

**CZ3006: Net Centric Computing  
AY 16/17 Semester 2**

**Assignment 2 Report**

**Student Name: Sim Long Siang**

**Matriculation Number: U1522053H**

**Lab Group: TS1**

**9 April 2017**

# Introduction

The objective of this report is to document the various features of the Fruit Buying System. The Fruit Buying System is a web-based application created using JavaScript and PHP language. It consists of the client side (written using HTML and JavaScript) and the server side (written using PHP). For this system, there are several main criteria to be met:   
  
1. The validity of the text box input whenever it is changed is checked at the client side. If the input is not valid, the user is prompted with an alert message and are required to input again. If the textboxes used to get inputs of the quantity of the fruit the user wants to buy contains an invalid value, it will display “NAN”.

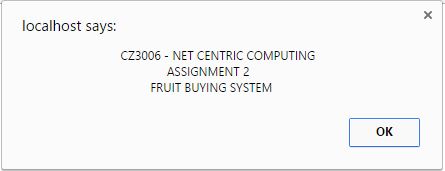
2. Calculation of the total cost is to be done at the client side using JavaScript and a text box are to be created to show the total cost of the order. This textbox should be blurred whenever it acquires focus.

3. A submit button should be in place to send the user’s order to the web server.

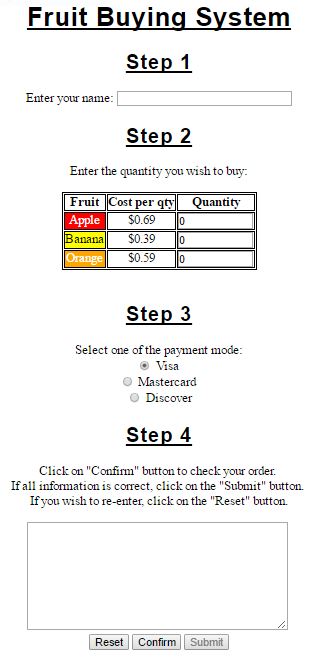
4. On receiving the order from the client side, the server side are required to calculate it and returns an HTML document to the user as a receipt, specifying all information entered by the user and the final cost.

5. The server are required to create and update a file named “order.txt” to store the user’s order information.  
  
The report will first walkthrough the whole process of the program from the ordering to the final calculation of the cost. After which, each criterion will be explained in details. The full code for both the client side and the server side are attached at the end of the report.

**Walkthrough of the Application**1. Upon loading the application



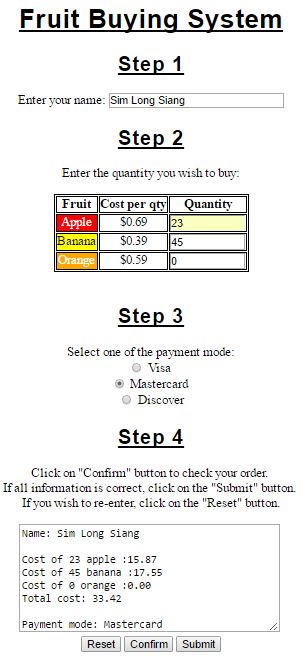
2. Client’s order form  
  
There are some assumptions made to the application and are implemented as follows:  
  
1. The users are required to press the “confirm” button to check his/her order before sending to the server by pressing the “submit” button. Therefore, the “submit” button is initially disabled, and will be enabled once the “confirm” button is pressed.



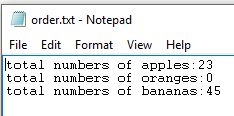
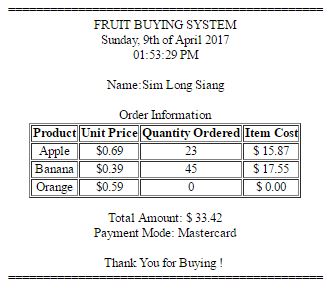
2. Upon pressing the “reset” button, the form will clear all the inputs and the “submit” button will be disabled.

3. The system will require the “Name” textbox to be filled before submitting to the server side. Therefore, an alert message will be shown upon pressing the “confirm” button if the textbox is empty.

3. After pressing the “Confirm” button



4. After submitting the order to the sever side



***Left:*** *receipt of order* ***Right:*** *order file updated by the server.*

**Criterion Explanation**

**1. Validity of the text box input**For the validation to happen immediately after a input change, the function *onchange()* is used for all textboxes. Upon detecting a change in the inputs, it will call it’s respective function to carry out the validation.*<input type="text" name ="user" size ="30" maxlength ="30" id="user"* ***onchange = "namevalidate('user');****">  
…  
<input type="text" name ="apple" size ="10" maxlength ="10" id="1"* ***onchange = "numvalidate('1');****" value = "0">   
<input type="text" name ="banana" size ="10" maxlength ="10" id="2"* ***onchange = "numvalidate('2');****" value = "0">*

*<input type="text" name ="orange" size ="10" maxlength ="10" id="3****" onchange = "numvalidate('3');****" value = "0">*

For name validation, the function first store the value of the “name” textbox element in a variable. The variable is then used together with the *search()* function, a pattern searching function to check if the pattern defined fits the value of the variable. The *search()* function will return the position of where the pattern starts to match. If the position returned is 0, it means the pattern starts matching correctly at the beginning. Therefore, if the position returned is not 0, an invalid input is detected and an aler message will prompt the user to enter again.

*function namevalidate(index){*

*var str = document.getElementById(index).value;//Get value of textbox input*

***var position = str.search(/^\w\D\*$/);*** *//pattern matching defined*

*if(position != 0){*

*alert("Invalid Name ! Please enter again !");*

*document.getElementById(index).value = '';*

*document.getElementById(index).focus();*

*}*

*}*

The validation for the “quantity” text box is like the name validation. The only difference is the pattern defined for validating the number. While the pattern for name validation defines only letters, the pattern for number validation defines only digits. In addition, instead of clearing the textbox, if an invalid input is detected for the quantity textbox, the textbox will be placed with “NaN” value.

*function numvalidate(index){*

*var str = document.getElementById(index).value; //Get value of textbox input*

***var position = str.search(/^\d\*$/);*** *//pattern matching defined*

*if(position != 0){*

*alert("Invalid Input ! Please enter again !");*

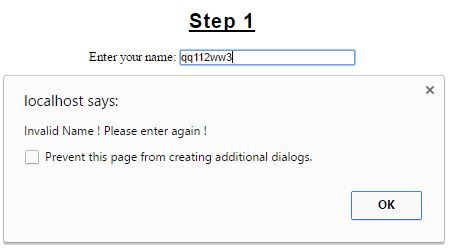
*document.getElementById(index).value = 'NaN';*

*document.getElementById(index).focus();*

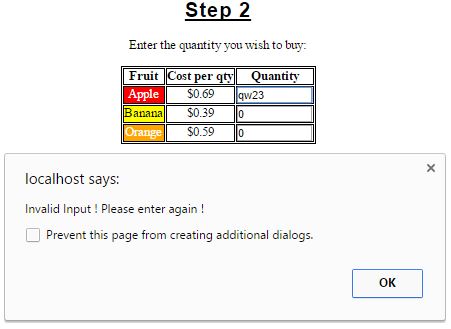
*}*

*}*

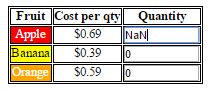
Below shows the execution of the validation mentioned above:



1.Name Validation



2. Number Validation



3. After number validation

**2. Calculations and display of result**  
  
The calculation and display of the result are handled by the function *compute()* upon pressing the “confirm” button. It will first check if the “Name” textbox has an input (it does not require to check for valid input as it is done immediately on input change). After which, it will convert all “NaN” values to zero, as we assumed that the user does not want to buy that fruit anymore. Subsequently, the “quantity” inputs will be stored in individual variable for calculation. *compute()* function also keep tracks of which radio button is checked using a variable. Lastly, all required information are then displayed out by setting the value of the “result” textarea.  
  
*function compute(){*

*if (document.getElementById("user").value == ""){*

*alert("Name must be filled out !");*

*return false; }*

*else{*

*for(n=1;n<4;n++){*

*if(document.getElementById(n).value == 'NaN')*

*document.getElementById(n).value = 0; }*

*var applecost = document.getElementById("1").value \* 0.69;*

*var bananacost = document.getElementById("2").value \* 0.39;*

*var orangecost = document.getElementById("3").value \* 0.59;*

*var result = applecost + bananacost+ orangecost;*

*var paymode = "";*

*if(document.getElementById("Visa").checked) paymode = "Visa";*

*if(document.getElementById("Mastercard").checked) paymode = "Mastercard";*

*if(document.getElementById("Discover").checked) paymode = "Discover";*

*document.getElementById("result").value =*

*"Name: " + document.getElementById("user").value + "\n\n" +*

*"Cost of " + document.getElementById("1").value + " apple :" + parseFloat(applecost).toFixed(2) + "\n" +*

*"Cost of " + document.getElementById("2").value + " banana :" + parseFloat(bananacost).toFixed(2) + "\n" +*

*"Cost of " + document.getElementById("3").value + " orange :" + parseFloat(orangecost).toFixed(2) + "\n" +*

*"Total cost: " + parseFloat(result).toFixed(2) + "\n\n" + "Payment mode: " + paymode;*

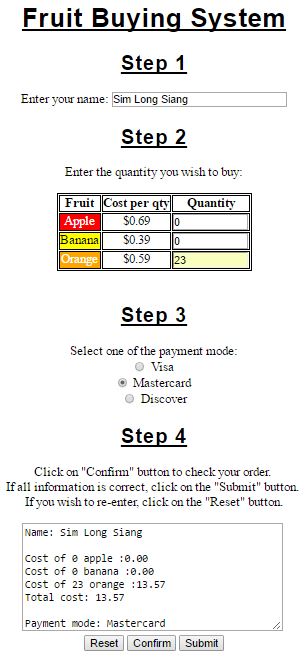
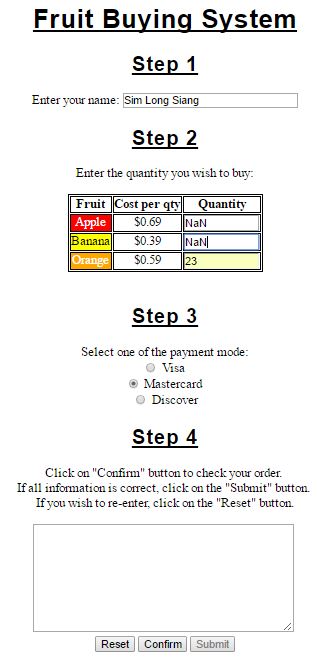
*document.getElementById("submit").disabled = false;*

*}*

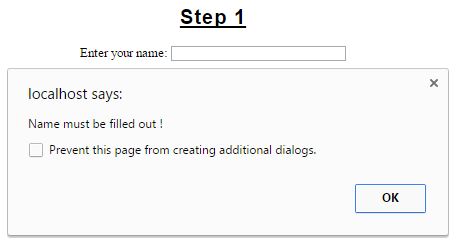
*}*

As mentioned in the criterion, the textarea used to diaply the result should not be edited by users through blurring it when it is focus. The *onfocus()* function helps to satisfy this criterion. It is implemented in such a way that when the textarea is focused, the textarea itself will call the *blur()* function.  
  
*<textarea id="result" rows="8" cols="40"* ***onfocus = "this.blur();****">*

Below shows the execution of the displaying of result mentioned above:



As stated at the walkthrough, the submit button, initially disabled, will be enabled when the “confirm” button is pressed. This is to avoid user to directly submitting the form will empty/invalid inputs to the server.



4.Checking if there is inputs in the textbox

**3. Submit Button**

The submit button is used to send the user’s order to the server side.

*<input type="submit" value ="submit" id="submit" disabled>*

For the user’s order to be sent over to server side through the “submit” button, the URL of the server implementation is defined under the “form action” function.

*<form action = "Assignment2\_Server.php" method = "POST" onreset="return resettime()">*

**4. Calculation and display on server side**

The calculation and display method on the server side is similar to the client side. The only difference is how the server retrieve the inputs from the client side and store it in its variable. This is done by implementing $\_post(“element\_id”) since the client side specified the method of submitting data to be POST method.

*//Calculation of totalcost at server side*

*<?php*

*date\_default\_timezone\_set('Asia/Singapore');*

*$name = $\_POST["user"];*

*$appleqty = $\_POST["apple"];*

*$orangeqty = $\_POST["orange"];*

*$bananaqty = $\_POST["banana"];*

*$payment = $\_POST["payment"];*

*$date = date('l, jS \of F Y');*

*$time = date('h:i:s A');*

*$applecost = $appleqty \* 0.69;*

*$bananacost = $bananaqty \* 0.39;*

*$orangecost = $orangeqty \* 0.59;*

*$totalcost = $applecost + $bananacost + $orangecost;*

*?>*

*//Display of the result as an receipt*

*<?php print("$date<br> $time<br><br> Name:$name<br><br>"); ?>*

*</div>*

*<table border = "border" align="center">*

*<caption> Order Information </caption>*

*<tr>*

*…*

*…*

*<td> <?php print ("$appleqty"); ?> </td>*

*<td> <?php printf ("$ %4.2f", $applecost); ?>*

*</td>*

*</tr>*

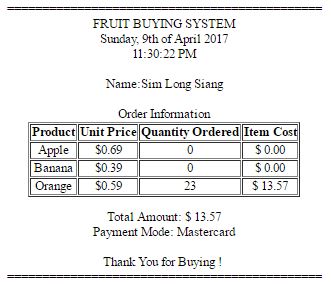
*</table>*

*<p /> <p />*

*<div align ="center">*

*<?php printf ("Total Amount: $ %4.2f<br>Payment Mode: %s",$totalcost, $payment); ?>*

*<br><br>Thank You for Buying ! <br>*Below shows the execution of the displaying of result mentioned above:



**5. creating file and updating the user’s order information**

In order for the user’s order information to be stored as a text file, PHP has a function library to create, write and store information. Firstly, the fopen() function allows the server to open a file for reading or writing (it creates the file if the filename specified cannot be found). Secondly, fwrite() is the main writing function to record any information into the file. Lastly, fclose() function closes and saves the file and its content.

*<?php*

*$myfile = fopen("order.txt", "w") or die("Unable to open file!");*

*fwrite($myfile, "total numbers of apples:$appleqty"."\r\n");*

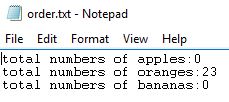
*fwrite($myfile, "total numbers of oranges:$orangeqty"."\r\n");*

*fwrite($myfile, "total numbers of bananas:$bananaqty"."\r\n");*

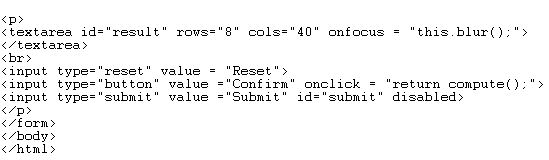
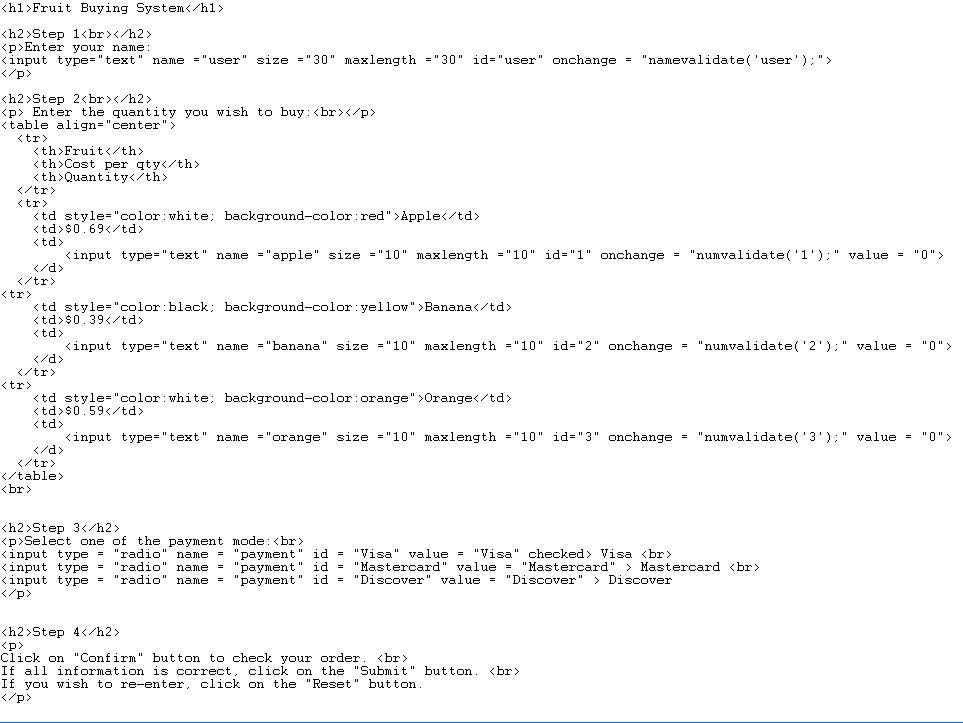
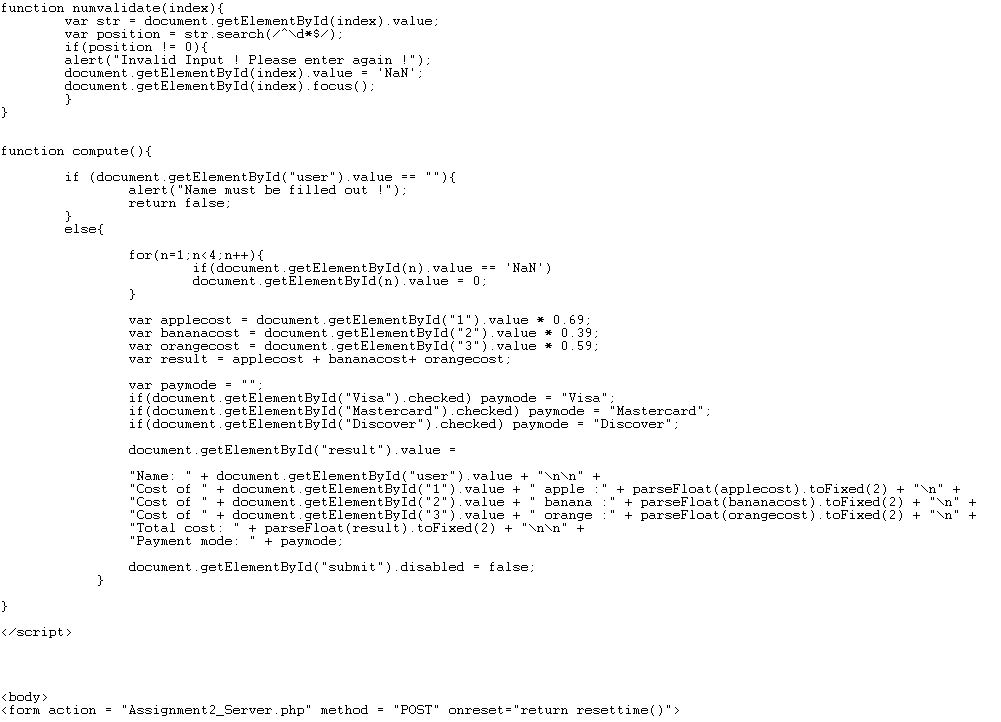
*fclose($myfile);*

*?>*

Below shows the execution of the displaying of result mentioned above:



**Appendix A – Client Side Program Code**



**Appendix B – Server Side PHP Program Code**

