**CSC 5750 – Principles of Web Technology**

**Homework 6 – Part 2 of 2 – Caleb Latimer**

**18 of 25 points – Due December 7, 6pm**

**Late deadline is December 9, 11:59pm, but 3 points off**

**a)** Save this document with your name and the homework number somewhere in the file name.

**b)** Type/paste your answers into the document.

**c)** Gather the following files into a ZIP file:

● This document

● Cashier page (.HTML)

● Any other optional files specific to your implementation

**d)** Submit the ZIP file to the Blackboard item where you downloaded this document.

You've been hired again by the local restaurant that you created a web site for in Homeworks 1, 3, 4, and 5. Instead of using local storage, data will be stored in a MySQL database. You are provided the following two files:

**RestaurantSales-DatabaseBuild.sql**

This SQL script will create database **dbRestaurantSales** and its two tables:

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Fields | Initial data | Rows |
| tbSales: stores information about each sale. | saleID – INT (PK\*)  entreeIndex – INT  drinkIndex –INT  tipPercentage – INT | (1001, 1, 1, 10)  (1002, 1, 2, 15)  (1003, 2, 1, 20)  (1004, 2, 2, 25) | One per sale. |
| tbStatus: stores last sale ID, current sales count, and last sale viewed. | saleID – INT  saleCount – INT  salePtr – INT | (1004, 4, 0) | Exactly one. |

\* PK = primary key

**RestaurantSales-DatabaseInterface.php**

This PHP script is the interface to and handles communication with database **dbRestaurantSales** and its two tables. Modify the connection parameters (lines 29-32) per your MySQL installation. The script responds to the following four AJAX (XMLHttpRequest) requests from the Cashier page. Note that in the Homework 5 key, **xhttp1** was used to get file JSONentrees.txt, and **xhttp2** was used to get file JSONdrinks.txt:

| AJAX request  AJAX object | DatabaseInterface function  Cashier page request and response |
| --- | --- |
| readSalesVars  xhttp3 | **Cashier page request**  RestaurantSales-DatabaseInterface.php?request=readSalesVars  **Database interface function**  Read values from table **tbStatus** including saleID, saleCount, and salePtr.  **Database interface returned string**  Query error message  OR  Sales variables - #saleID#saleCount#salePtr#  **Cashier page response**  ● Parse returned string into three variables using this.responseText.split("#").  ● Show responseText in status message label (lStatus). |
| readSalesArray  xhttp4 | **Cashier page request**  RestaurantSales-DatabaseInterface.php?request=readSalesArray  **Database interface function**  Read sales from table **tbSales**, and build and encode PHP array.  **Database interface returned string**  Query error message#result#  OR  Query success message#result#salesArray#  Where:  **result** is "good" or "bad". If result is "bad", don't attempt to parse the sales array.  **Cashier page response**  ● Parse returned string into three variables using this.responseText.split("#").  ● Show first variable in status message label (lStatus).  ● Parse third variable (salesArray) using JSON.parse. |
| writeSalesVars  xhttp5 | **Cashier page request**  RestaurantSales-DatabaseInterface.php? + qstring  Where:  qstring = "request=writeSalesVars" +  "&saleID=" + saleID +  "&saleCount=" + saleCount +  "&salePtr=" + salePtr;  **Database interface function**  Write values to table **tbStatus** including saleID, saleCount, and salePtr.  **Database interface returned string**  Query result message  **Cashier page response**  Show responseText in status message label (lStatus). |
| writeSalesArray  xhttp6 | **Cashier page request**  RestaurantSales-DatabaseInterface.php? + qstring  Where:  qstring = "request=writeSalesArray" +  "&sales=" + JSON.stringify(sales)  **Database interface function**  Clear all sales from table **tbSales**, decode JavaScript array, and write sales to table **tbSales**.  **Database interface returned string**  Query result message  **Cashier page response**  Show responseText in status message label (lStatus). |

**AJAX object summary**

Note that each AJAX request should be synchronous. This means the third argument should be **false**. For example:

xhttp1.open("GET", "JSONentrees.txt", false);

| Object | Purpose | Calling function |
| --- | --- | --- |
| xhttp1 | Get file **JSONentrees.txt** from the server. | loadComboBoxes |
| xhttp2 | Get file **JSONdrinks.txt** from the server. | loadComboBoxes |
| xhttp3 | Read **sales variables** from the database. | loadData |
| xhttp4 | Read the **sales array** from the database. | loadData |
| xhttp5 | Write **sales variables** to the database. | saveData |
| xhttp6 | Write the **sales array** to the database. | saveData |

**Cashier page enhancement**

Modify the **Cashier** page per the following:

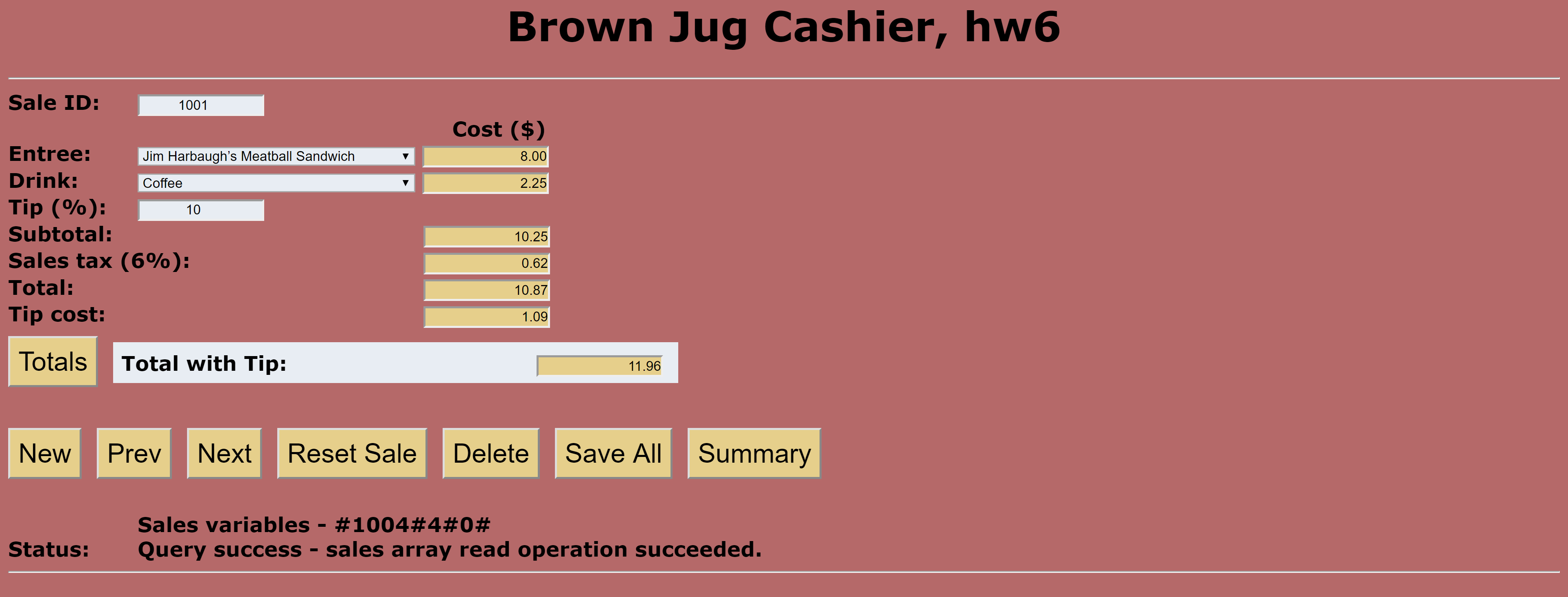
● In function **LoadData**, instead of reading data from local storage, construct AJAX requests **xhttp3** and **xhttp4** as described above.

● In function **SaveData**, instead of saving data to local storage, construct AJAX requests **xhttp5** and **xhttp6** as described above.

● Replace button **Reset All** with button **Save All**. When Save All is clicked, call function **SaveData**.

● Add one label with text "Status:" and another label with ID **lStatus** that may show query result messages. This is needed for AJAX requests xhttp3, xhttp4, xhttp5, and xhttp6:

lStatus.innerHTML = this.responseText;



**Cashier page**

*[your Cashier page HTML code here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

*[your Cashier page screenshots here]\*\**

\* **Copying-and-pasting web page code to a Word document**

1) From the HTML editor window, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

\*\* **Copying-and-pasting application output to a Word document**

1) From the web page, maximize the browser window.

2) From the browser window, press **ALT-PrintScreen**.

3) From within the Word document, press **CTRL-V**.