Auth Service

Overview

The AuthService is a microservice component of the OllamaNet system responsible for user authentication and authorization. It provides comprehensive identity management functionality including user registration, login, password management, and role-based access control. The service issues JWT tokens for authenticated users and manages refresh tokens for persistent sessions.

Core Functionality

User Authentication

- Secure user registration with automatic role assignment
- Login with username/password validation
- JWT token generation with appropriate claims
- Refresh token functionality for persistent sessions
- Secure logout with token revocation

User Management

- User profile management
- Role assignment and management
- Password change and reset functionality
- Account status management
- Root folder creation for new users

Token Management

- JWT token generation with configurable expiration
- Refresh token creation and validation
- Token revocation on logout
- Cookie-based token storage
- Claims generation based on user data

Security Features

- Password policy enforcement through ASP.NET Identity
- Role-based authorization
- Secure cookie handling for refresh tokens
- Token validation with comprehensive checks
- Protection against common authentication vulnerabilities

Architecture

Layered Architecture

The AuthService follows a clean, layered architecture:

- **API Layer**: AuthController handling HTTP requests and responses
- **Service Layer**: IAuthService implementation with business logic
- **Security Layer**: JWT token generation and validation (JWTManager)
- **Identity Layer**: User and role management via ASP.NET Identity
- **Data Access Layer**: Repository pattern for data operations
- **Infrastructure Layer**: Cross-cutting concerns (DataSeeding, etc.)

Key Components

- **AuthController**: Handles authentication endpoints
- **AuthService**: Implements authentication business logic
- **JWTManager**: Handles token creation and validation
- **UserManager**: ASP.NET Identity component for user operations
- **RoleManager**: ASP.NET Identity component for role operations
- **RefreshToken**: Entity for persistent session management
- **DataSeeding**: Infrastructure for initial data population

Authentication Flow

Registration Flow

1. Receive registration request with username, email, password
2. Validate input data
3. Create user with UserManager
4. Assign "User" role
5. Create root folder for the user
6. Generate JWT and refresh tokens
7. Set refresh token in HTTP-only cookie
8. Return tokens and user info

Login Flow

1. Receive login request with username/email and password
2. Validate credentials with UserManager
3. Generate JWT and refresh tokens
4. Set refresh token in HTTP-only cookie
5. Return tokens and user info

Token Refresh Flow

1. Extract refresh token from cookie or request body
2. Validate token existence and expiration
3. Check if token is already revoked
4. Revoke the used refresh token
5. Generate new JWT and refresh tokens
6. Set new refresh token in HTTP-only cookie
7. Return new tokens

Logout Flow

1. Extract refresh token from cookie or request body
2. Revoke token by setting RevokedOn property
3. Clear the refresh token cookie
4. Return success response

API Endpoints

Authentication Endpoints

- **POST /api/auth/register**: User registration
- **POST /api/auth/login**: User login
- **POST /api/auth/refresh**: Token refresh
- **POST /api/auth/logout**: User logout

Password Management Endpoints

- **POST /api/auth/forgot-password**: Initiate password reset
- **POST /api/auth/reset-password**: Complete password reset
- **POST /api/auth/change-password**: Change password for authenticated user

User Management Endpoints

- **GET /api/auth/profile**: Get current user profile
- **PUT /api/auth/profile**: Update user profile
- **GET /api/auth/roles**: Get available roles (admin only)
- **POST /api/auth/roles**: Assign role to user (admin only)

Security Implementation

JWT Configuration

- 30-day token lifetime (43200 minutes)
- Secure signing key
- Issuer and audience validation
- Standard JWT claims (sub, name, email, role)

Cookie Management

- HTTP-only cookies for refresh tokens
- Secure flag for HTTPS-only transmission
- SameSite=None for cross-site requests
- IsEssential=True for cookie policy compliance
- Expiration matched to refresh token lifetime

Password Policy

- Minimum length requirement
- Complexity requirements (uppercase, lowercase, numbers, symbols)

- Password hashing via ASP.NET Identity
- Account lockout for failed attempts

Integration Points

Database Layer

- User entity storage and retrieval
- Refresh token management
- Role definitions and assignments
- Transaction management via UnitOfWork

Frontend Application

- Authentication flow integration
- Token management
- User profile display and editing

Other Microservices

- JWT validation via shared configuration
- User identity propagation
- Role-based access control

Configuration

JWT Settings

```
"JwtSettings": {
  "SecretKey": "your-secret-key-here",
  "Issuer": "OllamaNetAuth",
  "Audience": "OllamaNetClients",
  "ExpirationMinutes": 43200
}
```

Data Seeding

```
"DataSeeding": {
  "AdminUser": {
    "UserName": "admin",
    "Email": "admin@example.com",
    "Password": "Admin123!"
  },
  "Roles": ["Admin", "User"]
}
```

Cookie Settings

```
"CookieSettings": {
  "HttpOnly": true,
  "Secure": true,
  "SameSite": "None",
  "IsEssential": true
}
```

Error Handling

Authentication Errors

- Invalid credentials: 401 Unauthorized
- Account locked: 401 Unauthorized with specific message
- Registration validation failures: 400 Bad Request with details
- Token validation failures: 401 Unauthorized
- Expired tokens: 401 Unauthorized with specific message
- Invalid refresh tokens: 401 Unauthorized with specific message

Authorization Errors

- Insufficient permissions: 403 Forbidden
- Missing token: 401 Unauthorized
- Invalid role claims: 403 Forbidden

Known Issues

- Refresh token rotation not fully implemented for enhanced security
- Limited audit logging for security-sensitive operations
- Rate limiting for login attempts not fully implemented

Performance Considerations

- JWT validation overhead on high-traffic systems
- Database access for refresh token validation
- User manager operations performance
- Token generation cryptographic operations