

NANYANG
TECHNOLOGICAL
UNIVERSITY

COURSE CODE : CZ3006

COURSE NAME : NET-CENTRIC COMPUTING

NAME : HTET NAING

MATRICULATION NO. : U1620683D

LAB GROUP : SSP4

COURSEWORK : ASSIGNMENT 2

Contents

Assigned tasks and their status	Page 3
--	--------

Writing a HTML document to create an online fruit order form	Page 3
1. Web UI Design of the fruit order form page	Page 3
2. Validation of the input for the user name text box	Page 5
3. Validation of the input for three fruit quantity text boxes	Page 6
4. Validation of the total cost textbox	Page 7
5. Implementation of the payment mode using radio buttons	Page 8
6. Implementation and validation of the order form submission	Page 9

Writing a server-side PHP program to receive the user's order and generate an order receipt in the form of a HTML document	Page 10
1. Web UI Design of the order receipt	Page 10
2. Implementation of retrieving inputs from HTML	Page 11
3. Generating an order receipt in the form of a HTML document	Page 12
4. Implementation of the update file "order.txt"	Page 12

Listing of source files	
1. index.html	Page 14
2. receipt.php	Page 18
3. style.css	Page 19
4. grid.css	Page 21

Appendix	Page 22
-----------------	---------

Assigned tasks and their status

1. Writing a HTML document to create an online fruit order form.
Status : **Completed**
2. Writing a server-side PHP program to receive the user's order and generate an order receipt in the form of a HTML document
Status : **Completed**

Writing a HTML document to create an online fruit order form

Web UI Design of the order form page

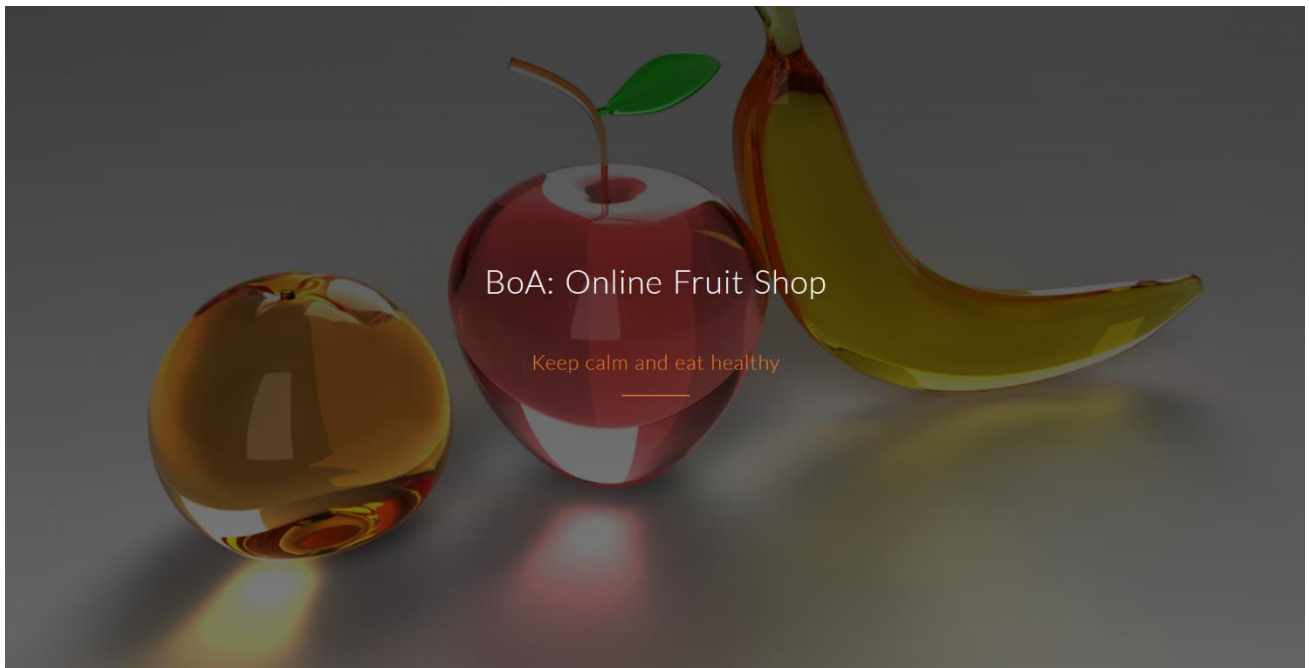


Figure 1: The first section of the online fruit order form page

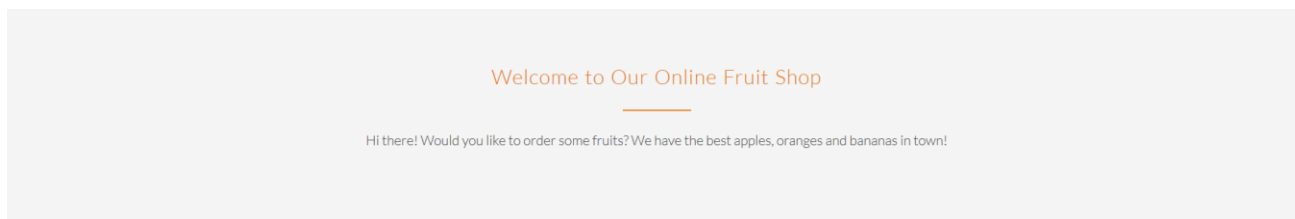


Figure 1.1: The second section of the online fruit order form page

Fruit Order Form

Customer

Apple Price : 69 cents each

Order Apple

Orange Price : 59 cents each

Order Orange

Banana Price : 39 cents each

Order Banana

Total Cost in dollar

Payment method

☒ Visa

☐ MasterCard

☐ Discover

Copyright © 2017 by Htet Naing. All rights reserved. This is created for CZ3006 Assignment 2, Computer Science, NTU.

Figure 1.2: The third section of the online fruit order form page

Validation of the input for the user name text box

The screenshot shows a web form titled "Fruit Order Form". It contains several input fields and labels. The "Customer" field is empty and has a red border, with a tooltip message "Please fill out this field." appearing below it. Other fields include "Apple Price" (59 cents each), "Order Apple" (1), "Orange Price" (59 cents each), "Order Orange" (2), "Banana Price" (39 cents each), "Order Banana" (3), "Total Cost in dollar" (3.04), and "Payment method" (radio buttons for Visa, MasterCard, and Discover). At the bottom are "Reset" and "Submit" buttons.

Figure 2: Checking if the customer name text box is empty

The name of the customer must be provided before submitting the form. Otherwise, a message will be prompted (as seen in fig. 2) to fill in the name. It is implemented as follows:

```
<div class="row">
  <div class="col span-1-of-3">
    <label for="name">Customer</label>
  </div>
  <div class="col span-2-of-3">
    <input type="text" name="customer" id="customer" placeholder="Name" required>
  </div>
</div>
```

Figure 2.1: Implementation for checking if a customer's name is absent in the textbox

By specifying the key word "**required**" in the input tag, a message will be prompted to provide the name whenever a user attempts to submit without filling in this field.

Validation of the input for three fruit quantity text boxes

A user cannot enter an invalid input such as alphabet and special characters for three fruit quantity text boxes. A valid input is defined as a sequence of one or more digits. If a user tries to enter an invalid input, an alert message will be displayed to ask the user to provide a valid input again. It is implemented at the client side using JavaScript as follows:

```
<!-- Javascripts -->

<script type="text/javascript">

    //the method below checks if the input is valid (i.e. it contains only digits), if invalid, it sets total cost to NaN
    function checkValidInput() {
        var numberOfApple = document.getElementById("numberOfApple").value;
        var numberOfOrange = document.getElementById("numberOfOrange").value;
        var numberOfBanana = document.getElementById("numberOfBanana").value;
        var totalCost = document.getElementById("totalCost");

        //checking if the input to order apple is valid (i.e. only contain digits)
        if(numberOfApple.match(/^[0-9]+$/) == null && numberOfApple != ""){
            alert("Invalid input at Order Apple. Please provide a valid integer.");
            totalCost.value = NaN;
            document.getElementById("numberOfApple").focus();
            document.getElementById("numberOfApple").select();
        }

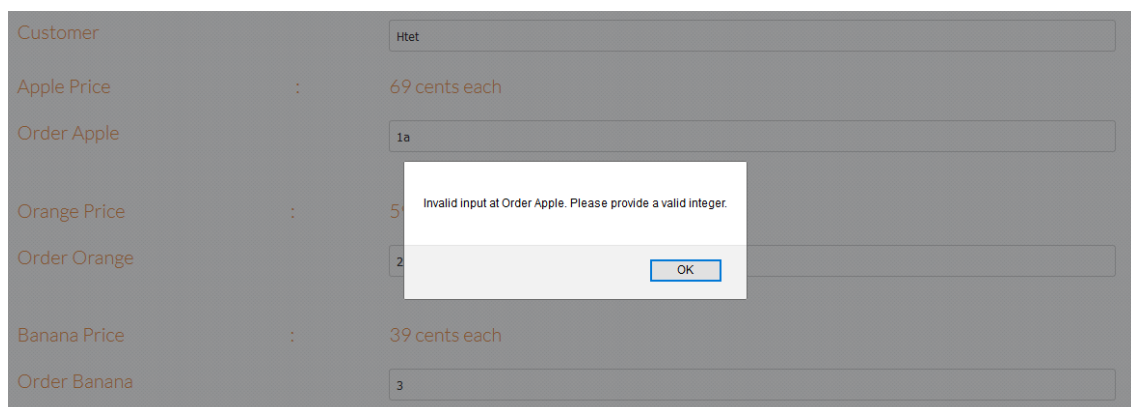
        //checking if the input to order orange is valid (i.e. only contain digits)
        }else if(numberOfOrange.match(/^[0-9]+$/) == null && numberOfOrange != ""){
            alert("Invalid input at Order Orange. Please provide a valid integer.");
            totalCost.value = NaN;
            document.getElementById("numberOfOrange").focus();
            document.getElementById("numberOfOrange").select();
        }

        //checking if the input to order banana is valid (i.e. only contain digits)
        }else if(numberOfBanana.match(/^[0-9]+$/) == null && numberOfBanana != ""){
            alert("Invalid input at Order Banana. Please provide a valid integer.");
            totalCost.value = NaN;
            document.getElementById("numberOfBanana").focus();
            document.getElementById("numberOfBanana").select();
        }
        }else{
            //calculate the total cost if all inputs are valid
            return true;
        }
    }

    return false;
}
```

Figure 3: Checking if an input is valid for ordering fruits

The method “**match(/^[0-9]+\$/)**” helps check if an entered string contains only digits from 0 to 9. Otherwise, it will return null and execute the code that will display the following alert message:



The screenshot shows a web form for ordering fruits. It includes input fields for Customer Name (Htet), Apple Price (69 cents each), Order Apple (1a), Orange Price (5), Order Orange (2), Banana Price (39 cents each), and Order Banana (3). An alert message box is displayed in the center, stating "Invalid input at Order Apple. Please provide a valid integer." with an "OK" button.

Figure 3.1: Displaying an alert message if the input for entering fruit quantity is invalid

Validation of the total cost textbox

The total cost textbox shows the total cost of the order to the customer during his selection. The customer cannot enter any input to the textbox. It will be blurred whenever it acquires focus. The following code implements the blurring of the textbox.

```
<div class = "total-cost">
  <div class="col span-1-of-3">
    <label>Total Cost in dollar</label>
  </div>
  <div class="col span-2-of-3">
    <input type="text" name="totalCost" id="totalCost" placeholder="Total cost" onfocus="this.blur()" >
  </div>
</div>
```

Figure 4: Implementation of the total cost textbox blurring

By having “**onfocus = “this.blur()”**” in the input tag, a particular textbox will be blurred whenever it acquires focus.

The total cost textbox will display the current total cost of the order throughout the user’s selection only if the input is valid for all three fruit quantity text boxes. An invalid input will cause the total cost textbox to display “NaN”. It is implemented in JavaScript as follows:

```
if(numberOfApple.match(/[0-9]+$/) == null && numberOfApple != ""){
  alert("Invalid input at Order Apple. Please provide a valid integer.");
  totalCost.value = NaN;
```

Figure 4.1: Setting the total cost value to NaN when the input is invalid

```
<input type="text" name="numberOfApple" id="numberOfApple" placeholder="Enter quantity here" onchange =
"checkValidInput(); updateTotalCost();" >
```

Figure 4.2: Updating the total cost to display if the input is valid

By including **onchange = “checkValidInput(); updateTotalCost();”** in the input tag, whenever there is a change in the input field, it will check if the input is valid by executing **checkValidInput()** . If the method returns **true**, then it will proceed to calculate the total cost and display it in the textbox. Otherwise (i.e returning **false**), it will just display “NaN” without executing **updateTotalCost()** .

Orange Price	:	59 cents each
Order Orange		<input type="text" value="1"/>
Banana Price	:	39 cents each
Order Banana		<input type="text" value="3b"/>
Total Cost in dollar		<input type="text" value="NaN"/>
Payment method		<input checked="" type="radio"/> Visa <input type="radio"/> MasterCard <input type="radio"/> Discover
<input type="button" value="Reset"/> <input type="button" value="Submit"/>		

Figure 4.3: The total cost textbox displaying NaN due to having invalid input at the banana quantity textbox

Implementation of the payment mode using radio buttons

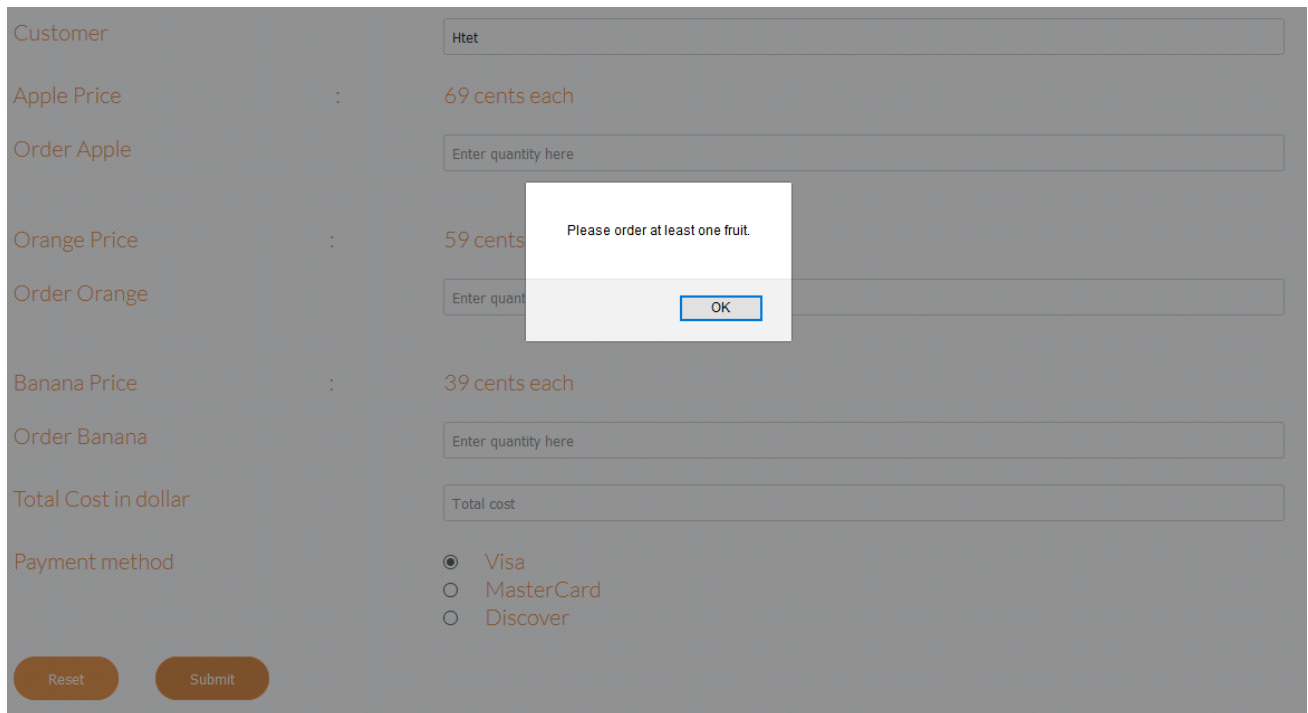
Three radio buttons are used for the user to select their preferred mode of payment (it can also be seen in Fig. 4.3). It is implemented in html document as follows:

```
<div class = "payment-method">
  <div class="col span-1-of-3">
    <label>Payment method</label>
  </div>
  <div class="col span-2-of-3">
    <input type="radio" name="paymentMode" value="Visa" checked="checked" />
    &emsp;Visa
    <br />
    <input type="radio" name="paymentMode" value="MasterCard" />
    &emsp;MasterCard
    <br />
    <input type="radio" name="paymentMode" value="Discover" />
    &emsp;Discover
    <br />
  </div>
</div>
```

Figure 5: Implementing radio buttons for mode of payment

The important thing to take note is specifying the same name for all radio buttons, namely “**paymentMode**” in the input tags. It will help us ensure that only one radio button is selected at one time. A user is not able to choose more than one mode of payment because of this implementation.

Implementation and validation of the order form submission



The screenshot shows a web form for ordering fruit. The form includes fields for Customer (Htet), Apple Price (69 cents each), Order Apple (Enter quantity here), Orange Price (59 cents), Order Orange (Enter quantity here), Banana Price (39 cents each), Order Banana (Enter quantity here), Total Cost in dollar (Total cost), and Payment method (radio buttons for Visa, MasterCard, Discover). At the bottom are Reset and Submit buttons. A modal alert box is displayed in the center with the message "Please order at least one fruit." and an OK button. The form is partially obscured by the alert box.

Figure 6: Submitting the order form without any order

When a customer attempts submit the order form without specifying any quantity for any fruits, an alert message will be displayed (as seen in Fig. 6). Moreover, if a user attempts to submit the form with invalid input present in the textboxes, an alert message will also be prompted and asked user to correct his/her input. The code implementation for these features can be seen below.

```
function verifyBeforeSubmission() {  
    return(checkValidInput() && checkValidOrder())  
}  
  
//the method below checks if a user is trying to submit the form without any fruit order  
function checkValidOrder() {  
  
    var totalCost = document.getElementById("totalCost").value;  
    if (totalCost == "") {  
        alert("Please order at least one fruit.");  
        return false;  
    } else if (totalCost == NaN || totalCost == 0) {  
        alert("Please enter valid entries (i.e. only integer) in order form.");  
        return false;  
    }  
    return true;  
}
```

Figure 6.1: Ensuring all the inputs are valid and correct before submission.

```

<div class="row">
  <div class="col">
    <input type="reset" value="Reset">
  </div>

  <div class="col">
    <input type="submit" value="Submit" onclick="return verifyBeforeSubmission();">
  </div>
</div>

```

Figure 6.2: Verifying every input is correct before submission

The **onclick="return verifyBeforeSubmission()"** in the submit input tag helps us ensure that the Javascript method `verifyBeforeSubmission()` is called before submitting the form to the web server. If the method returns false, it will not be submitted to the server yet, and instead will display corresponding alert messages to the user. Furthermore, the **Reset button** is implemented using the **"reset"** input tag to enable the user to clear all inputs.

Writing a server-side PHP program to receive the user's order

Web UI Design of the order receipt

Order Receipt

Customer:	Htet Naing		
Total Fruits Ordered:	Apple(69 cents each)	Orange(59 cents each)	Banana(39 cents each)
	10	10	10
Unit Cost:	\$ 6.90	\$ 5.90	\$ 3.90
Total Cost:	\$ 16.70		
Payment Mode:	MasterCard		

Figure 7: Order Receipt

Implementation of retrieving inputs from HTML

After an order form is successfully submitted, all the user inputs will be retrieved via a server-side PHP program. The program will calculate the total cost of the user's order and will generate an order receipt to the user in the form of an HTML document. The code implementation can be seen below:

```
<?php
/* Retrieving the form inputs */

// Customer's name
$customer = $_POST["customer"];

// Quantity of apple, orange and banana
$apples = $_POST["numberOfApple"];
$orange = $_POST["numberOfOrange"];
$banana = $_POST["numberOfBanana"];

// if any one of the fruits is not ordered, the quantity will be set to 0
if ($apples == "") {
    $apples = 0;
}
if ($orange == "") {
    $orange = 0;
}
if ($banana == "") {
    $banana = 0;
}

// Payment mode
$payment = $_POST["paymentMode"];

// calculate the cost of each fruit
$apples_cost = 0.69 * $apples;
$orange_cost = 0.59 * $orange;
$banana_cost = 0.39 * $banana;

// calculate the total cost
$total_cost = $apples_cost + $orange_cost + $banana_cost;

/* Produce order receipt */
?>
```

Figure 8: Implementing the input retrieval from HTML document

First, quantity of each fruit for an order is collected using (for instance) “`$_POST["numberOfApple"]`”. The retrieved value is stored in corresponding variables. If there is no quantity for a particular type of fruit, the value of the variable will be set to zero. After that, the total value will be calculated accordingly.

Generating an order receipt in the form of a HTML document

After collecting all the inputs and calculating the necessary values, the receipt will be generated in HTML format. The code implementation for printing all the necessary data to the HTML can be seen below:

```
<tr>
  <td><?php print ("apple"); ?></td>
  <td><?php print ("orange"); ?></td>
  <td><?php print ("banana"); ?></td>
</tr>
```

Figure 9: Printing number of fruits to the HTML document

```
<tr>
  <td><?php printf (" $ %4.2f", $apple_cost); ?></td>
  <td><?php printf (" $ %4.2f", $orange_cost); ?></td>
  <td><?php printf (" $ %4.2f", $banana_cost); ?></td>
</tr>

<!-- Total Cost Information -->
<tr>
  <th rowspan="2">Total Cost:</th>
</tr>
```

Figure 9.1: Printing the cost for each type of fruit order up to 2 decimal places

```
<td colspan="3"><?php printf (" $ %5.2f", $total_cost); ?></td>
r>

- Payment Mode Information ->
>
<th rowspan="2">Payment Mode: </th>
r>
>
<td colspan="3"><?php print ("payment"); ?></td>
```

Figure 9.2: Printing the total cost of the order up to 2 decimal places as well as the payment mode.

Implementation of the update file “order.txt”

In order to update an “order.txt” file, firstly it should be created in the directory. Therefore, the code below checks if the file has already existed and if so, it will proceed to open the file.

“**file_exists(\$filename)**” helps perform the check. The **fopen(\$filename, ‘r’)** where **r** indicates “**read only**” will be executed if the file is already present. “**or exit**” helps ensure that it exits the script if the file cannot be opened for some reasons. If “**file_exists(\$filename)**” returns false, a new txt file will be created with the assistance of the code “**file_put_contents(\$filename, \$output);**” .

```
// create a new order.txt if it does not exist yet
if (!file_exists($filename)) {
    $output = "Total number of apples: ".$apple."\r\nTotal number of oranges: ".$orange."\r\nTotal number of bananas: ".$banana."\r\n";
    file_put_contents($filename, $output);
} else {
    $file = fopen($filename, 'r') or exit ("unable to open file ($filename)");
```

Figure 10: Checking if the order.txt file has already existed

The purpose of the “order.txt” file is to keep track of the number of fruits ordered for each type of fruits regardless of the purchaser. Therefore, whenever orders are placed, the new quantity will be added to the old quantity for each fruit category. The following code is implemented to achieve that goal.

```
for ($x = 0; !feof($file); ++$x) {
    $tmpLine = fgets($file);
    preg_match("/\d+/", $tmpLine, $matches); // matching the digit from each line

    // Updating the total number for each type of fruit
    switch($x) {
        case 0:
            $output .= preg_replace("/\d+/", $matches[0] + $apple, $tmpLine);
            break;
        case 1:
            $output .= preg_replace("/\d+/", $matches[0] + $orange, $tmpLine);
            break;
        case 2:
            $output .= preg_replace("/\d+/", $matches[0] + $banana, $tmpLine);
            break;
    }
}
fclose($file);
file_put_contents($filename, $output);
```

Figure 10.1: Updating the text file after opening

In the for loop, the existing txt file will be read one line after another. The existing quantity will be searched with the help of “/\d+/. When the search is successful, the values found will be stored in the matches array correspondingly. The new quantity which are currently stored in variables “\$apple, \$orange and \$banana” will be added to the existing quantity (which is now present in the matches array). Eventually, the “preg_replace” method will carry out updating the new sum for each line of the txt file.

“fclose(\$file)” will close the txt file, and then the PHP program proceeds to put the updated contents back to the “order.txt” file with the assistance of “file_put_contents(\$filename, \$output)”. Finally, it has successfully recorded the total number of orders for each type of fruits in the “order.txt” file.

Listing of source files

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="vendors/css/grid.css">
    <link rel="stylesheet" type="text/css" href="resources/css/style.css">
    <link href='http://fonts.googleapis.com/css?family=Lato:100,300,400,300italic' rel='stylesheet' type='text/css'/>
    <title>Online Fruit Shop</title>
  </head>
  <body>
    <header>
      <div class="main-title-text-box">
        <h1>BoA: Online Fruit Shop</h1>
        <br>
        <br>
        <h2>Keep calm and eat healthy</h2>
      </div>
    </header>

    <section class="greeting-message" id="features">
      <div class="row">
        <h2>Welcome to Our Online Fruit Shop</h2>
        <p class="long-copy">
          Hi there! Would you like to order some fruits? We have the best apples, oranges and bananas in town!
        </p>
      </div>
    </section>

    <section class="section-order-form" id="orderForm">
      <div class="row">
        <h2>Fruit Order Form</h2>
      </div>

      <div class="row">
        <form method="post" action="receipt.php" class="order-form">

          <div class="row">
            <div class="col span-1-of-3">
              <label for="name">Customer</label>
            </div>
            <div class="col span-2-of-3">
              <input type="text" name="customer" id="customer" placeholder="Name" required>
            </div>
          </div>

          <div class = "apple-row">
            <div class="col span-1-of-3">
              <label>Apple Price &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~::~/>
            </div>
            <div class="col span-2-of-3">
              <label>69 cents each</label>
            </div>

            <div class="row">
              <div class="col span-1-of-3">
                <label>Order Apple</label>
              </div>
              <div class="col span-2-of-3">
                <input type="text" name="numberOfApple" id="numberOfApple" placeholder="Enter quantity here"
                  onchange = "checkValidInput(); updateTotalCost();" >
              </div>
            </div>
          </div>

          <br>

          <div class = "orange-row">
            <div class="col span-1-of-3">
              <label>Orange Price &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~::~/>
            </div>
            <div class="col span-2-of-3">
              <label>59 cents each</label>
            </div>

            <div class="row">
              <div class="col span-1-of-3">
                <label>Order Orange</label>
              </div>
            </div>
          </div>
        </form>
      </div>
    </section>
  </body>
</html>
```



```

var numberOfBanana = document.getElementById("numberOfBanana").value;
var totalCost = document.getElementById("totalCost");

//checking if the input to order apple is valid (i.e. only contain digits)
if(numberOfApple.match(/[0-9]+$/) == null && numberOfApple != ""){
    alert("Invalid input at Order Apple. Please provide a valid integer.");
    totalCost.value = NaN;
    document.getElementById("numberOfApple").focus();
    document.getElementById("numberOfApple").select();

//checking if the input to order orange is valid (i.e. only contain digits)
}else if(numberOfOrange.match(/[0-9]+$/) == null && numberOfOrange != ""){
    alert("Invalid input at Order Orange. Please provide a valid integer.");
    totalCost.value = NaN;
    document.getElementById("numberOfOrange").focus();
    document.getElementById("numberOfOrange").select();

//checking if the input to order banana is valid (i.e. only contain digits)
}else if(numberOfBanana.match(/[0-9]+$/) == null && numberOfBanana != ""){
    alert("Invalid input at Order Banana. Please provide a valid integer.");
    totalCost.value = NaN;
    document.getElementById("numberOfBanana").focus();
    document.getElementById("numberOfBanana").select();
}else{
    //calculate the total cost if all inputs are valid
    return true;
}

return false;
}

//the method below calculate the total cost dynamically whenever any of the input changes
function updateTotalCost(){
    var numberOfApple = document.getElementById("numberOfApple").value;
    var numberOfOrange = document.getElementById("numberOfOrange").value;
    var numberOfBanana = document.getElementById("numberOfBanana").value;
    var totalCost = document.getElementById("totalCost");

    var appleCost = 0;
    var orangeCost = 0;
    var bananaCost = 0;

    appleCost = 0.69 * numberOfApple;
    orangeCost = 0.59 * numberOfOrange;
    bananaCost = 0.39 * numberOfBanana;

    var totalSum = appleCost + orangeCost + bananaCost;
    totalCost.value = totalSum.toFixed(2);
}

function verifyBeforeSubmission(){
    return(checkValidInput() && checkValidOrder())
}

//the method below checks if a user is trying to submit the form without any fruit order
function checkValidOrder(){
    var totalCost = document.getElementById("totalCost").value;
    if (totalCost == ""){
        alert("Please order at least one fruit.");
        return false;
    }else if(totalCost == NaN || totalCost == 0){
        alert("Please enter valid entries (i.e. only integer) in order form.")
        return false;
    }
    return true;
}
</script>

</body>

</html>

```


receipt.php

```
<?php
/* Retrieving the form inputs */

// Customer's name
$customer = $_POST["customer"];

// Quantity of apple, orange and banana
$apple = $_POST["numberOfApple"];
$orange = $_POST["numberOfOrange"];
$banana = $_POST["numberOfBanana"];

// if any one of the fruits is not ordered, the quantity will be set to 0
if ($apple == "") {
    $apple = 0;
}
if ($orange == "") {
    $orange = 0;
}
if ($banana == "") {
    $banana = 0;
}

// Payment mode
$payment = $_POST["paymentMode"];

// calculate the cost of each fruit
$apple_cost = 0.69 * $apple;
$orange_cost = 0.59 * $orange;
$banana_cost = 0.39 * $banana;

// calculate the total cost
$total_cost = $apple_cost + $orange_cost + $banana_cost;

/* Produce order receipt */
?>
<html lang="en">

    <head>
        <!-- Basic setup for CSS files ---->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" type="text/css" href="vendors/css/grid.css">
        <link rel="stylesheet" type="text/css" href="resources/css/style.css">
        <link href='http://fonts.googleapis.com/css?family=Lato:100,300,400,300italic' rel='stylesheet' type='text/css'>
    </head>

    <body>
        <section class="section-order-receipt" id="receiptForm">
            <div class="col">

                </div>

            <div class="col">
                <h2>Order Receipt</h2>
                <div class="row">
                    <table>
                        <!-- Customer Information -->
                        <tr>
                            <th rowspan="2">Customer:</th>
                        </tr>
                        <tr>
                            <td colspan="3"><?php print ("{$customer}"); ?></td>
                        </tr>

                        <!-- Fruits Info -->
                        <tr>
                            <th rowspan="2">Total Fruits Ordered:</th>
                            <th>Apple (69 cents each)</th>
                            <th>Orange (59 cents each)</th>
                            <th>Banana (39 cents each)</th>
                        </tr>
                        <tr>
                            <td><?php print ("{$apple}"); ?></td>
                            <td><?php print ("{$orange}"); ?></td>
                            <td><?php print ("{$banana}"); ?></td>
                        </tr>

                        <tr>
                            <th rowspan="2">Unit Cost:</th>
                        </tr>
                        <tr>
                            <td><?php printf ("$ %4.2f", $apple_cost); ?></td>
                            <td><?php printf ("$ %4.2f", $orange_cost); ?></td>
                            <td><?php printf ("$ %4.2f", $banana_cost); ?></td>
                        </tr>

                        <!-- Total Cost Information -->
                        <tr>
                            <th rowspan="2">Total Cost:</th>
```

```

        </tr>

        <tr>
            <td colspan="3"><?php printf ("%5.2f", $total_cost); ?></td>
        </tr>

        <!-- Payment Mode Information -->
        <tr>
            <th rowspan="2">Payment Mode: </th>
        </tr>
        <tr>
            <td colspan="3"><?php print ("{$payment}"); ?></td>
        </tr>
    </table>
</div>
</div>
</section>
</body>
</html>

<?php
/* Updating order.txt */
$filename = 'order.txt';

// create a new order.txt if it does not exist yet
if (!file_exists($filename)) {
    $output = "Total number of apples: ".$apple."\nTotal number of oranges: ".$orange."\nTotal number of bananas:
".$banana."\n";
    file_put_contents($filename, $output);
} else {
    $file = fopen($filename, 'r') or exit ("unable to open file ($filename)");
    for ($x = 0; !feof($file); ++$x) {
        $tmpLine = fgets($file);
        preg_match("/\d+/", $tmpLine, $matches); // matching the digit from each line

        // Updating the total number for each type of fruit
        switch($x) {
            case 0:
                $output .= preg_replace("/\d+/", $matches[0] + $apple, $tmpLine);
                break;
            case 1:
                $output .= preg_replace("/\d+/", $matches[0] + $orange, $tmpLine);
                break;
            case 2:
                $output .= preg_replace("/\d+/", $matches[0] + $banana, $tmpLine);
                break;
        }
    }
    fclose($file);
    file_put_contents($filename, $output);
}
?>

<!-- table style --->
<style>
    table {
        font-family: 'Lato';
        border-collapse: collapse;
    }

    td, th {
        font-family: 'Lato';
        font-size: 12pt;
        padding: 15px;
        color: black;
        border: 2px solid #e67e22;
        text-align: center;
    }
}
</style>

```

style.css (Its directory is “resources/css/style.css” in the submitted folder)

```
/*
Orange color: #e67e22
Light Orange color: #cf6d17;
Maroon color: #c0392b
Light Maroon color: #e74c3c
*/

/* ----- */
/* BASIC SETUP */
/* ----- */

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

html,
body {
  background-color: #fff;
  color: #555;
  font-family: 'Lato', 'Arial', sans-serif;
  font-weight: 300;
  font-size: 20px;
  text-rendering: optimizeLegibility;
  overflow-x: hidden;
}

.clearfix {zoom: 1;}
.clearfix:after {
  content: '.';
  clear: both;
  display: block;
  height: 0;
  visibility: hidden;
}

/* ----- */
/* REUSABLE COMPONENTS */
/* ----- */

.row {
  max-width: 1140px;
  margin: 0 auto;
}

section {
  padding: 80px 0;
}

.box {
  padding: 1%;
}

/* ----- HEADINGS ----- */

h1,
h3 {
  font-weight: 300;
}

h1 {
  margin-top: 0;
  margin-bottom: 20px;
  color: #fff;
  font-size: 240%;
  word-spacing: 4px;
  letter-spacing: 1px;
  text-align: center;
}

h2 {
  font-weight: 300;
  font-size: 150%;
  word-spacing: 2px;
  text-align: center;
  margin-bottom: 30px;
  letter-spacing: 1px;
  color: #fff;
  color: #e67e22;
}

h3 {
  font-size: 110%;
  margin-bottom: 15px;
}
```

```

h2:after {
    display: block;
    height: 2px;
    background-color: #e67e22;
    content: " ";
    width: 100px;
    margin: 0 auto;
    margin-top: 30px;
}

/* ----- PARAGRAPHS ----- */

.box p {
    font-size: 90%;
    line-height: 145%;
}

/* ----- BUTTONS ----- */

input[type=submit],
input[type=reset]{
    display: inline-block;
    padding: 10px 30px;
    font-weight: 300;
    text-decoration: none;
    border-radius: 200px;
    -webkit-transition: background-color 0.2s, border 0.2s, color 0.2s;
    transition: background-color 0.2s, border 0.2s, color 0.2s;
}

input[type=submit],
input[type=reset]{
    background-color: #e67e22;
    border: 1px solid #e67e22;
    color: #fff;
    margin-right: 15px;
}

input[type=submit]:hover,
input[type=submit]:active,
input[type=reset]:hover,
input[type=reset]:active{
    background-color: #cf6d17;
}

/* ----- */
/* HEADER */
/* ----- */

header {
    background-image: -webkit-linear-gradient(rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 0.5)), url(img/fruits.jpg);
    background-image: linear-gradient(rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 0.5)), url(img/fruits.jpg);
    background-size: cover;
    background-position: center;
    height: 100vh;
    background-attachment: fixed;
}

.main-title-text-box {
    position: absolute;
    width: 1140px;
    top: 50%;
    left: 50%;
    -webkit-transform: translate(-50%, -50%);
    -ms-transform: translate(-50%, -50%);
    transform: translate(-50%, -50%);
}

/* ----- */
/* Greeting Message */
/* ----- */

.greeting-message .long-copy{
    margin-bottom: 30px;
    text-align: center;
}

.greeting-message{
    background-color: #f4f4f4;
}

/* ----- */
/* ORDER FORM */
/* ----- */

input[type=text],
select{
    width: 100%;
}

```

```

padding: 7px;
border-radius: 3px;
border: 1px solid #ccc;
}

.section-order-form{
color: #e67e22;
}

input[type=checkbox] {
margin: 10px 5px 10px 0;
}

*:focus {outline: none;}

/* ----- */
/* FOOTER */
/* ----- */

footer {
background-color: #333;
padding: 50px;
font-size: 80%;
}

footer p {
color: #888;
text-align: center;
margin-top: 20px;
}

```

grid.css (Its directory is “vendors/css/grid.css” in the submitted folder)

Disclaimer: grid.css is an open source css file. Since it is not self-written code, the code will not be shown here. However, **when assessing this assignment, it is highly recommended to place it together with other files such as index.html, receipt.php and style.css.**

It is available free online and is downloaded from <http://www.responsivegridsystem.com/> . No copyright infringement is intended. The purpose of grid.css is to help web developers build a responsive website whose UI stays consistent and beautiful across all devices and platforms without any issues.

Appendix

The free-of-charge or open-source contents from the following websites are referred or used when working on this assignment. Thus, I shall give full credits to all the original creators and authors for creating awesome works that are publicly shared for reuse.

- 1) <http://www.wallpapermania.eu/wallpaper/glass-fruits-banana-apple-and-orange-abstract-wallpaper>
- 2) <https://flatuicolors.com/>
- 3) <https://www.w3schools.com/>
- 4) <http://www.responsivegridsystem.com/>
- 5) <http://codingheroes.io/resources/>
- 6) <https://stackoverflow.com/>