I implemented the project as outlined in the email, introducing several improvements and modifications. Here's an overview:

Models Folder:

1. Item:

- Added an 'Id' property for each 'Item,' automatically calculated as (max(id) + 1).
- The 'Id' is used to identify the type of CRUD operation; a null value indicates a 'Create' request, while a non-null value indicates an 'Update' request.
- 'Title' must be non-null and unique.
- 'Price' should be a positive decimal value.

2. ShoppingCart:

- Includes an 'Id' referring to the 'Id' of an item.
- Introduces a 'Quantity' property with a positive integer value.

3. ShoppingCartDTO:

- Retrieves the contents of a ShoppingCart.
- 'Id' corresponds to the 'Id' of an 'Item.'
- 'RowNo' is an autogenerated integer representing the order of 'Title,' 'Price,' and 'Quantity.'
- Includes 'Title,' related to the 'Title' of an 'Item.'
- Calculates 'TotalPrice' as 'Price' * 'Quantity.'
- Ends with a 'Total Row' having 'Id' = -1, 'RowNo' = -1, 'Title' = 'Total', 'Price’ = 0, 'Quantity’ = Sum of the quantities of ShoppingCartItems, and 'TotalPrice' = Sum of the Total Prices.

4. CustomValidationException:

- Subclass of `Exception`.
- Includes a property to retrieve error messages as a list of strings, facilitating simultaneous propagation.
- Provides two constructors: one with a simple string and the other with a list of error messages.

Managers Folder:

1. ItemManager:

- Implements CRUD operations with a single function named 'SaveItem.'

- Uses 'CanSaveItem' for validation, checking constraints such as non-null or duplicated 'Title,' non-positive 'Price,' and the existence of 'Item.'
- 'RetrieveItem' methods for retrieving items by 'Id' or 'Title.'
- 'DeleteItem' methods for deleting items by 'Id' or 'Title.'
- Includes 'CanDelete' functions for checking whether an item can be deleted. An item cannot be deleted if it is not used in shopping cart.

2. ShoppingCartManager:

- Manages 'ShoppingCart' operations.
- Validates the selected id's presence in the list of 'Items' and checks for non-positive 'Quantity.'
- Implements a method for joining 'ShoppingCart' and 'Items,' sorting the result by 'Title,' 'Price,' and 'Quantity.'
- Adds a 'TotalRow' at the end of the list, summarizing quantities and total prices.

Controllers Folder:

1. ItemsController:

- Implements controllers for 'Item' operations.
- Includes methods for retrieving items by 'Id' or 'Title', deleting items by 'Id' or 'Title,' and adding/updating items with 'CreatedAtAction' returning the route to 'RetrieveItem.'
- Implements 'Update' which was not requested in the mail.

2. ShoppingCartController:

- Straightforward implementation for 'ShoppingCart' operations.

Middleware Folder:

ExceptionMiddleware:

- Handles all exceptions in the application.
- Logs exceptions using 'Serilog'.
- Returns 'BadRequest' for 'CustomValidationException' and 'InternalServerError' for other exceptions.
- Logs are preserved in the 'Logs' folder.

Utils Folder:

- Implements the Singleton pattern for managers and the list of items and shopping cart contents.
- Utilizes Swashbuckle for Swagger documentation.

Testing:

- Utilizes NUnit tests with an added item group (InternalsVisibleTo) in the properties to make the program accessible to NUnit tests.