CMA Developer Program - Case Study 1

Frosters is a Retailer who is planning to extend their presence online and automate some of the existing inventory processes.

Step 1: Build following Services. Use in-memory H2 database.

1. Customer Service
2. Vendor Service
3. Inventory Management Service

Step 2: Write Junit test cases for the application and use Jacaco for checking code coverage. Test and ensure atleast 90% Code Coverage and all test cases pass.

Step 3: Configure Flyway DB Migration and your script files to create tables and populate data. Override spring default configuration "ddl-auto : none" to avoid auto creation of tables.

More Specific Details on the Services

Customer Service

Customer JSON Data:

customerId (Long)
customerName (String)
contactNumber (Long)
address (String)
gender (String)

1. Get All Customers
[GET] /customer/

Returns all the customers
Returns status 200 if any customer is present
Returns status 404 if no customers are present

2. Get Customer By Id
[GET] /customer/{id}

Returns all the customer based on the id
Returns status 200 if customer is present
Returns status 404 if customer is not present

3. Add New Customer
[POST] /customer/

Response body should have the customer details to be saved
Returns the saved customer along with the id
Returns status 201 if customer is created

4. Update Existing Customer
[PUT] /customer/{id}

Response body should have the customer details to be saved
Returns the updated customer
Returns status 404 if customer is not present
Returns status 200 if customer is updated

5. Delete Customer
[DELETE] /customer/{id}

Returns status 404 if customer is not present
Returns status 200 if customer is deleted

6. Delete All Customers
[DELETE] /customer/

Returns status 200 if customers are deleted

Vendor Service

Vendor JSON Data

vendorId (Long)
vendorName (String)
vendorContactNo (Long)
vendorEmail (String)
vendorUsername (String)
vendorAddress (String)

1.Get All Vendors
[GET] /vendor/

Returns all the Vendors
Returns status 200 if any vendor is present
Returns status 404 if no Vendors are present

2. Get Vendor By Id
[GET] /vendor/{id}

Returns all the vendor based on the id
Returns status 200 if vendor is present
Returns status 404 if vendor is not present

3. Add New vendor
[POST] /vendor/

Response body should have the vendor details to be saved
Returns the saved vendor along with the id
Returns status 201 if vendor is created

4. Update Existing vendor
[PUT] /vendor/{id}

Response body should have the vendor details to be saved
Returns the updated vendor
Returns status 404 if vendor is not present
Returns status 200 if vendor is updated

5. Delete vendor
[DELETE] /vendor/{id}

Returns status 404 if vendor is not present
Returns status 200 if vendor is deleted

6. Delete All vendor
[DELETE] /vendor/

Returns status 200 if vendor are deleted

Inventory Service

Sku JSON Data

skuId (Long)
productName (String)
productLabel (String)
inventoryOnHand (int)
minQtyReq (int)
price (double)

1. Get All Items
[GET] /item/

Returns all the items
Returns status 200 if any item is present
Returns status 404 if no item are present

2. Get Item By Id
[GET] /item/{id}

Returns all the item based on the id
Returns status 200 if item is present
Returns status 404 if item is not present

3. Add New item
[POST] /item/

Response body should have the item details to be saved

Returns the saved item along with the id
Returns status 201 if item is created

4. Update Existing item
[PUT] /item/{id}

Response body should have the item details to be saved
Returns the updated item
Returns status 404 if item is not present
Returns status 200 if item is updated

5. Delete item
[DELETE] /item/{id}

Returns status 404 if item is not present
Returns status 200 if item is deleted

6. Delete All item
[DELETE] /item/
Returns status 200 if item are deleted