Product Management System - CLI Application

Problem Statement

Your task is to create a command-line interface (CLI) based application in C# that provides complete CRUD (Create, Read, Update, Delete) operations and search functionality for managing a product inventory. The application will store product information in memory during runtime, with an option to save data to and load data from a file for persistence.

Application Requirements

Core Features

1. **Product Model**
    - Implement a `Product` class with the following properties:
        - `ProductId` (unique identifier, integer)
        - `Name` (string, 3-50 characters)
        - `Description` (string, up to 500 characters)
        - `Category` (enum with values: Electronics, Clothing, Food, Books, Other)
        - `Price` (decimal, positive value)
        - `QuantityInStock` (integer, non-negative)
        - `DateAdded` (DateTime)
        - `IsDiscontinued` (boolean)
2. **Main Menu**
    - Display a menu with numbered options for all operations
    - Include clear instructions for user interaction
    - Implement proper input validation for menu selection
3. **CRUD Operations**
    - **Create**: Add new products to the inventory
    - **Read**: View product details (individual or all products)
    - **Update**: Modify existing product information
    - **Delete**: Remove products from the inventory
4. **Search Functionality**
    - Search by ID (exact match)
    - Search by name (partial match, case-insensitive)
    - Search by category
    - Search by price range (minimum and maximum)
    - Show products with low stock (below a specified threshold)
5. **Data Persistence**
    - Save product data to a JSON file

- Load product data from a JSON file
- Auto-save option when exiting the application

6. **Error Handling**
    - Comprehensive user input validation
    - Proper exception handling with user-friendly error messages
    - Confirmation requests for critical operations (update/delete)

Implementation Details

Menu Structure

```
========================================
    PRODUCT MANAGEMENT SYSTEM
========================================
1. View All Products
2. View Product Details
3. Add New Product
4. Update Product
5. Delete Product
6. Search Products
7. Save Data to File
8. Load Data from File
9. Exit
----------------------------------------
Enter your choice (1-9):
```

Search Sub-Menu

```
========================================
    SEARCH PRODUCTS
========================================
1. Search by ID
2. Search by Name
3. Search by Category
4. Search by Price Range
5. Show Low Stock Products
6. Return to Main Menu
----------------------------------------
Enter your choice (1-6):
```

Product Display Format

Products should be displayed in a tabular format with borders:

```
+--------+--------------------+------------+----------+--------+
| ID     | Name               | Category   | Price    | Stock  |
+--------+--------------------+------------+----------+--------+
| 1      | Laptop Pro         | Electronics| $1299.99 | 25     |
| 2      | Cotton T-Shirt     | Clothing   | $19.99   | 150    |
+--------+--------------------+------------+----------+--------+
```

For detailed view, display all properties including description and formatted date:

```
========================================
    PRODUCT DETAILS
========================================
ID: 1
Name: Laptop Pro
Description: 15.6" laptop with 16GB RAM, 512GB SSD
Category: Electronics
Price: $1299.99
Quantity in Stock: 25
Date Added: 01/15/2023
Discontinued: No
```

Input Validation Requirements

1. **ProductId**
    - Must be a positive integer
    - Must be unique in the system
2. **Name**
    - Required field (cannot be null or empty)
    - Length between 3 and 50 characters
    - Must be unique in the system
3. **Description**
    - Optional field
    - Maximum 500 characters if provided
4. **Category**
    - Must be one of the predefined enum values
    - Present options in a numbered menu format
5. **Price**
    - Must be a positive decimal value
    - Maximum two decimal places
    - Range between 0.01 and 1,000,000.00

6. **QuantityInStock**
    - Must be a non-negative integer
    - Maximum value 1,000,000

Error Handling Examples

1. **Invalid Menu Selection**

```
Invalid choice! Please enter a number between 1 and 9.
```

2. **Product Not Found**

```
Error: Product with ID 5 does not exist.
```

3. **Validation Error**

```
Error: Please correct the following issues:
- Name must be between 3 and 50 characters
- Price must be greater than zero
```

4. **File Operation Error**

```
Error: Unable to save data to file 'products.json'.
Details: Access to the path is denied.
```

Confirmation Prompts

1. **Delete Confirmation**

```
Are you sure you want to delete "Laptop Pro" (ID: 1)? (Y/N):
```

2. **Update Confirmation**

```
You have modified the following fields: Name, Price, Stock
Save these changes to product "Laptop Pro" (ID: 1)? (Y/N):
```

3. **Exit Confirmation**

```
Data has been modified since last save.
Do you want to save before exiting? (Y/N):
```

Implementation Requirements

1. **Architecture**
   - Implement proper separation of concerns (UI, business logic, data access)
   - Use appropriate design patterns (Repository pattern recommended)
   - Implement interfaces for service classes to enable future extensibility
2. **Code Quality**
   - Clear and consistent naming conventions
   - Proper commenting and documentation
   - Input validation and error handling
   - Efficient data structures for storage and retrieval
3. **Optional Advanced Features** (for extra credit)
   - Implement sorting options (by name, price, stock, etc.)
   - Add product image path/URL handling
   - Include a simple reporting feature (e.g., total inventory value)
   - Command history and undo functionality

Sample Implementation Flow

1. **Add Product Flow**

```
Enter product name: Laptop Pro
Enter description (optional): 15.6" laptop with 16GB RAM, 512GB SSD
Select category:
1. Electronics
2. Clothing
3. Food
4. Books
5. Other
Enter choice (1-5): 1
Enter price: 1299.99
Enter quantity in stock: 25
Is this product discontinued? (Y/N): N

Product added successfully!
ID: 1 | Name: Laptop Pro | Category: Electronics | Price: $1299.99 | Stock:
25
```

2. **Update Product Flow**

```
Enter product ID to update: 1

Current values:
Name: Laptop Pro
Description: 15.6" laptop with 16GB RAM, 512GB SSD
Category: Electronics
Price: $1299.99
Quantity in stock: 25
Discontinued: No

Enter new name (or press Enter to keep current): Laptop Pro Max
Enter new description (or press Enter to keep current):
Select new category (or press Enter to keep current):
1. Electronics
2. Clothing
3. Food
4. Books
5. Other
Enter choice (1-5 or Enter):
Enter new price (or press Enter to keep current): 1499.99
Enter new quantity (or press Enter to keep current): 20
Is this product discontinued? (Y/N or Enter to keep current):

You have modified the following fields: Name, Price, Stock
Save these changes to product "Laptop Pro" (ID: 1)? (Y/N): Y

Product updated successfully!
```

3. **Search by Price Range Flow**

```
Enter minimum price (or press Enter for no minimum): 500
Enter maximum price (or press Enter for no maximum): 2000

Found 3 products in the price range $500.00 - $2000.00:
+--------+--------------------+------------+----------+--------+
| ID     | Name               | Category   | Price    | Stock  |
+--------+--------------------+------------+----------+--------+
| 1      | Laptop Pro Max     | Electronics| $1499.99 | 20     |
| 3      | Gaming Console     | Electronics| $499.99  | 12     |
| 8      | Designer Jacket    | Clothing   | $599.99  | 5      |
+--------+--------------------+------------+----------+--------+
```

Exit Criteria & Grading Rubric

Functionality (50%)

- Complete implementation of all CRUD operations
- Working search functionality with all specified criteria
- Proper input validation for all user inputs

- Functional file saving and loading capability
- Clean, usable menu navigation system

Code Quality (30%)

- Proper separation of concerns (UI, business logic, data layer)
- Consistent error handling throughout the application
- Clean, well-documented code with appropriate comments
- Effective use of C# language features and data structures
- No code duplications or "magic numbers"

User Experience (20%)

- Clear menu navigation and instructions
- Informative error messages
- Confirmation prompts for critical operations
- Consistent formatting of output data
- Graceful handling of edge cases

Extra Credit

- Implementation of advanced features
- Unit tests for business logic
- Enhanced user interface features
- Performance optimizations for large datasets

Submission Requirements

1. Source code files (.cs)
2. Documentation explaining your design decisions
3. Sample data file (JSON) with at least 10 test products
4. Brief user guide for operating the application

Your application will be tested with various scenarios including invalid inputs, edge cases, and high-volume operations to verify robustness and performance.

Important Notes

- Do not code everything in Program.cs
    - In fact, you should have **only a minimal code**, just to start the app
- **Create proper namespaces for different types of classes (models, service, utilities etc)**
- The application should never crash under normal operation
- All user interactions should have appropriate feedback
- The interface should be intuitive with clear prompts
- Pay attention to the details of validation, error handling, and confirmation messages
- Focus on providing a smooth and consistent user experience

Example code to save list of products as JSON in a file:

```
using System.Text.Json;

// Serialize to JSON
string jsonString = JsonSerializer.Serialize(products, new JsonSerializerOptions
{
   WriteIndented = true // For pretty printing
});

// Write to file
File.WriteAllText("products.json", jsonString);
```