Overview

This document provides a comprehensive technical overview of an ASP.NET Core-based e-commerce application. The application implements a RESTful API with JWT authentication, role-based authorization, and standard e-commerce functionality including user management, product catalog, shopping cart, and order processing.

Technology Stack

- **Framework:** ASP.NET Core Web API
- **Authentication:** JWT (JSON Web Tokens)
- **Authorization:** ASP.NET Core Identity with role-based access control
- **ORM:** Entity Framework Core
- **Database:** SQL Server

Design Patterns Implemented

- **Repository Pattern:** Implied through service layer abstraction
- **Service Layer Pattern:** Business logic encapsulated in dedicated service classes
- **DTO Pattern:** Data Transfer Objects for API serialization
- **Dependency Injection:** Constructor-based DI throughout controllers

API Controllers

1. AuthController (/api/Auth)

Handles user authentication and registration.

Endpoints

```
POST /register
```

- Registers new users with role assignment ("User" by default)
- Validates model state and city existence
- Returns registration success or validation errors

`POST /Login`

- Authenticates users via username/password
- Generates JWT with claims (NameIdentifier, Name, Jti, Role)
- Token expires after 1 hour
- Returns JWT token, expiration time, and success message

> **Security Considerations**
>
> - Passwords validated through Identity's built-in validators
> - Generic error messages prevent username enumeration
> - JWT configured with symmetric key signing (HMAC-SHA256)
> - Token includes audience and issuer validation

2. UserController (/api/User)

Manages user profiles, addresses, and role assignments.

Key Endpoints

`GET /Me [Authorize]`

Returns current authenticated user's profile extracted from JWT claims.

`GET /View-Profile/{userName} [Admin]`

Admin endpoint for viewing any user profile with double authorization check.

`GET /GetAllPaginated [Admin]`

Paginated user listing (10 users per page) with performance optimization.

> **Critical:** Always orders by JoinDate to ensure consistent pagination across requests.

`PUT /Update-Profile [Authorize]`

Updates current user's profile with custom validation context.

`POST /Change-Password [Authorize]`

Requires old password verification, leverages Identity's password change mechanism.

Address Management

- `POST /Add-Address/{address}` - Add new address
- `DELETE /Remove-Address/{addressId}` - Remove address
- `PUT /Update-Address/` - Update existing address
- `GET /Get-Addresses` - List all user addresses

Role Management (Admin Only)

- `POST /Assign-Role` - Assign role to user
- `PUT /Deassign-Role` - Remove role from user

> **Performance Notes**
>
> **GetAllPaginated():** Recommended approach combining `.AsNoTracking()`, pagination, and explicit ordering for scalability with large user bases.

3. ProductController (/api/Product)

Handles product catalog, categories, tags, and reviews.

Product Management (Admin Only)

`POST /Add [Admin]`

Adds products with category and tag associations, validates ProductDTO model state.

`DELETE /Remove/{productId} [Admin]`

Removes product from catalog.

Category & Tag Management (Admin Only)

- `POST /Add-Category/{categoryName}` - Create new category
- `DELETE /Remove-Category/{categoryName}` - Delete category

- `POST /Add-Tag/{tagName}` - Create new tag
- `DELETE /Remove-Tag/{tagName}` - Delete tag

Review System (Authenticated Users)

> `POST /Add-Review/{productId} [Authorize]`

Users can add reviews with ratings associated with their user ID.

> `DELETE /Remove-Review/{reviewId} [Authorize]`

Users can only delete their own reviews. Returns 403 Forbidden if unauthorized.

> `PUT /Edit-Review/{reviewId} [Authorize]`

Edit own reviews only, validates ownership before update.

> `DELETE /Remove-User-Review/{reviewId} [Admin]`

Admins can remove any review regardless of ownership.

Product Browsing (Authenticated Users)

> `GET /View/{productId} [Authorize]`

Returns product details with all reviews and calculated average rating.

> `GET /Filter [Authorize]`

Multi-dimensional filtering with pagination support:

- Category name
- Price range (min/max)
- Sorting: price (1) or reviews (2)
- Sort order: ascending/descending

> `GET /Search [Authorize]`

Basic product search by query string with paginated results.

> `GET /Filtered-Search [Authorize]`

Combines search with filtering capabilities using in-memory LINQ filtering.

4. CartController (/api/Cart)

Shopping cart management for authenticated users.

Endpoints

`POST /Add/{productId} [Authorize]`

Adds product to cart with quantity (default: 1, minimum: 1).

`PUT /Update-Quantity/{productId} [Authorize]`

Increment/decrement quantity via boolean flag (add=true: increment, add=false: decrement).

`DELETE /Remove-Product/{productId} [Authorize]`

Removes specific product from cart.

`DELETE /Clear [Authorize]`

Empties entire cart.

`GET /View [Authorize]`

Returns cart contents with calculated totals as ViewCartDTO.

Error Handling

Exception Type	HTTP Status	Description
KeyNotFoundException	404	Product or cart not found
InvalidOperationException	400	Business rule violations

NotAddedOrUpdatedException	400	Database operation failures
Generic Exception	500	Internal server error

5. OrderController (/api/Order)

Order processing and management.

User Endpoints (Authenticated)

`POST /Place/{addressId} [Authorize]`

Places order from current cart, requires delivery address, validates address ownership.

`GET /Past-Orders [Authorize]`

Lists user's order history as ViewOrderDTO collection.

`GET /View-Order/{orderId} [Authorize]`

Retrieves specific order details scoped to authenticated user.

`PUT /Cancel/{orderId} [Authorize]`

Cancels order (subject to status restrictions).

Admin Endpoints

`PUT /Update-Status/{orderId},{status} [Admin]`

Updates order status (0-4 range).

> **Status Progression**
>
> Pending → Processing → Delivered → Cancelled → Returned

`GET /Status-Sort [Admin]`

Returns orders sorted by status with pagination.

`GET /Date-Sort [Admin]`

Returns orders sorted by date with pagination.

6. RoleController (/api/Role)

Role administration (Admin only).

Endpoints

`POST /add-role/{RoleName} [Admin]`

Creates new role in Identity system.

`DELETE /remove-role/{RoleName} [Admin]`

Deletes existing role with existence validation.

`GET /get-roles [Admin]`

Lists all available roles in the system.

Security Implementation

JWT Authentication

Configuration

- **Algorithm:** HMAC-SHA256
- **Expiration:** 1 hour
- **Claims:** NameIdentifier, Name, Jti (unique token ID), Role(s)

Authorization Patterns

Pattern	Description
`[Authorize]`	Requires authentication only
`[Authorize(Roles = "Admin")]`	Requires Admin role
`[Authorize(Roles = "User")]`	Requires User role