# Sample Application II: Warehouse Management System

**Please download resources from:** [**https://smu.sg/hq0**](https://smu.sg/hq0)

**Given:**

* q2/model
  + ConnectionManager.php (**complete**)
  + InventoryRecord.php (**complete**)
  + InventoryRecordDAO.php (**complete**)
  + Item.php (**complete**)
  + ItemDAO.php
  + Warehouse.php
  + WarehouseDAO.php
* q2/
  + common.php (**complete**)
  + add.php
  + analytics1.php
  + analytics2.php
  + analytics3.php
  + index.php
  + list\_common\_items.php
  + process\_add.php
  + process\_update.php
  + wms.sql *(****run this before you start)***

# This is a warehouse management system application. It provides the following functions for the user:

* update the inventory level for a warehouse
* add a new warehouse
* analytics 1 – to identify the top 2 warehouse with the most items
* analytics 2 - find common items between 2 warehouses
* analytics 3 – find out pairs of items that always appear together in warehouses.

The application uses three classes – Item, InventoryRecord and Warehouse. The Item class provides a unique id, name and description for each item in the warehouses. The InventoryRecord class provides the object which the user is able to find out about the quantity of an Item in a warehouse. The Warehouse class can be used to identify a warehouse and has a unique id, name and location as its property.

The corresponding DAO for each of the classes are ItemDAO, WarehouseDAO and InventoryRecordDAO. The data stored in the database can be viewed via PHPMyAdmin.

## Part A - Display Warehouse Listing

Make changes to the home page index.php such that when the page is loaded, it will display all warehouses which are available in the database, as shown in diagram below. The following points should be taken into considerations.

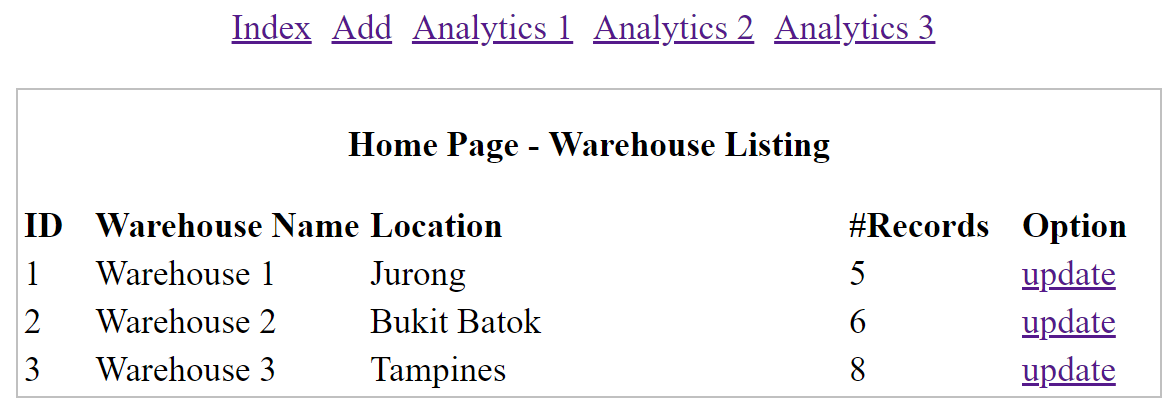
* The list of warehouses following the order they are returned by the getAll() method of WarehouseDAO
* **#Records** is the number of inventory records stored in that warehouse



**Part B - Update Warehouse Inventory Records**

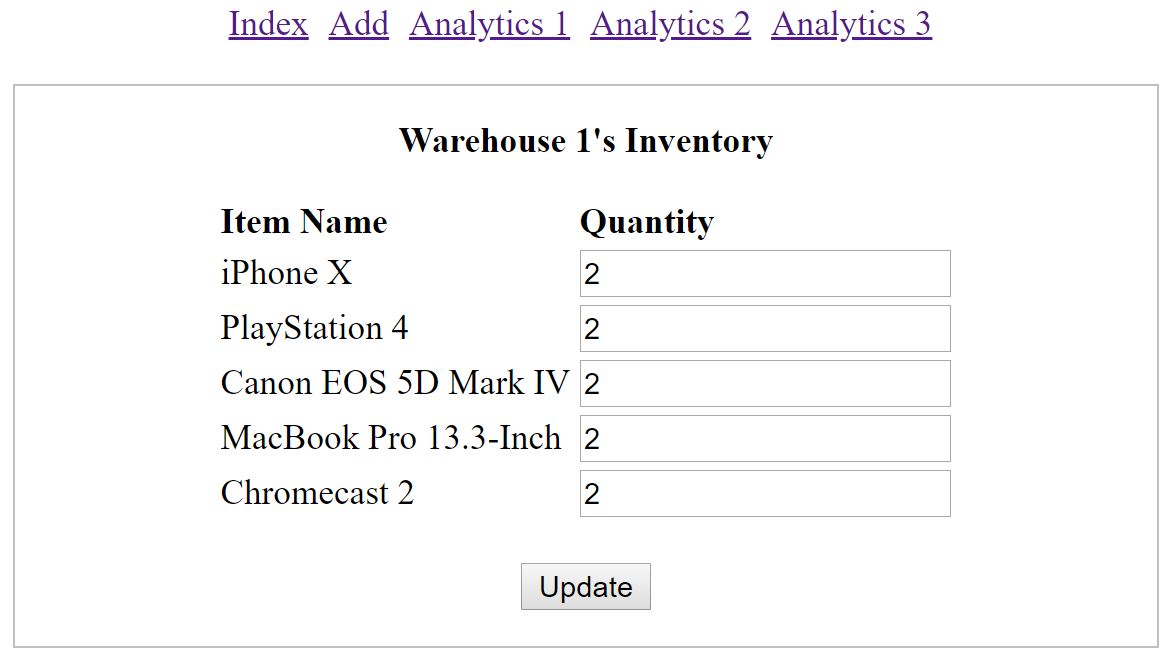
To implement the change warehouse functionality option by following these steps:

* Create hyperlinks to update.php from each line of the warehouse listing (in column **Option**) shown in index.php
* The id of the corresponding warehouse needs to be passed from index.php to update.php by appending it at the end of the URL pointed by each of the hyperlink.
* index.php should appear as follows:

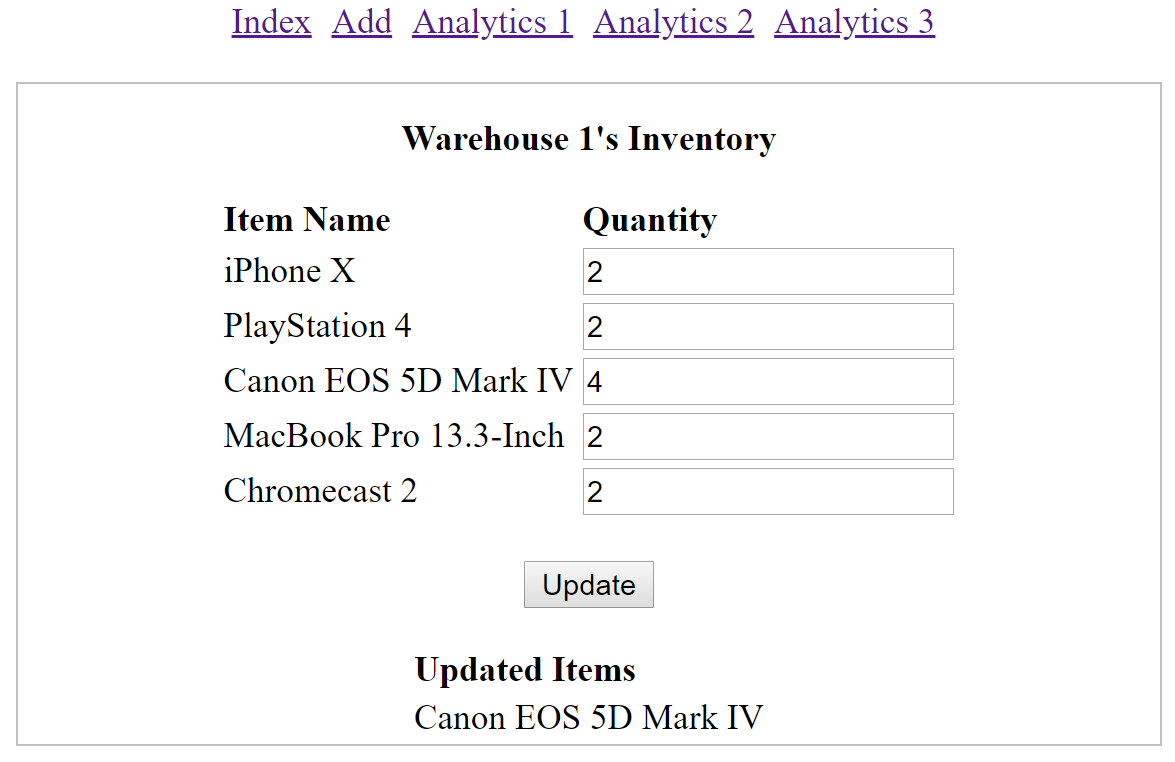


The user will be allowed to modify quantities of items in the selected warehouse. You must implement the provided update.php to:

* Retrieve warehouse id passed using HTTP GET   
  (e.g., <http://localhost/sampleapp2/resources/update.php?warehouse_id=1>)
* Display warehouse name at the top of the page.
* Display the list of items in the warehouse and their quantities.
* The page will look as below when done correctly.



* After **Update** button is clicked, the form is submitted to process\_update.php   
  The program process\_update.php should:
  + Save the updated details and redirect to update.php
  + The list of updated inventory will be shown at the bottom of update.php
  + If no inventory items are updated, the heading **Updated Items** should not be shown



**Part C - Debug Add Warehouse**

The functionality "Add Warehouse" has already been implemented (in add.php and process\_add.php) but the developers are having some problem getting it to work correctly. User just needs to enter the name and location of the new warehouse. The ID of the warehouse will automatically be generated by the database.



The process\_add.php will do server-side validation as follows:

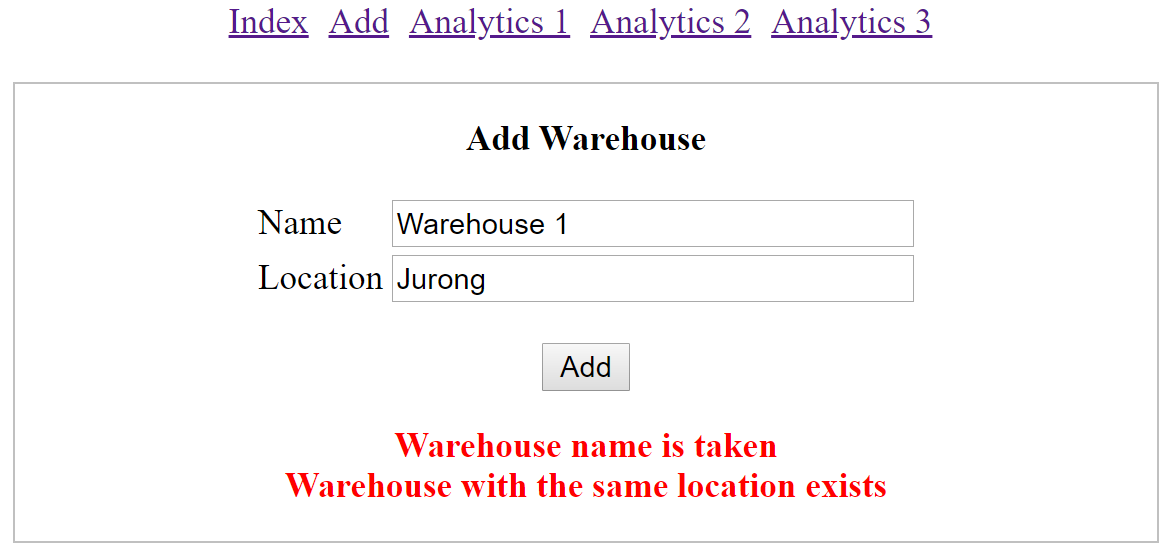
* + If warehouse name exists in the database, error message "Warehouse name is taken" will be generated.
  + If location exists in the database, error message "Warehouse with the same location exists" will be generated. Assume that the location of each warehouse is unique

If there are errors in the submitted form values, process\_add.php will:

* Put generated error messages as an array in $\_SESSION.
* Forward back to add.php.

In turn, add.php will:

* Display the list of error messages. Order of error messages does not matter.
* Any user's entered/selected values will be displayed in add.php so that user does not need to enter them again.



If there is no error in the submitted form values, process\_add.php will:

* Add a new warehouse.
* Redirect the admin to index.php. The new warehouse should be listed correctly in index.php

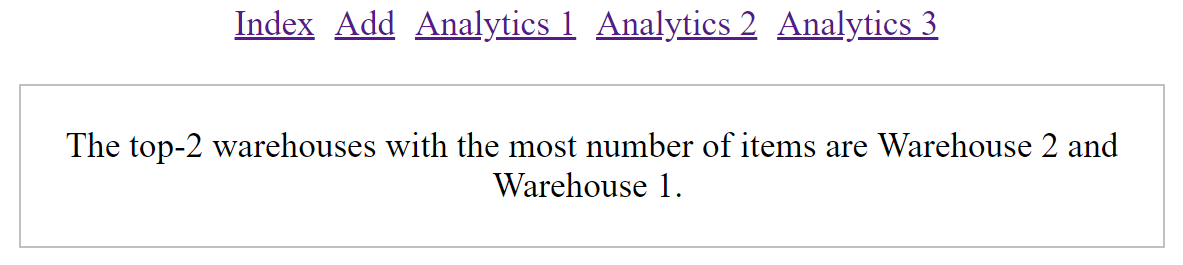
Fix all the problem(s) you can find in add.php and process\_add.php to get them to work correctly.

**Part D - Analytics**

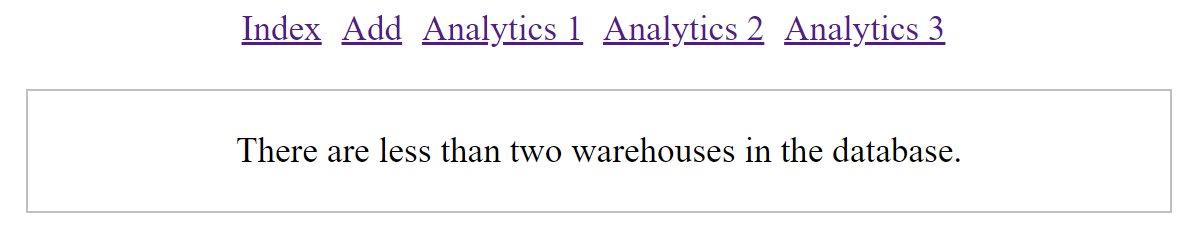
We have analytics pages to help administrators understand how items are stored in warehouses:

* Find the top-2 warehouses with the most number of items
* Find items that are stored in common in a set of warehouses
* Find pairs of items that always appear together in warehouses (i.e., whenever one item of the pair appears in a warehouse, the other appears in the same warehouse too)

The first analytics page (<http://localhost/sampleapp2/resources/analytics1.php>) should display the following if there are more than two warehouses in the database:



Otherwise it will display the following:



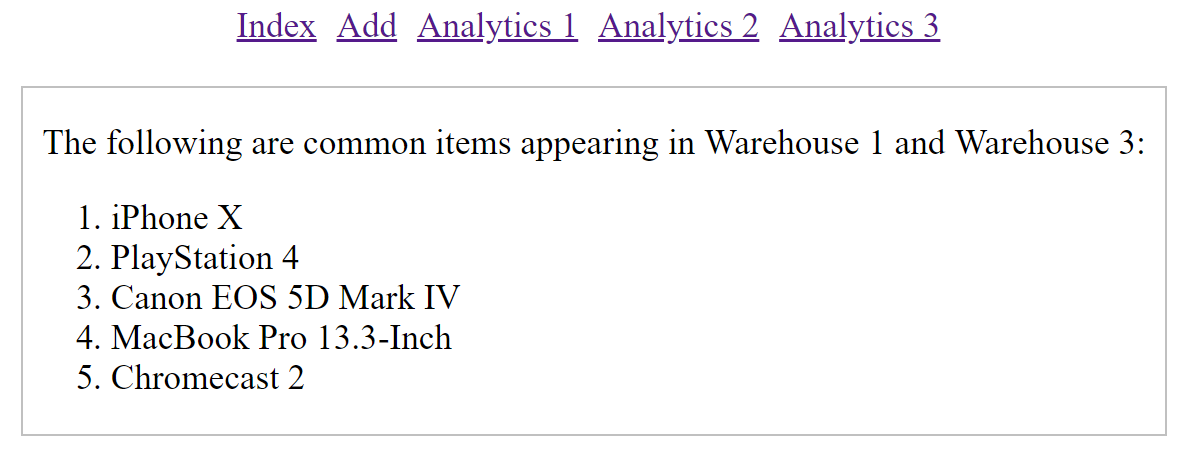
Complete the getTopNWarehousesWithMostItems method of WarehouseDAO (instructions are provided in WarehouseDAO.php). The SQL command to retrieve the top-10 warehouses with the most number of items is:



The second analytics page (<http://localhost/sampleapp2/resources/analytics2.php>) should display:



After some warehouses are selected (e.g., Warehouse 1 and Warehouse 3) and **Find Common Items** button is clicked, send the **list of ids** of selected warehouses to list\_common\_items.php. It should display the following:



The third analytics page ( <http://localhost/sampleapp2/resources/analytics3.php>) should display:

