
```

File: `src/DistributionManagement.Application/Interfaces/IProductRepository.cs`

```
using DistributionManagement.Domain.Entities;

namespace DistributionManagement.Application.Interfaces;

/// &lt;summary&gt;
```

```
/// Product repository interface for data access
/// &lt;/summary&gt;
public interface IProductRepository
{
    Task&lt;IEnumerable&lt;Product&gt;&gt; GetAllAsync();
    Task&lt;Product?&gt; GetByIdAsync(Guid id);
    Task&lt;Product&gt; AddAsync(Product product);
    Task&lt;Product&gt; UpdateAsync(Product product);
    Task&lt;bool&gt; DeleteAsync(Guid id);
    Task&lt;bool&gt; ExistsAsync(Guid id);
}
```

File: `src/DistributionManagement.Application/Services/ProductService.cs`

```
using DistributionManagement.Application.DTOs;
using DistributionManagement.Application.Interfaces;
using DistributionManagement.Domain.Entities;

namespace DistributionManagement.Application.Services;

/// &lt;summary&gt;
/// Product service implementation
/// &lt;/summary&gt;
public class ProductService : IProductService
{
    private readonly IProductRepository _productRepository;

    public ProductService(IProductRepository productRepository)
    {
        _productRepository = productRepository;
    }

    public async Task&lt;IEnumerable&lt;ProductDto&gt;&gt; GetAllProductsAsync()
    {
        var products = await _productRepository.GetAllAsync();
        return products.Select(MapToDto);
    }

    public async Task&lt;ProductDto?&gt; GetProductByIdAsync(Guid id)
    {
        var product = await _productRepository.GetByIdAsync(id);
        return product != null ? MapToDto(product) : null;
    }

    public async Task&lt;ProductDto&gt; CreateProductAsync(ProductDto productDto, string
    {
        var product = new Product
        {
            Id = Guid.NewGuid(),
            Name = productDto.Name,
            Description = productDto.Description,
            Price = productDto.Price,
            StockQuantity = productDto.StockQuantity,
            Category = productDto.Category,
            Sku = productDto.Sku,
```

```
            IsActive = productDto.IsActive,
            CreatedAt = DateTime.UtcNow,
            CreatedBy = username
        };

        var createdProduct = await _productRepository.AddAsync(product);
        return MapToDto(createdProduct);
    }

    public async Task&lt;ProductDto?&gt; UpdateProductAsync(Guid id, ProductDto productDt
    {
        var existingProduct = await _productRepository.GetByIdAsync(id);
        if (existingProduct == null)
            return null;

        existingProduct.Name = productDto.Name;
        existingProduct.Description = productDto.Description;
        existingProduct.Price = productDto.Price;
        existingProduct.StockQuantity = productDto.StockQuantity;
        existingProduct.Category = productDto.Category;
        existingProduct.Sku = productDto.Sku;
        existingProduct.IsActive = productDto.IsActive;
        existingProduct.UpdatedAt = DateTime.UtcNow;
        existingProduct.UpdatedBy = username;

        var updatedProduct = await _productRepository.UpdateAsync(existingProduct);
        return MapToDto(updatedProduct);
    }

    public async Task&lt;bool&gt; DeleteAsync(Guid id)
    {
        return await _productRepository.DeleteAsync(id);
    }

    private static ProductDto MapToDto(Product product)
    {
        return new ProductDto
        {
            Id = product.Id,
            Name = product.Name,
            Description = product.Description,
            Price = product.Price,
            StockQuantity = product.StockQuantity,
            Category = product.Category,
            Sku = product.Sku,
            IsActive = product.IsActive,
            CreatedAt = product.CreatedAt,
            UpdatedAt = product.UpdatedAt
        };
    }
}
```

File: `src/DistributionManagement.Application/DistributionManagement.Application.csproj`

```
&lt;Project Sdk="Microsoft.NET.Sdk"&gt;

  &lt;PropertyGroup&gt;
    &lt;TargetFramework&gt;net9.0&lt;/TargetFramework&gt;
    &lt;ImplicitUsings&gt;enable&lt;/ImplicitUsings&gt;
    &lt;Nullable&gt;enable&lt;/Nullable&gt;
  &lt;/PropertyGroup&gt;

  &lt;ItemGroup&gt;
    &lt;ProjectReference Include="..\DistributionManagement.Domain\DistributionManagement
  &lt;/ItemGroup&gt;

  &lt;ItemGroup&gt;
    &lt;PackageReference Include="Microsoft.Extensions.DependencyInjection.Abstractions"
  &lt;/ItemGroup&gt;

&lt;/Project&gt;
```

Infrastructure Layer

File: `src/DistributionManagement.Infrastructure/Data/ApplicationDbContext.cs`

```
using DistributionManagement.Domain.Entities;
using Microsoft.EntityFrameworkCore;

namespace DistributionManagement.Infrastructure.Data;

/// &lt;summary&gt;
/// Application database context
/// &lt;/summary&gt;
public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions&lt;ApplicationDbContext&gt; options)
        : base(options)
    {
    }

    public DbSet&lt;Product&gt; Products { get; set; }
    public DbSet&lt;User&gt; Users { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        // Product configuration
        modelBuilder.Entity&lt;Product&gt;(entity =&gt;
        {
            entity.HasKey(e =&gt; e.Id);
            entity.Property(e =&gt; e.Price).HasPrecision(18, 2);
            entity.HasIndex(e =&gt; e.Sku).IsUnique();
```

```
        });

        // User configuration
        modelBuilder.Entity&lt;User&gt;(entity =&gt;
        {
            entity.HasKey(e =&gt; e.Id);
            entity.HasIndex(e =&gt; e.Username).IsUnique();
            entity.HasIndex(e =&gt; e.Email).IsUnique();
        });
    }
}
```

File: `src/DistributionManagement.Infrastructure/Data/DbInitializer.cs`

```
using DistributionManagement.Domain.Entities;
using Microsoft.EntityFrameworkCore;

namespace DistributionManagement.Infrastructure.Data;

/// &lt;summary&gt;
/// Database initializer with seed data
/// &lt;/summary&gt;
public static class DbInitializer
{
    public static async Task InitializeAsync(ApplicationDbContext context)
    {
        await context.Database.EnsureCreatedAsync();

        // Check if database has data
        if (await context.Products.AnyAsync())
        {
            return; // Database has been seeded
        }

        // Seed 11 products
        var products = new List&lt;Product&gt;
        {
            new() { Id = Guid.NewGuid(), Name = "Product A", Description = "Description f
                    Price = 99.99m, StockQuantity = 100, Category = "Category 1", Sku = "S
            new() { Id = Guid.NewGuid(), Name = "Product B", Description = "Description f
                    Price = 149.99m, StockQuantity = 75, Category = "Category 1", Sku = "S
            new() { Id = Guid.NewGuid(), Name = "Product C", Description = "Description f
                    Price = 199.99m, StockQuantity = 50, Category = "Category 2", Sku = "S
            new() { Id = Guid.NewGuid(), Name = "Product D", Description = "Description f
                    Price = 249.99m, StockQuantity = 60, Category = "Category 2", Sku = "S
            new() { Id = Guid.NewGuid(), Name = "Product E", Description = "Description f
                    Price = 299.99m, StockQuantity = 40, Category = "Category 3", Sku = "S
            new() { Id = Guid.NewGuid(), Name = "Product F", Description = "Description f
                    Price = 349.99m, StockQuantity = 30, Category = "Category 3", Sku = "S
            new() { Id = Guid.NewGuid(), Name = "Product G", Description = "Description f
                    Price = 399.99m, StockQuantity = 25, Category = "Category 4", Sku = "S
            new() { Id = Guid.NewGuid(), Name = "Product H", Description = "Description f
                    Price = 449.99m, StockQuantity = 20, Category = "Category 4", Sku = "S
            new() { Id = Guid.NewGuid(), Name = "Product I", Description = "Description f
                    Price = 499.99m, StockQuantity = 15, Category = "Category 5", Sku = "S
```

```
            new() { Id = Guid.NewGuid(), Name = "Product J", Description = "Description f
                    Price = 549.99m, StockQuantity = 10, Category = "Category 5", Sku = "S
            new() { Id = Guid.NewGuid(), Name = "Product K", Description = "Description f
                    Price = 599.99m, StockQuantity = 5, Category = "Category 6", Sku = "SK
        };

        await context.Products.AddRangeAsync(products);
        await context.SaveChangesAsync();
    }
}
```

File: `src/DistributionManagement.Infrastructure/Repositories/ProductRepository.cs`

```
using DistributionManagement.Application.Interfaces;
using DistributionManagement.Domain.Entities;
using DistributionManagement.Infrastructure.Data;
using Microsoft.EntityFrameworkCore;

namespace DistributionManagement.Infrastructure.Repositories;

/// &lt;summary&gt;
/// Product repository implementation
/// &lt;/summary&gt;
public class ProductRepository : IProductRepository
{
    private readonly ApplicationDbContext _context;

    public ProductRepository(ApplicationDbContext context)
    {
        _context = context;
    }

    public async Task&lt;IEnumerable&lt;Product&gt;&gt; GetAllAsync()
    {
        return await _context.Products
            .Where(p =&gt; p.IsActive)
            .OrderBy(p =&gt; p.Name)
            .ToListAsync();
    }

    public async Task&lt;Product?&gt; GetByIdAsync(Guid id)
    {
        return await _context.Products.FindAsync(id);
    }

    public async Task&lt;Product&gt; AddAsync(Product product)
    {
        _context.Products.Add(product);
        await _context.SaveChangesAsync();
        return product;
    }

    public async Task&lt;Product&gt; UpdateAsync(Product product)
    {
        _context.Entry(product).State = EntityState.Modified;
```

```
        await _context.SaveChangesAsync();
        return product;
    }

    public async Task&lt;bool&gt; DeleteAsync(Guid id)
    {
        var product = await _context.Products.FindAsync(id);
        if (product == null)
            return false;

        _context.Products.Remove(product);
        await _context.SaveChangesAsync();
        return true;
    }

    public async Task&lt;bool&gt; ExistsAsync(Guid id)
    {
        return await _context.Products.AnyAsync(p =&gt; p.Id == id);
    }
}
```

File:
src/DistributionManagement.Infrastructure/ExternalServices/WSO2AuthenticationService.cs

```
using System.IdentityModel.Tokens.Jwt;
using System.Net.Http.Headers;
using System.Text;
using System.Text.Json;
using DistributionManagement.Application.DTOs;
using DistributionManagement.Application.Interfaces;
using Microsoft.Extensions.Configuration;

namespace DistributionManagement.Infrastructure.ExternalServices;

/// &lt;summary&gt;
/// WSO2 Identity Server authentication service implementation
/// Uses OAuth2 Resource Owner Password Credentials (ROPC) flow
/// &lt;/summary&gt;
public class WSO2AuthenticationService : IAuthenticationService
{
    private readonly HttpClient _httpClient;
    private readonly IConfiguration _configuration;
    private readonly string _tokenEndpoint;
    private readonly string _userInfoEndpoint;
    private readonly string _clientId;
    private readonly string _clientSecret;

    public WSO2AuthenticationService(HttpClient httpClient, IConfiguration configuration)
    {
        _httpClient = httpClient;
        _configuration = configuration;
        _tokenEndpoint = _configuration["WSO2:TokenEndpoint"] ?? throw new InvalidOperati
        _userInfoEndpoint = _configuration["WSO2:UserInfoEndpoint"] ?? throw new InvalidO
        _clientId = _configuration["WSO2:ClientId"] ?? throw new InvalidOperationExceptio
        _clientSecret = _configuration["WSO2:ClientSecret"] ?? throw new InvalidOperation
```

```
    }

    public async Task&lt;LoginResponse&gt; LoginAsync(LoginRequest request)
    {
        try
        {
            // Prepare token request using password grant type
            var tokenRequestData = new Dictionary&lt;string, string&gt;
            {
                { "grant_type", "password" },
                { "username", request.Username },
                { "password", request.Password },
                { "scope", "openid profile email roles groups" }
            };

            var content = new FormUrlEncodedContent(tokenRequestData);

            // Add Basic Authentication header (Client ID and Secret)
            var authHeader = Convert.ToBase64String(
                Encoding.UTF8.GetBytes($"{_clientId}:{_clientSecret}"));
            _httpClient.DefaultRequestHeaders.Authorization =
                new AuthenticationHeaderValue("Basic", authHeader);

            // Request token from WSO2 IS
            var response = await _httpClient.PostAsync(_tokenEndpoint, content);
            var responseBody = await response.Content.ReadAsStringAsync();

            if (!response.IsSuccessStatusCode)
            {
                throw new UnauthorizedAccessException($"Authentication failed: {responseB
            }

            var tokenResponse = JsonSerializer.Deserialize&lt;TokenResponse&gt;(
                responseBody,
                new JsonSerializerOptions { PropertyNameCaseInsensitive = true });

            if (tokenResponse == null || string.IsNullOrEmpty(tokenResponse.AccessToken))
            {
                throw new InvalidOperationException("Invalid token response from WSO2 IS'
            }

            // Get user information
            var userInfo = await GetUserInfoAsync(tokenResponse.AccessToken);

            return new LoginResponse
            {
                AccessToken = tokenResponse.AccessToken,
                TokenType = tokenResponse.TokenType ?? "Bearer",
                ExpiresIn = tokenResponse.ExpiresIn,
                RefreshToken = tokenResponse.RefreshToken,
                UserInfo = userInfo
            };
        }
        catch (HttpRequestException ex)
        {
            throw new InvalidOperationException($"Failed to communicate with WSO2 IS: {ex
```

```
        }
    }

    public async Task&lt;UserInfoDto&gt; GetUserInfoAsync(string accessToken)
    {
        try
        {
            _httpClient.DefaultRequestHeaders.Authorization =
                new AuthenticationHeaderValue("Bearer", accessToken);

            var response = await _httpClient.GetAsync(_userInfoEndpoint);
            var responseBody = await response.Content.ReadAsStringAsync();

            if (!response.IsSuccessStatusCode)
            {
                throw new UnauthorizedAccessException($"Failed to get user info: {respons
            }

            var userInfoResponse = JsonSerializer.Deserialize&lt;WSO2UserInfoResponse&gt;
                responseBody,
                new JsonSerializerOptions { PropertyNameCaseInsensitive = true });

            if (userInfoResponse == null)
            {
                throw new InvalidOperationException("Invalid user info response");
            }

            // Extract primary role from roles array
            var primaryRole = userInfoResponse.Roles?.FirstOrDefault() ?? string.Empty;

            return new UserInfoDto
            {
                Username = userInfoResponse.Username ?? userInfoResponse.PreferredUsernam
                Email = userInfoResponse.Email ?? string.Empty,
                FirstName = userInfoResponse.GivenName,
                LastName = userInfoResponse.FamilyName,
                Role = primaryRole,
                Roles = userInfoResponse.Roles ?? new List&lt;string&gt;(),
                Groups = userInfoResponse.Groups ?? new List&lt;string&gt;()
            };
        }
        catch (HttpRequestException ex)
        {
            throw new InvalidOperationException($"Failed to get user info from WSO2 IS: {
        }
    }

    public Task&lt;bool&gt; ValidateTokenAsync(string token)
    {
        try
        {
            var tokenHandler = new JwtSecurityTokenHandler();
            var jwtToken = tokenHandler.ReadJwtToken(token);

            // Check if token is expired
            if (jwtToken.ValidTo &lt; DateTime.UtcNow)
```

```
            {
                return Task.FromResult(false);
            }

            return Task.FromResult(true);
        }
        catch
        {
            return Task.FromResult(false);
        }
    }

    // Private classes for deserialization
    private class TokenResponse
    {
        public string AccessToken { get; set; } = string.Empty;
        public string? TokenType { get; set; }
        public int ExpiresIn { get; set; }
        public string? RefreshToken { get; set; }
        public string? IdToken { get; set; }
        public string? Scope { get; set; }
    }

    private class WSO2UserInfoResponse
    {
        public string? Sub { get; set; }
        public string? Username { get; set; }
        public string? PreferredUsername { get; set; }
        public string? Email { get; set; }
        public bool? EmailVerified { get; set; }
        public string? GivenName { get; set; }
        public string? FamilyName { get; set; }
        public string? Name { get; set; }
        public List&lt;string&gt;? Roles { get; set; }
        public List&lt;string&gt;? Groups { get; set; }
    }
}
```

File:

src/DistributionManagement.Infrastructure/DistributionManagement.Infrastructure.csproj

```
&lt;Project Sdk="Microsoft.NET.Sdk"&gt;

  &lt;PropertyGroup&gt;
    &lt;TargetFramework&gt;net9.0&lt;/TargetFramework&gt;
    &lt;ImplicitUsings&gt;enable&lt;/ImplicitUsings&gt;
    &lt;Nullable&gt;enable&lt;/Nullable&gt;
  &lt;/PropertyGroup&gt;

  &lt;ItemGroup&gt;
    &lt;ProjectReference Include="..\DistributionManagement.Domain\DistributionManagement
    &lt;ProjectReference Include="..\DistributionManagement.Application\DistributionManag
  &lt;/ItemGroup&gt;

  &lt;ItemGroup&gt;
```

```
    &lt;PackageReference Include="Microsoft.EntityFrameworkCore" Version="9.0.0" /&gt;
    &lt;PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite" Version="9.0.0" /
    &lt;PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="9.0.0"&g
      &lt;PrivateAssets&gt;all&lt;/PrivateAssets&gt;
      &lt;IncludeAssets&gt;runtime; build; native; contentfiles; analyzers; buildtransiti
    &lt;/PackageReference&gt;
    &lt;PackageReference Include="Npgsql.EntityFrameworkCore.PostgreSQL" Version="9.0.0"
    &lt;PackageReference Include="System.IdentityModel.Tokens.Jwt" Version="8.2.0" /&gt;
  &lt;/ItemGroup&gt;

&lt;/Project&gt;
```

API Layer

File: `src/DistributionManagement.API/Controllers/AuthController.cs`

```
using DistributionManagement.Application.DTOs;
using DistributionManagement.Application.Interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace DistributionManagement.API.Controllers;

[ApiController]
[Route("api/[controller]")]
public class AuthController : ControllerBase
{
    private readonly IAuthenticationService _authService;
    private readonly ILogger&lt;AuthController&gt; _logger;

    public AuthController(
        IAuthenticationService authService,
        ILogger&lt;AuthController&gt; logger)
    {
        _authService = authService;
        _logger = logger;
    }

    /// &lt;summary&gt;
    /// Login endpoint - authenticates user with WSO2 IS using password grant
    /// &lt;/summary&gt;
    [HttpPost("login")]
    [AllowAnonymous]
    public async Task&lt;IActionResult&gt; Login([FromBody] LoginRequest request)
    {
        try
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var response = await _authService.LoginAsync(request);
```

```
            _logger.LogInformation(
                "User {Username} logged in successfully with role {Role}",
                response.UserInfo.Username,
                response.UserInfo.Role);

            return Ok(response);
        }
        catch (UnauthorizedAccessException ex)
        {
            _logger.LogWarning(ex, "Login failed for user {Username}", request.Username);
            return Unauthorized(new { message = "Invalid username or password" });
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error during login for user {Username}", request.Userna
            return StatusCode(500, new { message = "An error occurred during login" });
        }
    }

    /// &lt;summary&gt;
    /// Get current user information
    /// &lt;/summary&gt;
    [HttpGet("me")]
    [Authorize]
    public async Task&lt;IActionResult&gt; GetCurrentUser()
    {
        try
        {
            var token = Request.Headers.Authorization.ToString().Replace("Bearer ", "");
            var userInfo = await _authService.GetUserInfoAsync(token);
            return Ok(userInfo);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error getting current user info");
            return StatusCode(500, new { message = "Failed to get user information" });
        }
    }

    /// &lt;summary&gt;
    /// Validate token endpoint
    /// &lt;/summary&gt;
    [HttpPost("validate")]
    [AllowAnonymous]
    public async Task&lt;IActionResult&gt; ValidateToken([FromBody] TokenValidationReques
    {
        try
        {
            var isValid = await _authService.ValidateTokenAsync(request.Token);
            return Ok(new { isValid });
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error validating token");
            return BadRequest(new { isValid = false });
```

```
        }
    }
}

public class TokenValidationRequest
{
    public string Token { get; set; } = string.Empty;
}
```

File: `src/DistributionManagement.API/Controllers/ProductController.cs`

```
using DistributionManagement.Application.DTOs;
using DistributionManagement.Application.Interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace DistributionManagement.API.Controllers;

[ApiController]
[Route("api/[controller]")]
[Authorize]
public class ProductController : ControllerBase
{
    private readonly IProductService _productService;
    private readonly ILogger&lt;ProductController&gt; _logger;

    public ProductController(
        IProductService productService,
        ILogger&lt;ProductController&gt; logger)
    {
        _productService = productService;
        _logger = logger;
    }

    /// &lt;summary&gt;
    /// Get all products - accessible by all authenticated users
    /// &lt;/summary&gt;
    [HttpGet]
    [Authorize(Roles = "yks_admin,yks_user,yks_test")]
    public async Task&lt;IActionResult&gt; GetAll()
    {
        try
        {
            var products = await _productService.GetAllProductsAsync();
            return Ok(products);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error getting all products");
            return StatusCode(500, new { message = "Failed to retrieve products" });
        }
    }

    /// &lt;summary&gt;
```

```
    /// Get product by ID - accessible by all authenticated users
    /// &lt;/summary&gt;
    [HttpGet("{id}")]
    [Authorize(Roles = "yks_admin,yks_user,yks_test")]
    public async Task&lt;IActionResult&gt; GetById(Guid id)
    {
        try
        {
            var product = await _productService.GetProductByIdAsync(id);
            if (product == null)
            {
                return NotFound(new { message = "Product not found" });
            }
            return Ok(product);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error getting product {ProductId}", id);
            return StatusCode(500, new { message = "Failed to retrieve product" });
        }
    }

    /// &lt;summary&gt;
    /// Create new product - only admin and user roles
    /// &lt;/summary&gt;
    [HttpPost]
    [Authorize(Roles = "yks_admin,yks_user")]
    public async Task&lt;IActionResult&gt; Create([FromBody] ProductDto productDto)
    {
        try
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var username = User.FindFirst(ClaimTypes.Name)?.Value ?? "unknown";
            var product = await _productService.CreateProductAsync(productDto, username);

            return CreatedAtAction(nameof(GetById), new { id = product.Id }, product);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error creating product");
            return StatusCode(500, new { message = "Failed to create product" });
        }
    }

    /// &lt;summary&gt;
    /// Update product - only admin and user roles
    /// &lt;/summary&gt;
    [HttpPut("{id}")]
    [Authorize(Roles = "yks_admin,yks_user")]
    public async Task&lt;IActionResult&gt; Update(Guid id, [FromBody] ProductDto productD
    {
        try
```

```
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var username = User.FindFirst(ClaimTypes.Name)?.Value ?? "unknown";
            var product = await _productService.UpdateProductAsync(id, productDto, userna

            if (product == null)
            {
                return NotFound(new { message = "Product not found" });
            }

            return Ok(product);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error updating product {ProductId}", id);
            return StatusCode(500, new { message = "Failed to update product" });
        }
    }

    /// &lt;summary&gt;
    /// Delete product - only admin role
    /// &lt;/summary&gt;
    [HttpDelete("{id}")]
    [Authorize(Roles = "yks_admin")]
    public async Task&lt;IActionResult&gt; Delete(Guid id)
    {
        try
        {
            var result = await _productService.DeleteProductAsync(id);
            if (!result)
            {
                return NotFound(new { message = "Product not found" });
            }

            return NoContent();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error deleting product {ProductId}", id);
            return StatusCode(500, new { message = "Failed to delete product" });
        }
    }
}
```

File: `src/DistributionManagement.API/Controllers/UserController.cs`

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace DistributionManagement.API.Controllers;
```

```
[ApiController]
[Route("api/[controller]")]
[Authorize(Roles = "yks_admin")]
public class UserController : ControllerBase
{
    private readonly ILogger&lt;UserController&gt; _logger;

    public UserController(ILogger&lt;UserController&gt; logger)
    {
        _logger = logger;
    }

    /// &lt;summary&gt;
    /// Get all users - only accessible by admin
    /// &lt;/summary&gt;
    [HttpGet]
    public IActionResult GetAllUsers()
    {
        // This would typically fetch users from WSO2 IS or local database
        // For now, returning mock data
        var users = new[]
        {
            new { Username = "yks", Email = "yks@example.com", Role = "yks_admin" },
            new { Username = "yks1", Email = "yks1@example.com", Role = "yks_test" },
            new { Username = "bimdevops", Email = "bimdevops@example.com", Role = "yks_us
        };

        return Ok(users);
    }
}
```

File: `src/DistributionManagement.API/Middleware/ExceptionHandlingMiddleware.cs`

```
using System.Net;
using System.Text.Json;

namespace DistributionManagement.API.Middleware;

/// &lt;summary&gt;
/// Global exception handling middleware
/// &lt;/summary&gt;
public class ExceptionHandlingMiddleware
{
    private readonly RequestDelegate _next;
    private readonly ILogger&lt;ExceptionHandlingMiddleware&gt; _logger;

    public ExceptionHandlingMiddleware(
        RequestDelegate next,
        ILogger&lt;ExceptionHandlingMiddleware&gt; logger)
    {
        _next = next;
        _logger = logger;
    }

    public async Task InvokeAsync(HttpContext context)
```

```
    {
        try
        {
            await _next(context);
        }
        catch (Exception ex)
        {
            await HandleExceptionAsync(context, ex);
        }
    }

    private Task HandleExceptionAsync(HttpContext context, Exception exception)
    {
        _logger.LogError(exception, "An unhandled exception occurred");

        var statusCode = exception switch
        {
            UnauthorizedAccessException =&gt; HttpStatusCode.Unauthorized,
            InvalidOperationException =&gt; HttpStatusCode.BadRequest,
            _ =&gt; HttpStatusCode.InternalServerError
        };

        var response = new
        {
            error = exception.Message,
            statusCode = (int)statusCode,
            timestamp = DateTime.UtcNow
        };

        context.Response.ContentType = "application/json";
        context.Response.StatusCode = (int)statusCode;

        var options = new JsonSerializerOptions
        {
            PropertyNamingPolicy = JsonNamingPolicy.CamelCase
        };

        return context.Response.WriteAsync(JsonSerializer.Serialize(response, options));
    }
}
```

File: `src/DistributionManagement.API/Configuration/AuthenticationConfiguration.cs`

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;

namespace DistributionManagement.API.Configuration;

/// &lt;summary&gt;
/// JWT Authentication configuration
/// &lt;/summary&gt;
public static class AuthenticationConfiguration
{
    public static IServiceCollection AddJwtAuthentication(
```

```
        this IServiceCollection services,
        IConfiguration configuration)
    {
        var jwtSettings = configuration.GetSection("JWT");
        var issuer = jwtSettings["Issuer"] ?? throw new InvalidOperationException("JWT Is
        var audience = jwtSettings["Audience"] ?? throw new InvalidOperationException("JW

        services.AddAuthentication(options =&gt;
        {
            options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
            options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
        })
        .AddJwtBearer(options =&gt;
        {
            options.Authority = issuer;
            options.Audience = audience;
            options.RequireHttpsMetadata = true;
            options.SaveToken = true;

            options.TokenValidationParameters = new TokenValidationParameters
            {
                ValidateIssuer = true,
                ValidIssuer = issuer,
                ValidateAudience = true,
                ValidAudience = audience,
                ValidateIssuerSigningKey = false, // WSO2 provides JWKS endpoint
                ValidateLifetime = true,
                ClockSkew = TimeSpan.FromMinutes(5),
                RoleClaimType = "roles" // Map roles claim from WSO2 token
            };

            options.Events = new JwtBearerEvents
            {
                OnAuthenticationFailed = context =&gt;
                {
                    if (context.Exception.GetType() == typeof(SecurityTokenExpiredExcepti
                    {
                        context.Response.Headers.Append("Token-Expired", "true");
                    }
                    return Task.CompletedTask;
                }
            };
        });

        services.AddAuthorization();

        return services;
    }
}
```

File: `src/DistributionManagement.API/Configuration/DependencyInjection.cs`

```
using DistributionManagement.Application.Interfaces;
using DistributionManagement.Application.Services;
using DistributionManagement.Infrastructure.Data;
using DistributionManagement.Infrastructure.ExternalServices;
using DistributionManagement.Infrastructure.Repositories;
using Microsoft.EntityFrameworkCore;

namespace DistributionManagement.API.Configuration;

/// &lt;summary&gt;
/// Dependency injection configuration
/// &lt;/summary&gt;
public static class DependencyInjection
{
    public static IServiceCollection AddApplicationServices(
        this IServiceCollection services,
        IConfiguration configuration)
    {
        // Database configuration
        var connectionString = configuration.GetConnectionString("DefaultConnection");
        services.AddDbContext&lt;ApplicationDbContext&gt;(options =&gt;
        {
            // Use SQLite for development, PostgreSQL for production
            if (configuration["Database:Provider"] == "PostgreSQL")
            {
                options.UseNpgsql(connectionString);
            }
            else
            {
                options.UseSqlite(connectionString ?? "Data Source=distribution.db");
            }
        });

        // HttpClient for WSO2 authentication
        services.AddHttpClient&lt;IAuthenticationService, WSO2AuthenticationService&gt;(c
        {
            client.Timeout = TimeSpan.FromSeconds(30);
        });

        // Register repositories
        services.AddScoped&lt;IProductRepository, ProductRepository&gt;();

        // Register services
        services.AddScoped&lt;IProductService, ProductService&gt;();

        // Add CORS
        services.AddCors(options =&gt;
        {
            options.AddPolicy("AllowAll", builder =&gt;
            {
                builder
                    .AllowAnyOrigin()
                    .AllowAnyMethod()
                    .AllowAnyHeader();
```

```
            });
        });

        return services;
    }
}
```

File: `src/DistributionManagement.API/Program.cs`

```
using DistributionManagement.API.Configuration;
using DistributionManagement.API.Middleware;
using DistributionManagement.Infrastructure.Data;
using DotNetEnv;

var builder = WebApplication.CreateBuilder(args);

// Load environment variables from .env file
if (File.Exists(".env"))
{
    Env.Load();
}

// Override configuration with environment variables
builder.Configuration.AddEnvironmentVariables();

// Add services to the container
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(c =&gt;
{
    c.SwaggerDoc("v1", new() {
        Title = "Distribution Management API",
        Version = "v1",
        Description = "API for Distribution Management System with WSO2 IS integration"
    });

    // Add JWT authentication to Swagger
    c.AddSecurityDefinition("Bearer", new Microsoft.OpenApi.Models.OpenApiSecurityScheme
    {
        Description = "JWT Authorization header using the Bearer scheme. Enter 'Bearer' [
        Name = "Authorization",
        In = Microsoft.OpenApi.Models.ParameterLocation.Header,
        Type = Microsoft.OpenApi.Models.SecuritySchemeType.ApiKey,
        Scheme = "Bearer"
    });

    c.AddSecurityRequirement(new Microsoft.OpenApi.Models.OpenApiSecurityRequirement
    {
        {
            new Microsoft.OpenApi.Models.OpenApiSecurityScheme
            {
                Reference = new Microsoft.OpenApi.Models.OpenApiReference
                {
                    Type = Microsoft.OpenApi.Models.ReferenceType.SecurityScheme,
                    Id = "Bearer"
```

```
                }
            },
            Array.Empty&lt;string&gt;()
        }
    });
});

// Add JWT authentication
builder.Services.AddJwtAuthentication(builder.Configuration);

// Add application services
builder.Services.AddApplicationServices(builder.Configuration);

var app = builder.Build();

// Initialize database
using (var scope = app.Services.CreateScope())
{
    var dbContext = scope.ServiceProvider.GetRequiredService&lt;ApplicationDbContext&gt;(
    await DbInitializer.InitializeAsync(dbContext);
}

// Configure the HTTP request pipeline
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseCors("AllowAll");

// Add exception handling middleware
app.UseMiddleware&lt;ExceptionHandlingMiddleware&gt;();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllers();

app.Run();
```

File: `src/DistributionManagement.API/appsettings.json`

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=distribution.db"
```

```
  },
  "Database": {
    "Provider": "SQLite"
  },
  "WSO2": {
    "TokenEndpoint": "https://iam.bimats.com/oauth2/token",
    "UserInfoEndpoint": "https://iam.bimats.com/oauth2/userinfo",
    "WellKnownEndpoint": "https://iam.bimats.com/oauth2/token/.well-known/openid-configu
    "ClientId": "7eWli_Xh2SdqbNQHfex0nZfC1mUa",
    "ClientSecret": "oyvSQgpwsvJWrbncCsqTR9lbVjPr4Sq8abdUhfN11R4a"
  },
  "JWT": {
    "Issuer": "https://iam.bimats.com/oauth2/token",
    "Audience": "7eWli_Xh2SdqbNQHfex0nZfC1mUa"
  }
}
```

File: `src/DistributionManagement.API/appsettings.Development.json`

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Debug",
      "Microsoft.AspNetCore": "Information"
    }
  },
  "Database": {
    "Provider": "SQLite"
  }
}
```

File: `src/DistributionManagement.API/appsettings.Production.json`

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "Database": {
    "Provider": "PostgreSQL"
  },
  "ConnectionStrings": {
    "DefaultConnection": ""
  }
}
```

File: `src/DistributionManagement.API/.env.example`

```
# WSO2 Identity Server Configuration
WSO2__TokenEndpoint=https://iam.bimats.com/oauth2/token
WSO2__UserInfoEndpoint=https://iam.bimats.com/oauth2/userinfo
WSO2__ClientId=7eWli_Xh2SdqbNQHfex0nZfC1mUa
WSO2__ClientSecret=oyvSQgpwsvJWrbncCsqTR9lbVjPr4Sq8abdUhfN11R4a

# JWT Configuration
JWT__Issuer=https://iam.bimats.com/oauth2/token
JWT__Audience=7eWli_Xh2SdqbNQHfex0nZfC1mUa

# Database Configuration
Database__Provider=SQLite
ConnectionStrings__DefaultConnection=Data Source=distribution.db

# ASPNET Environment
ASPNETCORE_ENVIRONMENT=Development
```

File: `src/DistributionManagement.API/DistributionManagement.API.csproj`

```
&lt;Project Sdk="Microsoft.NET.Sdk.Web"&gt;

  &lt;PropertyGroup&gt;
    &lt;TargetFramework&gt;net9.0&lt;/TargetFramework&gt;
    &lt;Nullable&gt;enable&lt;/Nullable&gt;
    &lt;ImplicitUsings&gt;enable&lt;/ImplicitUsings&gt;
  &lt;/PropertyGroup&gt;

  &lt;ItemGroup&gt;
    &lt;PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version=
    &lt;PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="9.0.0" /&gt;
    &lt;PackageReference Include="Swashbuckle.AspNetCore" Version="7.2.0" /&gt;
    &lt;PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="9.0.0"&gt
      &lt;PrivateAssets&gt;all&lt;/PrivateAssets&gt;
      &lt;IncludeAssets&gt;runtime; build; native; contentfiles; analyzers; buildtransiti
    &lt;/PackageReference&gt;
    &lt;PackageReference Include="DotNetEnv" Version="3.1.2" /&gt;
    &lt;PackageReference Include="System.IdentityModel.Tokens.Jwt" Version="8.2.0" /&gt;
  &lt;/ItemGroup&gt;

  &lt;ItemGroup&gt;
    &lt;ProjectReference Include="..\DistributionManagement.Application\DistributionManag
    &lt;ProjectReference Include="..\DistributionManagement.Infrastructure\DistributionMa
  &lt;/ItemGroup&gt;

&lt;/Project&gt;
```

Configuration

Environment Variables Setup

For macOS Terminal:

```
# Add to ~/.zshrc or ~/.bash_profile
export WSO2__TokenEndpoint="https://iam.bimats.com/oauth2/token"
export WSO2__UserInfoEndpoint="https://iam.bimats.com/oauth2/userinfo"
export WSO2__ClientId="7eWli_Xh2SdqbNQHfex0nZfC1mUa"
export WSO2__ClientSecret="oyvSQgpwsvJWrbncCsqTR9lbVjPr4Sq8abdUhfN11R4a"

# Reload profile
source ~/.zshrc
```

Using .env file (Recommended):

1. Copy `.env.example` to `.env` in the API project directory
2. Update values as needed
3. The `.env` file is loaded automatically by the application

Database Migration

```
cd src/DistributionManagement.API

# Create initial migration
dotnet ef migrations add InitialCreate --project ../DistributionManagement.Infrastructure

# Apply migration
dotnet ef database update --project ../DistributionManagement.Infrastructure
```

Deployment

Run the Application

```
cd src/DistributionManagement.API

# Development
dotnet run

# Production
dotnet run --environment Production
```

The API will be available at:

- HTTP: http://localhost:5000
- HTTPS: https://localhost:5001

- Swagger UI: https://localhost:5001/swagger

Build and Publish

```
# Build release
dotnet build -c Release

# Publish for deployment
dotnet publish -c Release -o ./publish
```

Docker Deployment (Optional)

Create `Dockerfile` in the API project:

```
FROM mcr.microsoft.com/dotnet/aspnet:9.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:9.0 AS build
WORKDIR /src
COPY ["DistributionManagement.API/DistributionManagement.API.csproj", "DistributionManage
COPY ["DistributionManagement.Infrastructure/DistributionManagement.Infrastructure.csproj
COPY ["DistributionManagement.Application/DistributionManagement.Application.csproj", "Di
COPY ["DistributionManagement.Domain/DistributionManagement.Domain.csproj", "Distributior
RUN dotnet restore "DistributionManagement.API/DistributionManagement.API.csproj"
COPY . .
WORKDIR "/src/DistributionManagement.API"
RUN dotnet build "DistributionManagement.API.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "DistributionManagement.API.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "DistributionManagement.API.dll"]
```

Testing

Test Authentication with cURL

1. Login:

```
curl -X POST https://localhost:5001/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "yks",
    "password": "your_password"
```

```
  }' \
  -k
```

2. Get Products (with token):

```
curl -X GET https://localhost:5001/api/product \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN" \
  -k
```

3. Create Product (admin/user only):

```
curl -X POST https://localhost:5001/api/product \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "New Product",
    "description": "Product Description",
    "price": 99.99,
    "stockQuantity": 100,
    "category": "Category A",
    "sku": "SKU999",
    "isActive": true
  }' \
  -k
```

Using Swagger UI

1. Navigate to https://localhost:5001/swagger
2. Click "Authorize" button
3. Login using the `/api/auth/login` endpoint
4. Copy the `accessToken` from the response
5. Click "Authorize" again and enter: `Bearer YOUR_ACCESS_TOKEN`
6. Test all endpoints with proper authorization

Best Practices & Security

Security Considerations

1. **Never commit secrets**: Use `.env` files and add them to `.gitignore`
2. **Use HTTPS**: Always use HTTPS in production
3. **Token Expiration**: Implement token refresh mechanism
4. **Input Validation**: Always validate user input
5. **Logging**: Implement comprehensive logging (avoid logging sensitive data)
6. **Rate Limiting**: Add rate limiting middleware

7. **CORS**: Configure CORS properly for production

Production Checklist

- [ ] Use PostgreSQL instead of SQLite
- [ ] Configure proper connection strings
- [ ] Set up SSL/TLS certificates
- [ ] Configure logging to external service (e.g., Seq, Application Insights)
- [ ] Implement health checks
- [ ] Add API versioning
- [ ] Set up monitoring and alerting
- [ ] Configure backup strategy
- [ ] Implement rate limiting
- [ ] Review and update CORS policy

.gitignore Configuration

```
## Ignore Visual Studio temporary files, build results, and
## files generated by popular Visual Studio add-ons.

# User-specific files
*.suo
*.user
*.userosscache
*.sln.docstates

# Build results
[Dd]ebug/
[Dd]ebugPublic/
[Rr]elease/
[Rr]eleases/
x64/
x86/
build/
bld/
[Bb]in/
[Oo]bj/

# Environment files
.env
.env.local
.env.production

# Database files
*.db
*.db-shm
*.db-wal

# VS Code
```

```
.vscode/

# JetBrains Rider
.idea/
*.sln.iml

# Mac
.DS_Store
```

Additional Resources

WSO2 IS 7.1 Documentation

- Token Endpoint: [1] [2] [3]
- Password Grant Flow: [2] [4] [5]
- Role-Based Access Control: [6] [7] [8]

.NET 9 Documentation

- Authentication: [9] [10] [11]
- Configuration: [12] [13] [14] [15]
- Clean Architecture: [16] [17] [18] [19]

OAuth2 Best Practices

- Resource Owner Password Credentials: [2] [3]
- Token Management: [20] [21] [22]
- Security Considerations: [2] [3] [23]

Troubleshooting

Common Issues

1. Authentication Failed

- Verify WSO2 IS credentials
- Check if user exists in WSO2 IS
- Ensure password grant type is enabled for the application

2. Role-Based Authorization Not Working

- Verify role claim is present in JWT token
- Check role mapping in JWT configuration
- Ensure role names match exactly (case-sensitive)

3. Database Connection Issues

- Verify connection string
- Check database provider configuration
- Ensure migrations are applied

4. HTTPS Certificate Errors

- Trust development certificate: `dotnet dev-certs https --trust`
- Use `-k` flag with curl for testing

Support

For issues and questions:

- WSO2 IS Documentation: https://is.docs.wso2.com/en/7.1.0/
- .NET Documentation: https://learn.microsoft.com/en-us/dotnet/
- Stack Overflow: Use tags `wso2-identity-server`, `asp.net-core`, `oauth-2.0`

Document Version: 1.0
Last Updated: November 2025
Compatible With: WSO2 IS 7.1, .NET 9

[24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89]

⁂

1. https://arxiv.org/pdf/1808.10624.pdf
2. https://curity.io/resources/learn/oauth-resource-owner-password-credential-flow/
3. https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth-ropc
4. https://stackoverflow.com/questions/79546062/wso2-is-7-1-0-ga-how-to-get-token-for-sub-organization-user-using-password-gran
5. https://stackoverflow.com/questions/79678142/password-grant-type-is-not-visible-in-wso2-identity-server-7-0
6. https://stackoverflow.com/questions/53399478/rbacrole-back-access-control-in-wso2-api-manager-and-identity-server
7. https://hackernoon.com/rbac-role-based-access-control-via-oauth20-scopes-with-wso2-identity-server-1t2f3t3y
8. https://blog.dreamfactory.com/oauth-scopes-vs-rbac-key-differences
9. https://dev.to/imzihad21/custom-role-based-authorization-with-jwt-in-aspnet-core-e7p
10. https://learn.microsoft.com/en-us/aspnet/core/security/authorization/roles?view=aspnetcore-9.0
11. https://learn.microsoft.com/en-us/aspnet/core/security/authentication/configure-jwt-bearer-authentication?view=aspnetcore-9.0
12. https://stackoverflow.com/questions/75331824/net-7-environment-variables-in-appsettings-json
13. https://enlabsoftware.com/development/dotnet-core-environment-how-config-examples.html

14. https://dev.to/imzihad21/managing-production-configurations-in-aspnet-core-webapi-using-environment-variables-3nmf
15. https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-9.0
16. https://atalupadhyay.wordpress.com/2024/11/19/clean-architecture-with-asp-net-core-9-a-comprehensive-hands-on-laboratory/
17. https://www.c-sharpcorner.com/article/a-guide-for-building-a-net-project-with-clean-architecture/
18. https://www.youtube.com/watch?v=VKc88NWf4lw
19. https://ardalis.com/clean-architecture-asp-net-core/
20. https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-client-creds-grant-flow
21. https://www.shapepayroll.com/blog/dot-net-client-credentials-authentication/
22. https://webscraping.ai/faq/httpclient-c/can-i-use-httpclient-c-to-perform-oauth-authentication
23. https://auth0.com/blog/how-to-validate-jwt-dotnet/
24. https://aws.amazon.com/marketplace/pp/prodview-z6cv6teuzkl4u
25. http://arxiv.org/pdf/1807.11052.pdf
26. http://arxiv.org/pdf/2501.14397.pdf
27. https://www.mdpi.com/1424-8220/21/17/5716
28. http://arxiv.org/pdf/0909.1639.pdf
29. https://arxiv.org/pdf/2301.11092.pdf
30. http://eudl.eu/pdf/10.4108/eai.5-10-2015.150479
31. https://www.mdpi.com/1424-8220/19/8/1884/pdf
32. https://www.youtube.com/watch?v=C82qlYq8aCQ
33. https://github.com/carolemad965/ASP.NET-Core-Web-API-with-JWT-Authentication
34. https://www.c-sharpcorner.com/article/securing-asp-net-core-web-api-with-jwt-authentication-and-role-based-authorizati/
35. http://arxiv.org/pdf/2301.12496.pdf
36. https://www.youtube.com/watch?v=6EEltKS8AwA
37. https://learn.microsoft.com/en-us/aspnet/core/fundamentals/minimal-apis/security?view=aspnetcore-9.0
38. https://onepetro.org/SPEMEOS/proceedings/23MEOS/23MEOS/D021S066R003/517338
39. https://www.semanticscholar.org/paper/3788a9740e4db4a16a88f3164b0e544ff3d0a164
40. https://amkcorp.in/index.php/books/concept-to-completion/
41. https://academic.oup.com/ofid/article/doi/10.1093/ofid/ofae631.1911/7987953
42. https://www.semanticscholar.org/paper/8f25aa2eca9966610bbdb8ed89867b121a155ba5
43. https://www.semanticscholar.org/paper/2ed0d26ac3d3d0ef789405e8e2aef35c6b5027ad
44. https://onepetro.org/SPEAPOG/proceedings/20APOG/20APOG/D013S103R010/451794
45. https://www.semanticscholar.org/paper/f3934a481b85bfe1b44edb08f9cd2d077e09f491
46. http://arxiv.org/pdf/2404.01803.pdf
47. https://www.semanticscholar.org/paper/ea089f54852526020bb4bb99c023e7f9bead25c2
48. https://www.semanticscholar.org/paper/a48dbae05e03f481fc85bdc2bb32467e4587cca8
49. https://joss.theoj.org/papers/10.21105/joss.05350.pdf

50. http://arxiv.org/pdf/1607.03748.pdf
51. http://arxiv.org/pdf/1407.6103.pdf
52. https://arxiv.org/ftp/arxiv/papers/2312/2312.03049.pdf
53. http://arxiv.org/pdf/0905.4613.pdf
54. https://arxiv.org/ftp/arxiv/papers/2307/2307.06340.pdf
55. https://arxiv.org/pdf/1606.07941.pdf
56. https://ecotipe.ubb.ac.id/index.php/ecotipe/article/download/4327/2241
57. https://arxiv.org/pdf/1601.01229.pdf
58. https://developers.google.com/api-client-library/dotnet/guide/aaa_oauth
59. https://www.linkedin.com/posts/maninder-singh-455876b8_dotnet-dotnet9-softwarearchitecture-activity-7384934756502675456-kJYw
60. https://arxiv.org/pdf/2307.16607.pdf
61. https://stackoverflow.com/questions/38494279/how-do-i-get-an-oauth-2-0-authentication-token-in-c-sharp
62. https://www.youtube.com/watch?v=zw-ZtB1BNl8
63. https://www.youtube.com/watch?v=m0yjMV7LA1U
64. https://learn.microsoft.com/en-us/aspnet/core/fundamentals/environments?view=aspnetcore-9.0
65. https://ironpdf.com/blog/net-help/csharp-oauth2-guide/
66. https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures
67. https://www.reddit.com/r/csharp/comments/y1u6ni/use_env_variables_in_appsettingsjson/
68. https://www.mdpi.com/1424-8220/22/5/1703/pdf
69. https://www.mdpi.com/1424-8220/11/5/5020/pdf
70. https://zenodo.org/records/13254832/files/CR_Zero_Trust_Single_Sign_on___IEEE_Conf.pdf
71. https://www.mdpi.com/1424-8220/22/22/8793/pdf?version=1668424931
72. http://arxiv.org/pdf/1412.1623.pdf
73. https://www.mdpi.com/1424-8220/13/8/9589/pdf
74. https://www.mdpi.com/1424-8220/14/6/10081/pdf
75. https://arxiv.org/pdf/1203.0640.pdf
76. https://stackoverflow.com/questions/75044125/oauth2-token-request-failing-using-c-sharp-httpclient-but-working-in-postman
77. https://trailheadtechnology.com/calling-apis-with-oauth2-access-tokens-the-easy-way/
78. https://www.reddit.com/r/dotnet/comments/17zlkr7/how_authorize_automatically_reads_role_claim_from/
79. https://arxiv.org/pdf/2103.02579.pdf
80. https://learn.microsoft.com/en-us/answers/questions/729834/create-client-token-oauth2-question
81. https://stackoverflow.com/questions/42036810/asp-net-core-jwt-mapping-role-claims-to-claimsidentity
82. https://auth0.com/blog/call-protected-api-in-aspnet-core/
83. https://docs.duendesoftware.com/identityserver/tokens/requesting/
84. https://www.milanjovanovic.tech/blog/extending-httpclient-with-delegating-handlers-in-aspnetcore
85. https://www.codeproject.com/articles/ASP-NET-Core-WebAPI-secured-using-OAuth-Client-Cre

86. https://dev.to/angelodotnet/how-to-manage-roles-permissions-and-more-using-jwt-token-and-aspnet-core-identity-11k0
87. https://www.answeroverflow.com/m/1098148327884857415
88. https://juliocasal.com/blog/diagnosing-jwt-failures-in-asp-net-core-the-right-way
89. https://stackoverflow.com/questions/tagged/wso2-identity-server