# eCommerce Training Specification – Backend

* Generic:
  + All errors should throw reasonable exceptions
  + You should not throw 500 level exceptions
* User and Roles (Advanced)
  + Add 2 new roles ROLE\_STAFF (STAFF) and ROLE\_MANAGER (MANAGER), where MANAGER should have higher permission than STAFF, and ADMIN should have higher permission than MANAGER, MANAGER and ADMIN should be 2 composite roles, basically, STAFF belongs to MANAGER and MANAGER belongs to ADMIN
* Staff (In Depth)
  + Create a new table through liquibase called “TblStaff”, it should connect to User and have authority ROLE\_STAFF as default value
* Permission Control (Advanced)
  + Everyone should be able to get product list
  + Only STAFF allows to update/delete product
  + Inventory can only be created/updated/deleted by MANAGER
  + Dimension can only be created/updated/deleted by STAFF
  + Inventory media can only be created/updated/deleted by STAFF
  + Everyone can register customer; each customer can only update he/her own info. Staff can update all customer’s info
  + Only STAFF allow to deactivate customer
  + Order should be created only for register customers
  + Only PENDING order can be updated/delete by its own customer or STAFF
  + Customer can update its own CUSTOMER contact anytime; STAFF can update all customers’ CUSTOMER contact. SHIPPING (order) contact can only be updated by its own customer or any STAFF when order status is PENDING
  + Only customers can pay their own Payment, STAFF cannot even touch any payment
* Product
  + complete post product endpoint implementation
  + complete update product endpoint implementation
  + complete deactivate product endpoint implementation (logical delete)
  + product displayName is required
  + currency is fixed to CAD in database no matter what incoming request says
  + price I want request pass in a fraction of number (eg. 5.35) and store in database as a Long (eg. 535)
  + isActive default set to true, but if the request sends isActive to false, then set it false
  + remark is optional, request can send remark or not
  + you need to create inventory before create product
  + the request will have to specify the which inventory it connecting to in order to create or update product
* Inventory
  + complete post inventory endpoint implementation
  + complete update inventory endpoint implementation
  + complete delete inventory endpoint implementation (physical delete), inventory can be deleted only if there is no product associated to it
  + inventory sku must be set automatically with random unique string internally no matter what request says
  + inventory createTime should set to current time no matter what request says
  + weight is a mandatory field for request
  + description must be sent through request (mandatory)
  + remark is optional
  + dimension can be sent with inventory creation and update request and cannot be empty if it sent through creation request
* Dimension
  + Complete post dimension endpoint implementation
  + Complete update dimension endpoint implementation
  + When create dimension, it also allows to pass inventory Id along with the create request to connect the dimension directly to that inventory
* Inventory Media (Advanced)
  + inventory media should be stored on amazon s3 service, staff can upload image/video/text files
  + complete post inventory media endpoint implementation, and it should upload media file to s3 service, there should no file be stored on local
  + for post request, you must specify the inventory it connect to and of course, link the existing inventory with inventory media object
  + complete update inventory media endpoint implementation, for update, you also need to delete the existing media file on s3 and upload the new file
  + complete delete inventory media endpoint implementation (physical delete), and it should also delete the file on s3 service
  + type is a mandatory field that must be passed through create request
* Order
  + complete post order endpoint implementation
  + complete update order endpoint implementation
  + complete delete order endpoint implementation (physical delete) only if the payment has not been created
  + order status should be set to PENDING when first created no matter what request says
  + to create an order, you must pass an item object list which is derived from products
  + order contact should be a mandatory field when creating an order, if request does not provide contact info, you should copy customer contact as the order contact and save it
* Customer (Advanced)
  + complete post customer (register) endpoint implementation
  + complete update customer endpoint implementation
  + complete delete customer endpoint implementation (logical delete)
  + status should set to ACTIVE once customer get created, no matter what request says
  + totalSpend should set to 0 when customer get created, and when customer pay any order, totalSpend should add the amount
  + user is the default setup for jHIpster, it contains the basic login info for all customers, in this case, when customer created, you must create a user associated with it, and you must set the user authority to “ROLE\_USER”
  + taxExempts is certificates that allow customer to avoid tax charge of the certain region/province/state/etc. Each customer may have multiple taxExempts
  + contact is the default shipping/billing address for customer, it is a mandatory field when creating a customer.
  + Contact type should set to CUSTOMER no matter request says
* Contact
  + Contact is the wrapper of Address, it just adds name/phone/email info against Address, contact should never be created independently, it should be created by other endpoints. But you should be able to use contact endpoint to update contact info
  + Complete update contact endpoint to update both Contact and Address attributes
  + Order and customer should never be updated through contact update endpoint
  + Complete delete endpoint to deactivate contact (logical delete)
* Address
  + Address endpoints should be never exposed to anyone. All address attributes change should be completed by contact endpoints
* Tax Exempt
  + complete post tax exempt endpoint implementation, which also require to create a tax region within the same request
  + complete update tax exempt endpoint implementation
  + tax region of the tax exempt should allow to be updated within the same request
  + complete delete tax exempt endpoint implementation (physical delete)
  + each tax exempt should have a corresponding tax region which illustrate which region’s tax customer can be avoided
  + all fields in tax region are mandatory when creating tax exempt
* Payment
  + Complete the “/pay/{orderId}” endpoint, and customers should use this endpoint to pay their order
  + (In Depth) Create a new database field in Liquibase, entity and dto called “PaymentStatus”, which should be an enum value of {COMPLETED, FAILED, PENDING}
  + Choose 1 of the following 2 options:
    - (Advanced) You only need to implement only 1 payment method: CREDITCARD, which using the GLOBALPAYMENTS API. Once any request to the pay endpoint, you should initialize a transaction to GLOBALPAYMENTS and complete the transaction, and record the response in remark field, and then set payment status and the corresponding order status to status you received
    - (Basic) once customer call pay endpoint, set order status to “COMPELTED”
  + Once payment successfully completed, update customer totalSpend field