**.NET Core Web API Order Management System Assignment**

**Deadline:** [**29-02-2024**]

**Instructions:**

1. This assignment is aimed at testing your ability to design and implement a .NET Core Web API for managing orders in a system.
2. Follow RESTful principles and best practices throughout your implementation.
3. Provide clear documentation and comments in your code to explain the functionality and design decisions.
4. Utilize Swagger for API documentation and Postman for testing.
5. Develop and implement a UI using .NET Core MVC to consume the Order Management API.
6. Submit your solution as a zip file containing all necessary project files, including source code, unit tests, documentation, Postman collection, UI project files, and any required configuration files.

**Assignment Tasks:**

* **Task 1: Setup Project**

1. Create a new .NET Core Web API project named "OrderManagementAPI".
2. Create a new .NET Core MVC project named "OrderManagementUI".
3. Configure the projects to use your preferred ports.
4. Integrate Swagger for API documentation.

* **Task 2: Define Models**
* Create the following models:
* Order: Represents an order placed by a customer with properties like OrderId, CustomerId, OrderDate, TotalAmount, and Status.
* OrderItem: Represents an item within an order with properties including OrderItemId, OrderId, ProductId, Quantity, and UnitPrice.
* Product: Represents a product available for purchase with properties such as ProductId, Name, Description, Price, and StockQuantity.
* Customer: Represents a customer with properties like CustomerId, Name, Email, and Address.
* **Task 3: Implement CRUD Operations**
* Design and implement CRUD operations for managing orders, order items, products, and customers.
* Include endpoints for:
* Orders: GET, POST, PUT, DELETE
* OrderItems: GET, POST, PUT, DELETE
* Products: GET, POST, PUT, DELETE
* Customers: GET, POST, PUT, DELETE
* Implement appropriate validation to ensure data integrity.
* **Task 4: Implement Order Processing**
* Implement functionality to process orders:
* Allow customers to place new orders by selecting products and quantities.
* Update product stock quantities upon order placement.
* Calculate the total order amount based on selected products and quantities.
* Change order status (e.g., Pending, Shipped, Delivered) as orders are processed.
* **Task 5: Implement Search and Filtering**
* Implement search and filtering capabilities for orders and products:
* Allow users to search for orders by customer name, order status, or date range.
* Enable filtering products by name, price range, or category.
* **Task 6: Implement Reporting**
* Design and implement reporting functionalities:
* Generate sales reports to show total sales revenue over a specified period.
* Generate inventory reports to display the current stock status of products.
* **Task 7: Develop UI using .NET Core MVC**
* Create a new .NET Core MVC project named "OrderManagementUI".
* Implement views and controllers to consume the Order Management API.
* Create views for listing orders, adding/editing orders, managing products, and managing customers.
* Implement client-side validation for UI forms.
* Consume the Order Management API in the MVC project.
* **Task 8: Documentation and Deployment**
* Provide comprehensive documentation for your API using Swagger, including endpoint descriptions, request/response formats, and usage examples.
* Export your Postman collection containing test requests for each API endpoint.
* Deploy your API and provide deployment instructions.
* Include screenshots of your deployed UI and API documentation in your submission.

**Submission:**

1. Submit your solution as a zip file containing all project files, including source code, documentation (including Swagger documentation), Postman collection, UI project files, deployment instructions, and any required configuration files.
2. Ensure that your solution is well-organized, thoroughly tested, and meets all the requirements specified in the assignment tasks.

This comprehensive assignment now includes developing and implementing a UI using .NET Core MVC to consume the Order Management API, Students are required to demonstrate their skills in both backend API development and frontend UI development.