# *Web Programming VI (420-H60-HR)*

# *Assignment 3 – Shopping Cart*

Date Assigned: Tuesday, October 30, 2024

Due Date: **Thursday, December 6, 11:50PM**

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

* Set Up and Use ReST Web Services
* Apply TDD
* Use SOAP Web Services
* Provide JavaScript access to .NET Web Services via SPA
* Use Docker and Docker compose to deploy microservices applications

**Setup**

Create a copy of your assignment 2 folder and call it *initials*H60A03. Do not change the name of your solution; leave it as *initials*H60Store. If, after reading over the assignment, you intend to change the database structure, make a copy of your database as per the instructions from the previous assignment. Call the new database H60Assignment3DB\_*yourinitials.*

Make sure that your assignment still works.

If you did not do so for assignment 2, create a non-.NET project as well in assignment 3. Create a folder called ***yourinitials*H60Manager**. This must be at the same level as your solution (NOT THE PROJECTS, THE SOLUTION). This will be JavaScript (with HTML and CSS). I recommend using the React framework, but the requirement is that it is a JS-driven, client-side application that interacts directly with the ReST services.

Part A: Services Update – Shopping Cart

NB: For this Part A, write tests first. Before implementing any functionality, write unit tests for each CRUD operation. For each operation, provide screenshots showing :

* Sreenshot for initial unit test code  
  (named it CrudOperation\_initialUnitTest e.g. Create\_ initialUnitTest)
* Screenshot for test failing (named it CrudOperation\_TestFailing e.g Read\_TestFailing)
* Screenshot for enhanced code (named it CrudOperation\_EnhancedCode e.g. Update\_EnhancedCode)
* Screenshot for test passing (named it CrudOperation\_TestPassing e.g. Delete\_TestPassing)

Put your screenshots in the root folder of your solution folder inside a subfolder called “ServicesScreenshots”. You should create different subfolder for each operation in “ServicesScreenshots” folder.

1. Prior to any coding, make sure you understand the functionality required.
2. You are to add REST functions to the Services project to allow the following functionality
3. Perform CRUD operations on the ShoppingCart
4. Perform CRUD operations on products to be added and product removed from the cart (CartItem table). Remember, the stock on the product in the table must be adjusted when an item is added/removed from the cart or the quantity of the item is changed; you created the operation to do that in the previous assignment.
5. Remember, deletes cannot be done if a cart has products in it (you must maintain referential integrity).

Part B: Customer project – Shopping Cart

1. In the Customer Project, provide an interface to perform the following operations:
2. Update your ReST calls for getting the product information to use a DTO to only get the required fields. (This may already be done in Assignment 2).
3. Add and remove products to the current (customer’s) shopping cart (CartItem table) allowing user to enter a quantity and taking the current product price. (Perform validation to ensure that there is sufficient stock of a product before adding it). Remember stock values must be updated in the product table. Remember also that I will be doing concurrency testing. Be robust.

Part C: Complete an Order

1. Add REST functions to the Services project to allow the following functionality. Perform CRU operations on the Order (no need for deletes this time)
   1. Perform R operation on the Order to select by date fulfilled (all the orders for given date)
   2. Perform R operation on the Order to select by customer (all the orders for a customer)
   3. Perform CRU operations on OrderItem table. You should not have to worry about the stock as it is taken care of in the Shopping Cart.
   4. Call the provided SOAP service to validate the customer’s credit card (see below).
   5. Call the provided SOAP service to calculate the taxes based on the customer’s province.
   6. Calculate the total and taxes and store them in the Order.
2. In the Customer project, provide an interface (combine it with the previous one) to perform the following operations:
3. Go to Check Out (validate that there is a cart).
4. Call the SOAP Service to validate the credit card (see below) (this should be done in the model)
5. Add the total amount to the order
6. Move the cart and items to the order and order items and delete the cart and the cart items (careful about integrity here).
7. Confirm the order with the customer before completing it.
8. Display a nice message to the client indicating that their order is being processed and will be delivered.

Part D: Management Access (SPA)

1. In the *yourinitials*H60Managerfolder create an HTML/JavaScript (with CSS for styling) application that does the following. Note, most of these REST services have been completed, but you may have to add a few more.
2. Login as a manager (you will need to add a ReST service for this). (To expedite, this is optional and now part of the final five).
3. Gets/searches by category, product description, how many stock are left, the buy and sell price.
4. Update the amount of stock remaining for a chosen item (yes, I know this is a duplicate)
5. Update the buy and sell price for a chosen item (yes, this is a duplicate too)
6. Get a report on all the orders for a given date and the total of those orders.
7. Get a report on all orders for a given customer and the total of those orders.

SOAP Services

A SOAP service has been provided for you.

That SOAP service, at csdev.cegep-heritage.qc.ca/cartService/checkCreditCard.asmx validates (completely artificially) the credit card passed. Validation is as follows: cards must be between 12 and 16 characters, all numbers, sum of each group of 4 (or last group <4) must be less than 30 and product of last 2 digits must be multiple of 2. If everything is okay, it still returns an error 10% of the time for no balance on card. Return codes are GoodCard = 0, ErrInvalidLength = -1, ErrNotAllNum = -2, ErrCheckSum = -3, ErrProduct = -4 and ErrBalance (random) = -5.

It is recommended that you run them and play with this service. You MUST use this service appropriately in this assignment.

Implementation notes:

1. Make it look nice. There are three interfaces here: the Store one to add and remove products, the Customer one to put products in a Cart and create Orders and a Management one to Iperform roll up functions.
2. Choose the names of your routes carefully and follow the standards discussed in class.
3. If you ever want to reset your database, just delete it and copy the one from the previous assignment again.
4. Marking is being done based on functionality completed, so you can pick and choose what functionality you implement and accept the lower mark if you want.
5. If you want to add different/extra functionality to the REST services so that things do not have to be repeated that is absolutely fine (like removing from shopping cart items when adding to order).

Part E: Deployment

1. Create Dockerfiles for each part of the application (e.g., Services, Customer project, Manager project). Use multi-stage builds to optimize Docker images
2. Create a docker-compose.yml file to define and manage multi-container applications. Include services for the database, web services, and front-end applications.
3. Use environment variables to configure services within Docker containers.
4. Deploy the application using Docker Compose and take screenshots of your terminal and browser showing that your deployment is working properly. Put your screenshots in the root folder of your solution folder

**Final Five**

Completing the above receives a maximum of 95%. The final 5% is available if you complete at least two of the following extra pieces of functionality.

1. Provide the ability (using web services) for customers to leave product reviews. These reviews can be seen by other customers (database change required, new tables required) when looking at the product.
2. Add ability for a customer to have a wishlist. This can be a second cart except that the items in the wishlist cart are not removed from stock. Items on the wishlist must be moved to the shopping cart before being added to an order.
3. Add account creation by the customer using a link from the login screen.
4. Provide Forgotten password support using the functionality from Identity from ALL login screens.
5. Provide appropriate graphs for the management interface for the reports such as trends and bar charts for sales (hint: there are cool JavaScript libraries for this).
6. Provide proper login and authentication of the manager for your SPA

**To submit**

When you have completed the assignment, submit it in your github classroom repository.

|  |  |  |
| --- | --- | --- |
| Milestone | Date |  |
| M1 | November 19, 23h50 | Part A |
| M2 | December 5th, 23h50 | Part B |
| Final | Saturday Dec 7, 23h50 | Part C, D, E |

**Marking**

|  |  |  |
| --- | --- | --- |
| **Web Programming VI (420-H60-HR) -  Assignment 3 (Shopping Cart)** |  | **Mark** |
|  |  | 0% |
|  |  |  |
|  | **Mark** | **Out Of** |
| **Milestones** |  |  |
| M1 – Part A |  | 5 |
| M2 – Part B |  | 10 |
|  |  |  |
|  |  |  |
| **Services Project** |  |  |
| Screenshots for CRUD operation on Shopping Cart (At least 04 screenshots for each operation) |  | 16 |
| Screenshots for CRUD operation on Cart item (At least 04 screenshots for each operation) |  | 16 |
| CRUD on Shopping Cart/Item - includes checking stock and updating - includes attempting delete with items |  | 8 |
| CRU on Order/OrderItem |  | 4 |
| Select Order by Date Filled |  | 3 |
| Select Order by Customer Name |  | 2 |
| SOAP for credit card |  | 4 |
| Total stored in Order |  | 3 |
| Move Cart to Order and Delete Cart |  | 3 |
| **Customer Project** |  |  |
| Add/remove Products to Shopping Cart |  | 6 |
| Complete Order |  | 6 |
| Confirm order and receive messages |  | 4 |
| **Management Access** |  |  |
| ~~Login~~ |  |  |
| Search products |  | 4 |
| Update Stock |  | 4 |
| Update prices |  | 4 |
| Report of all orders by day |  | 4 |
| Report all orders by customer |  | 4 |
| **Design** |  |  |
| Proper design standards |  | 6 |
| Code/Interface design/efficiency |  | 3 |
| Customer visible design/workflow |  | 6 |
|  |  |  |
| Handed in properly |  | 2 |
| **Deployment** |  |  |
| Create dockerfiles |  | 5 |
| Create docker-compose file |  | 3 |
| Use environment variable for configuration |  | 2 |
| Screenshot for Docker compose deployment |  | 5 |
| **Final 5** 1. Ability for customers to leave product reviews.  2. Customer wishlist.  3. Account creation by the customer  4. Forgotten password support  5.Appropriate graphs for management interface 6 Proper login and security for Manager project |  | 5 |
| **Total** | 0 | 147 |