WRITTEN BY MR. HUYNH NAM
YOUTUBE CHANNEL: https://www.youtube.com/@nammedia

WORKSHOP ON MVC, MIDDLEWARE AND ERROR PROPAGATION

Topic: Build RESTFUL API to implement CRUD application with Node JS.

Requirement 1

Build RESTFUL API with product data stored in data.json. Product object has four properties: code, name, price and quantity.

Table APIs description on routes of product:

Method	**URIs**	**Endpoints**	**Description**
GET	products/	/	Get all products
GET	products/:id	/:id	Get product by code
POST	products/	/	Create new product
PUT	products/:id	/:id	Modify product with code
DELETE	products/:id	/:id	Delete product by code

Watch video Lesson 5 on Youtube Channel https://www.youtube.com/@nammedia

Software Design and Architecture: *Attend the offline class to discuss with lecturer.*

Lab: Determine the layer to modify the logic to satisfy requirments:

1. Add method total in Product class by calculating quantity*price
2. Endpoint “GET /:id” : If the product has total is more than 1000 then writing in uppercase the name in result returned.
3. Endpoint “DELETE /:id”: Prevent removing the item whose quantity is more than zero

Requirement 2

Extend more requirements to build RESTFUL API with transaction data. Build RESTFUL API with transaction data stored in ***trans.json***. Transaction object has four properties: tid (id of transaction), pcode (code of product), tamount (number of item processed in transaction) and ttime (the timestamp when recording transaction).

Table APIs description on routes of transaction:

Method	**URIs**	**Endpoints**	**Description**	**Logic**
GET	trans/	/	Get all transactions	
GET	trans/:id	/:id	Get transaction by id	
POST	trans/	/	Add new transaction	The tid is format as TXYYYYY, in there Y is digit. The *tamount* must be more than zero. The *tamount* must be less than quantity of product. When transaction is recorded successfully then updating **quantity = quantity – tamount**. The *ttime* is format **hh:mm:ss:dd:mm:yyyy** which generated automatically by following server timer.
PUT	trans/:id	/:id	Modify transaction with id.	Can not modify transaction.
DELETE	trans/:id	/:id	Remove transaction by id.	Can not delete transaction.

Table APIs description on routes of tx:

Method	**URIs**	**Endpoints**	**Description**
GET	tx/find/:code	/find/:code	Get all transaction by product code
GET	tx/summary/:code	/summary/:code	Return json data as {**count**: *number of transaction that has product code*, **total**: *sum of (tamount*price) of transaction that has product code*}

Lab: Student research Swagger to demonstrate APIs.