Requirements:

1. **Project Setup:**
    a. Create a new ASP.NET Core (version 6 / 7) Web API project.
    b. Configure the project to use the latest version of Entity Framework Core.
2. **Web API Controllers:**
    a. Create a set of Web API controllers to handle CRUD operations for at least two entities (as mentioned below) in your application.
    b. Implement appropriate HTTP methods (GET, POST, PUT, DELETE) for each controller.
    c. Ensure that the controllers return appropriate HTTP status codes and responses.
3. **Custom Controller Base Class:**
    a. Create a custom base class for your Web API controllers that includes common functionality or behavior shared across multiple controllers.
    b. Implement exception handling and logging logic in the base class to handle common errors and provide meaningful error responses.
4. **ProblemDetails Class:**
    a. Utilize the ProblemDetails class from the Microsoft.AspNetCore.Mvc namespace to handle and return structured error responses.
    b. Implement custom error responses for various scenarios such as validation errors, resource not found, and server-side errors.
5. **Entity Framework Core Integration:**
    a. Set up the database context and configure Entity Framework Core to work with your chosen database provider (e.g., SQL Server, SQLite, etc.).
    b. Create the necessary entity classes to represent your application's data model.
    c. Implement migrations and seed the database with initial data using Entity Framework Core. d. Use Entity Framework Core to perform database operations (CRUD) within your Web API controllers.
6. **Testing and Documentation:**
    a. Write unit tests to validate the functionality of your Web API controllers and any other critical components.
    b. Document the API endpoints, request/response formats, and any additional notes that might be useful for developers who will be consuming your API.
7. **Best Practices and Security:**
    a. Apply best practices for secure API development, such as validating input data and handling sensitive information securely.
    b. Ensure that your API is well-structured, follows RESTful principles, and uses appropriate naming conventions.

Deliverables:

1. Complete source code of your ASP.NET Core Web API application.
2. Documentation describing the API endpoints, request/response formats, and any additional notes.
3. Unit tests for critical components.
4. Deployment instructions or scripts.

Tables/Models:

1. Create an entity model called "**Order**" with the following properties:
    - OrderId: A unique identifier for the order. (Primary Key)
    - OrderNumber: A system-generated order number for easy identification.
        - Auto-Generation Rule: The OrderNumber should be auto-generated using a sequential number or any desired pattern. e.g: Order_2024_1, Order_2024_2, Order_2024_3 etc. The year should be automatically generated as current year.
    - CustomerName: The name of the customer who placed the order.
        - Validation Rule: Required field, maximum length of 50 characters.
    - OrderDate: The date when the order was placed.
        - Validation Rule: Required field.
    - TotalAmount: The total amount of the order.
        - Validation Rule: Must be a positive number.
2. Create an entity model called "**OrderItem**" with the following properties:
    - OrderItemId: A unique identifier for the order item. (Primary Key)
    - OrderId: The identifier of the order to which the item belongs. (Foreign Key to Order table)
        - Validation Rule: Required field.
    - ProductName: The name of the product in the order item.
        - Validation Rule: Required field, maximum length of 50 characters.
    - Quantity: The quantity of the product in the order item.
        - Validation Rule: Must be a positive number.
    - UnitPrice: The unit price of the product in the order item.
        - Validation Rule: Must be a positive number.
    - TotalPrice: The total price of the order item (Quantity * UnitPrice).
        - Auto-Generation Rule: The TotalPrice should be calculated automatically based on the Quantity and UnitPrice.

In this example, the Order table represents customer orders, and the OrderItem table represents the individual items within each order. Each OrderItem has a foreign key (OrderId) that references the Order it belongs to.

API Endpoint Requirement Specification:

1. Retrieve all orders
    - HTTP Method: GET
    - Endpoint: /api/orders
    - Response:
        - Status Code: 200 OK
        - Body: List of OrderResponse objects representing all orders
    - Description: Retrieves a list of all orders.
2. Retrieve an order by ID
    - HTTP Method: GET
    - Endpoint: /api/orders/{id}
    - Parameters:
        - id: The ID of the order to retrieve
    - Response:
        - Status Code: 200 OK
        - Body: OrderResponse object representing the retrieved order
        - Status Code: 404 Not Found (if order with the given ID does not exist)
    - Description: Retrieves an order by its ID.
3. Add a new order
    - HTTP Method: POST
    - Endpoint: /api/orders
    - Request Body: OrderAddRequest object containing the order details
    - Response:
        - Status Code: 201 Created
        - Body: OrderResponse object representing the added order
    - Description: Adds a new order.
4. Update an existing order
    - HTTP Method: PUT
    - Endpoint: /api/orders/{id}
    - Parameters:

        - id: The ID of the order to update
    - Request Body: OrderUpdateRequest object containing the updated order details
    - Response:
        - Status Code: 200 OK
        - Body: OrderResponse object representing the updated order
        - Status Code: 400 Bad Request (if the ID in the request body does not match the route parameter)
    - Description: Updates an existing order.
5. Delete an order
    - HTTP Method: DELETE
    - Endpoint: /api/orders/{id}
    - Parameters:
        - id: The ID of the order to delete
    - Response:
        - Status Code: 204 No Content (if deletion is successful)
        - Status Code: 404 Not Found (if order with the given ID does not exist)
    - Description: Deletes an order by its ID.
6. Retrieve all order items for a specific order
    - HTTP Method: GET
    - Endpoint: /api/orders/{orderId}/items
    - Parameters:
        - orderId: The ID of the order
    - Response:
        - Status Code: 200 OK
        - Body: List of OrderItemResponse objects representing all order items for the specified order
    - Description: Retrieves all order items for a specific order.
7. Retrieve an order item by ID
    - HTTP Method: GET
    - Endpoint: /api/orders/{orderId}/items/{id}
    - Parameters:
        - orderId: The ID of the order
        - id: The ID of the order item
    - Response:
        - Status Code: 200 OK
        - Body: OrderItemResponse object representing the retrieved order item

        - Status Code: 404 Not Found (if order item with the given ID does not exist)
    - Description: Retrieves an order item by its ID.
8. Add a new order item
    - HTTP Method: POST
    - Endpoint: /api/orders/{orderId}/items
    - Parameters:
        - orderId: The ID of the order
    - Request Body: OrderItemAddRequest object containing the order item details
    - Response:
        - Status Code: 201 Created
        - Body: OrderItemResponse object representing the added order item
    - Description: Adds a new order item to the specified order.
9. Update an existing order item
    - HTTP Method: PUT
    - Endpoint: /api/orders/{orderId}/items/{id}
    - Parameters:
        - orderId: The ID of the order
        - id: The ID of the order item
    - Request Body: OrderItemUpdateRequest object containing the updated order item details
    - Response:
        - Status Code: 200 OK
        - Body: OrderItemResponse object representing the updated order item
        - Status Code: 400 Bad Request (if the ID in the request body does not match the route parameter)
    - Description: Updates an existing order item.
10. Delete an order item
    - HTTP Method: DELETE
    - Endpoint: /api/orders/{orderId}/items/{id}
    - Parameters:
        - orderId: The ID of the order
        - id: The ID of the order item to delete
    - Response:
        - Status Code: 204 No Content (if deletion is successful)
        - Status Code: 404 Not Found (if order item with the given ID does not exist)
    - Description: Deletes an order item.

Sample Output:

1. Retrieve all orders (GET `/api/orders`)

   Status Code: 200 OK

   Response Body:

```
[
  {
    "orderId": "f4816224-70d6-4491-ac52-34f298ace16f",
    "orderNumber": "ORD001",
    "customerName": "John Doe",
    "orderDate": "2023-07-14T10:30:00Z",
    "totalAmount": 66.5
  },
  {
    "orderId": "735886c0-faf3-49ca-9776-8a20b756f1cb",
    "orderNumber": "ORD002",
    "customerName": "Jane Smith",
    "orderDate": "2023-07-14T11:45:00Z",
    "totalAmount": 225.8
  }
]
```

2. Retrieve an order by ID (GET `/api/orders/{id}`)

   Status Code: 200 OK

   Response Body:

```
{
  "orderId": "f4816224-70d6-4491-ac52-34f298ace16f",
  "orderNumber": "ORD001",
  "customerName": "John Doe",
  "orderDate": "2023-07-14T10:30:00Z",
  "totalAmount": 66.5
}
```

3. Add a new order (POST `/api/orders`)

Status Code: 201 Created

Response Body:

```
{
  "orderNumber": "ORD003",
  "customerName": "Alice Johnson",
  "orderDate": "2023-07-14T12:00:00Z",
  "totalAmount": 120.0
}
```

4. Update an existing order (PUT `/api/orders/{id}`)

   Status Code: 200 OK

   Response Body:

```
{
  "orderId": "f4816224-70d6-4491-ac52-34f298ace16f",
  "orderNumber": "ORD001",
  "customerName": "John Doe",
  "orderDate": "2023-07-14T10:30:00Z",
  "totalAmount": 75.0
}
```

5. Delete an order (DELETE `/api/orders/{id}`)

   Status Code: 204 No Content

6. Retrieve all order items for a specific order (GET `/api/orders/{orderId}/items`)

   Status Code: 200 OK

   Response Body:

```
[
  {
    "orderItemId": "d20882df-7fca-4ee8-88bb-37d2fc75e63f",
    "orderId": "f4816224-70d6-4491-ac52-34f298ace16f",
    "productName": "Product A",
    "quantity": 2,
    "unitPrice": 10.0,
    "totalPrice": 20.0
  },
```

```
  {
    "orderItemId": "2e27b6a4-469d-4d7f-8b8b-54af129675fd",
    "orderId": "f4816224-70d6-4491-ac52-34f298ace16f",
    "productName": "Product B",
    "quantity": 3,
    "unitPrice": 15.5,
    "totalPrice": 46.5
  }
]
```

7. Retrieve an order item by ID (GET `/api/orders/{orderId}/items/{id}`)

   Status Code: 200 OK

   Body:

```
{
  "orderItemId": "d20882df-7fca-4ee8-88bb-37d2fc75e63f",
  "orderId": "f4816224-70d6-4491-ac52-34f298ace16f",
  "productName": "Product A",
  "quantity": 2,
  "unitPrice": 10.0,
  "totalPrice": 20.0
}
```

8. Add a new order item (POST `/api/orders/{orderId}/items`)

   Status Code: 201 Created

   Body:

```
{
  "orderItemId": "1a23b156-79fd-4e9a-9e07-3d42374a78f5",
  "orderId": "f4816224-70d6-4491-ac52-34f298ace16f",
  "productName": "Product C",
  "quantity": 4,
  "unitPrice": 8.0,
  "totalPrice": 32.0
}
```

9. Update an existing order item (PUT `/api/orders/{orderId}/items/{id}`)

   Status Code: 200 OK

Body:

```
{
  "orderItemId": "d20882df-7fca-4ee8-88bb-37d2fc75e63f",
  "orderId": "f4816224-70d6-4491-ac52-34f298ace16f",
  "productName": "Product A",
  "quantity": 5,
  "unitPrice": 10.0,
  "totalPrice": 50.0
}
```

10. Delete an order item (DELETE `/api/orders/{orderId}/items/{id}`)

    Status Code: 204 No Content

Note: You are encouraged to extend the requirements and add any additional features or enhancements that you believe will demonstrate your proficiency in ASP.NET Core Web API development.