**WMS MVP**

As part of this exercise, you will be writing two programs 1) A client program using core PHP(**Marketplace order generator)** 2) A REST API using simple PHP development server, this is for a very tiny WMS(**WMS simulator API)**. You can use any database/Simple .txt file for the storing the data (SQLite, MySql, or .txt file etc..)

We want to see your software design and coding ability.

**Marketplace order generator:**

  The client should be simulating the generation of orders from one or more ecommerce Marketplaces. Take an input of Source from User before client is run like below: “amazon, flipkart, jiomart”.

  An order consists of a unique identifier (Marketplace code) called the “source” and an order number and a demand for between zero and five units each of items A,B,C,D, and E.

Rules:

1. At least one unit demanded of any one of the SKUs for an order to be valid
2. You can’t request for 0 quantity of an order.
3. You can’t request for more than 5 quantities

  A valid order (in whatever format you choose): {"source": “flipkart”,”order\_number”:1001, "Lines": {"Product": "A", "Quantity": "1"},{"Product": "C", "Quantity": "4"}}

  An invalid order: {"source": “jiomart”,”order\_number”:1001, "Lines": {"Product": "B", "Quantity": "0"}}

  Another invalid order: {"source": “amazon”,”order\_number”:1001, "Lines": {"Product": "D", "Quantity": "6"}}

**WMS simulator API:**

Please build an API backend using PHP Development server, which will be called by the Marketplace Order Generator. This is the main core of the program, and you are expected handle the inventory reduction logic as given below once you receive a valid order.

The same rules given for order generation should be validated by the API before proceeding with processing the order further.

1. At least one unit demanded of any one of the SKUs for an order to be valid
2. You can’t request for 0 quantity of an order.
3. You can’t request for more than 5 quantities

Once you receive a valid order store that in a table and process it based on the below logic to adjust the inventory and if successful send a 200 status message to the client. If the order is not valid return 400 as the status code.

  The inventory allocation will be according to the following rules:

1. Each order should have a source element to identify the Marketplace that is submitting it.
2. Once an order is valid and received reduce the inventory for items on what is requested in order
3. Keep reducing the inventory based on valid orders. Once inventory of all items is either 0 or negative return a 500 error message to the client and once the client receives 500 it should stop sending further orders.
4. When one of the items has inventory if you receive order for another items it should be reduced even if the quantity goes below 0 – which is called backordering. Example: Starting point of inventory table is [ A=1 and B=1 ]. a) Order received {A=1} Now the inventory will be [A=0 and B=1] New order received {A=1, B=1}. Now inventory will be [A=-1 and B=0]

Have an end point on the api as /order\_report calling this will produce a JSON response which will list all the valid orders received and what was the backordered values for the SKUs.

**Inventory setup**

Before starting the run of the client, Please provide a set up file which can be edited and which will have SKU=QTY as input in each line and have a command to run setup which will create your schema and set up SKUs and inventory and the required order table to store the valid orders.

Below is an example.

  A=3

  B=2

  C=5

  D=4

  E=2

  When we run the setup command make sure all tables have been dropped and schema recreated and the inventory is updated correctly by SKU(product id)