# COMP 4601A
# Fall 2023 - Lab #2

## Objectives

The goal for this lab is to incorporate MongoDB into the basic e-commerce example from the first lab. You will also be expanding a bit on the existing RESTful system you created previously.

## Demonstrating/Submitting

There will be two ways to receive credit for completed labs, outlined below:
1. Attend an in-person lab or office hours and demonstrate your completed lab before the deadline. You will have to show that the goals of the lab have been completed and answer some questions about the lab and your code (see the lab reflection questions for some examples). Your grade will depend on the level of completion, as well as the quality of your design and answers. Only one partner is required for demonstration, though all partner's are encouraged to take part. **If you demonstrate your lab this way, you don't need to submit anything on Brightspace.**
2. Record a video demonstration that is <10 minutes long. Ensure that your discussion in the video makes it clear that you have understood the content that the lab covers and that you demonstrate all the required functionality. Submit a ZIP containing a copy of your code (don't include database files, etc.), your answers to the lab reflection questions, and a copy of your demonstration video (either link to a public URL in your README or include the video file directly) to Brightspace. **If you are working with a partner, only one of you should make a submission. Include the names of all group members in the README file.**

---

## Lab Description

To start this lab, convert your existing RESTful system from lab #1 so that it stores and retrieves the product data using a MongoDB database. Your system must support all the previous functionality that was implemented as part of lab #1. The functionality below must also work with the database data.

Create an additional handler in your server to process POST requests to the URL /orders. The role of this handler will be to accept new purchase orders, process those orders, and send a proper response to the client that initiated the request. You are free to decide on the final structure of your order resources, but at minimum an order must contain the name of the person making the purchase, a set of products being purchased, as well as the quantity of each product

being purchased. Upon receiving an order request, your server must check that the order is valid (all products in the order exist and enough quantity of each is in-stock to fulfill this order). If the order is valid, the stock quantities of each item in the order must be reduced by the number of units ordered and the client should receive a response status code of 201 (Created). A new order resource should be created on the server containing the information about this order. If the order is invalid, the server should send a response code of 409 (Conflict) and specify in the body of the response what part(s) of the order was invalid. You may add functionality for creating and placing orders to your browser-based front-end, or test your new functionality using a tool such as Postman (you do not need to support both for this lab).

Your server must also support GET requests for the URL /orders. The response for this request should contain a list of the orders stored on your server. The returned data must also include links for each single order resource. If the client requests an order (e.g., /orders/5 for order #5), they should be able to view the products that were purchased as part of that order.

To receive full grades for this lab, you must demonstrate:

1. All of your server's data is now stored in a database
2. The previous functionality from lab #1 still works
3. You can create orders successfully and the database is appropriately updated
4. Your server rejects orders that are not valid due to:
    a. Missing the purchaser's name
    b. Product does not exist
    c. Product does not have enough stock
5. You can view the list of orders that exist on the server
6. You can view specific orders and see products that were part of the order

## Lab Reflection Questions

1. Discuss the collections and documents that you store in the database (e.g., what data are you storing about products, orders, etc.).
2. What is the structure of your new order data? How is it sent to the server (e.g., through an HTML form, JSON request)?
3. How do you load the data for a specific order so it can be displayed to the user?