Advanced Higher Computing Project

Online Order Taking Service

Kenneth Wong

**Table of Contents**

Analysis 4

Project Proposal 4

Problem 4

Details 4

Project Limits 5

System Specification 6

Boundaries 6

Functional Requirements 6

Project Plan 7

It must be noted that some of these subtasks were done concurrently as shown by the Gantt chart so the total time of 40 hrs allowed for the project was not exceeded as it could be assumed from the completion table.Resource List 8

Resource List 9

Hardware 9

Software 9

Other 9

Selection of a Strategy 10

Comparing and Choosing a Suitable Language and Environment 10

Design 12

Pseudo code for taking the order 12

Pseudo code for checking order 15

User Interface Design 16

Menu displayed in tabular form 16

Design for order output 16

Top down design 17

Implementation 18

Taking Order 18

Display Menu 18

Get customer details 27

Write meal order data to databases 30

Checking Order 33

Display order details 33

Testing 37

Test Plan 37

Sub-programs testing 37

Functional requirements testing 37

Systematic and Comprehensive Testing 38

Making an order 38

Beta Testing 40

The Customers 40

The Company 40

Errors Discovered 40

Documentation 42

User Guide 42

Software features 42

Using the system 42

FAQ and Troubleshooting 44

Technical Guide 45

Software requirements 45

Hardware requirements 45

List of files 45

Evaluation 46

Evaluation of system 46

Fitness for purpose 46

User interface 46

Robustness 46

Reliability 47

Portability 47

Efficiency 47

Maintainability 48

Evaluation of process skills 48

Time management 48

Alternative Strategies 48

Appendix 49

Full Program Listing 49

# Analysis

## Project Proposal

### Problem

A service is needed which will allow the ordering of meals on the school network for a small takeaway company, potentially adaptable for the World Wide Web.

This system will aid the company and the customer, making for automated and more efficient order taking for the company, and a faster, straightforward and more accessible way for the customer to make an order.

This should create a better service for the customer and in turn more profit for the company.

### Details

The system will allow a customer to order a meal by accessing the company’s website. The company should be able to receive the orders onto their local server and view them via the same website.

This must be able to be run on existing hardware in school. Ideally the solution should be able to be accessed from any computer in school and potentially the world; a client-server model can be used to achieve this.

The system should be able to read a text file containing the details of the menu and display this to the customer in a tabular format.

The system should be able to take standard orders, including names, phone numbers and addresses and present all information to the company.

The system should make use of databases to store this information and be able to write records to the database, and retrieve records from the database.

The company should be able to access any order, print them and update their status. The customer should also be able to access the details of their own order and check the status of their order.

### Project Limits

The system will not allow items on the menu text file to be ordered into different categories such as starters and main meals, but all items under one list of meals and prices.

The system will not keep a record of stock available and update it depending on what has been ordered.

The web page interface will not be made particularly attractive, just enough functionality to be easily usable.

The system will not check for real users through methods such as the use of Captchas to ensure it is not a spam program making orders.

The payment of the orders will not be available to be made online.

Customer details will not be encrypted prior to writing to database.

## System Specification

### Boundaries

The system will be completed when:

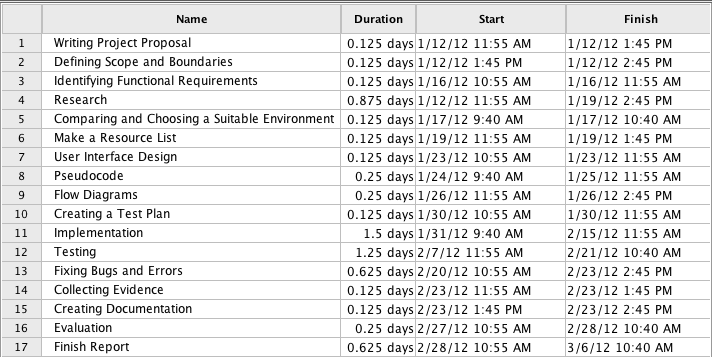
* It contains and generates suitable web pages to provide information and allow input to order a meal on the school network.
* It is able to read the contents of a text file and create a web page that displays the contents of the file for the menu
* It uses of a relational databases to store orders and customer details.
* There is a method where the customer can check their order and its state and the company can easily update this state.

### Functional Requirements

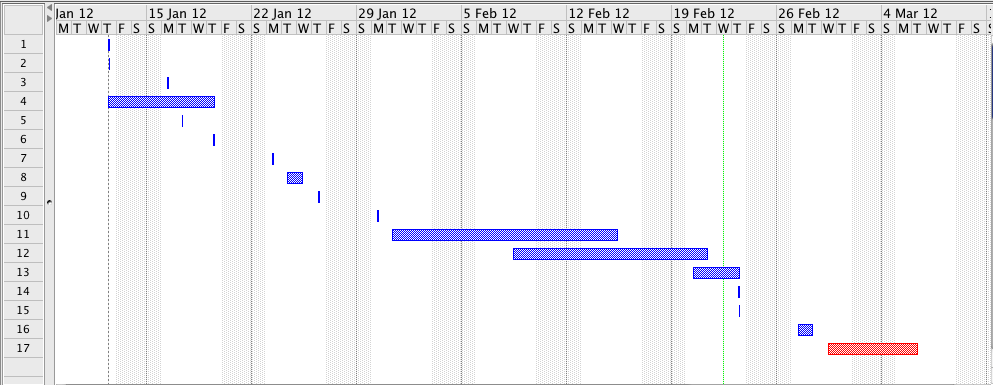
* Accessible website.
* All options on website available as clickable buttons or links on User input for the quantities of orders, and customer details screen.
* User input for the quantities of orders, and customer details.
* Validation of some inputs.
* User ability to track order on website.
* Relational database to store orders.
* Company ability to print these orders.
* Company can update the state of an order on a database which will be able to be checked on the website.

### Project Plan

Timetable:



Gantt chart:



Completion table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sub Task** | **Time Allocated** | **Time Taken** | **Comment** |
| **Writing Project Proposal** | 1 hr | 1 hr |  |
| **Defining Scope and Boundaries** | 1 hr | 1 hr |  |
| **Identifying Functional Requirements** | 1 hr | 1 hr |  |
| **Research** | 5 hrs | 7 hrs | Took slightly longer to learn PHP and MySQL basics |
| **Comparing and Choosing a Suitable Environment** | 1 hr | 0.5 hr | Chosen quickly due to availability of only one truly suitable environment |
| **Make a Resource List** | 1 hr | 0.5 hr | Not much had to be considered |
| **User Interface Design** | 1 hr | 1 hr |  |
| **Pseudo code** | 2 hrs | 2 hr |  |
| **Flow Diagrams** | 2 hrs | 1 hr | Use of Inspiration sped up process |
| **Creating a Test Plan** | 1 hr | 1 hr |  |
| **Implementation** | 12 hrs | 15 hrs | Due to non familiarity with languages used, implementation took longer than planned |
| **Testing** | 10 hrs | 6 hrs | Most of the testing was done during implementation so only beta testing and some module testing needed to be done |
| **Fixing Bugs and Errors** | 4 hrs | 2 hrs | Not many found |
| **Collecting Evidence** | 1 hr | 1 hr |  |
| **Creating Documentation** | 1 hr | 1 hr |  |
| **Evaluation** | 2 hrs | 2 hrs |  |
| **Finish Report** | 4 hrs | 6 hrs |  |
| **Total Hours:** | 50 hrs | 49 hrs |  |

## It must be noted that some of these subtasks were done concurrently as shown by the Gantt chart so the total time of 40 hrs allowed for the project was not exceeded as it could be assumed from the completion table.Resource List

### Hardware

* Web Server
* Database Server
* Computer on the network (all on the school network)

### Software

* Netbeans IDE 6.8
* Preferably PHPMyAdmin however not necessary
* Web browser such as Mozilla Firefox
* Word processor such as Microsoft Word

### Other

* Books on learning PHP and MySQL

I have all of these resources readily available at school.

## Selection of a Strategy

### Comparing and Choosing a Suitable Language and Environment

|  |  |  |  |
| --- | --- | --- | --- |
| **Language** | **Java** | **PHP** | **ASP.NET** |
| **Advantages** | Object-Oriented, Portable due to JVM, Automatic memory management, High level data constructs, extensive module libraries | Designed for dynamic web content, run server side so nothing needs to be installed on client, very flexible, not limited to dynamically generating HTML, wide ranging support for databases which makes it easy to store and access data, loosely typed which allows for faster development. | ASP.NET improves performance by utilizing the strengths of JIT compilation, automatic resource optimization, runtime profiling, automatic memory management, early binding, better caching and exception handling, inbuilt data-caching module. |
| **Disadvantages** | Memory overhead of JVM makes it less efficient, More difficult to implement in a web based environment due to security issues in a Java Applet | Loosely typed language which may create problems during implementation, executes noticeably more slowly than other languages such as Java | ASP does not have platform compatibility like PHP. This is because it is mostly dependent on Microsoft products and needs a Windows platform along with an ASP-Apache installation at the server side.  ASP is slower than PHP, as it is based on COM architecture that is actually an overhead to the server. |
| **Previous Experience** | I have some experience coding in Java so it would be easier and less effort for me to code in. | I have no experience with PHP so time will have to be spent learning the language. | I have no experience with ASP so time will have to be spent learning the language. |

Final Decision: PHP was decided to be the most viable option due to its optimization as a web development language with libraries and help designed around common web functions, and its ability to integrate with MySQL to store data in databases. Java was not used as Java applets are very inefficient ways to program for the web, and requires the client to have installed a JVM, ASP was not chosen due to its proprietary nature and the computer on which the solution is to be created is a Macintosh running on MAC OS X.

|  |  |  |
| --- | --- | --- |
| **Environment** | **Netbeans** | **Eclipse** |
| **Advantages** | Supports code folding, code completion, navigation, mark occurrences, bookmarks, configurable syntax coloring, task tags, code templates, faster code completion, code formatting for PHP, easy to test PHP code with integrated support for PHPUnit | Very configurable, supports code folding, code completion, navigation, mark occurrences, bookmarks, configurable syntax coloring, task tags, code templates, many plug-ins available for all sorts of tasks |
| **Disadvantages** | Slower performance compared to Eclipse | No code formatting for PHP |

Final Decision: Netbeans was chosen mainly due to the fact that it was already installed on the machines at school whereas Eclipse was not, as they are both very good environments for developing in PHP. However Netbeans was also chosen due to the ability to format code for PHP.

# Design

## Pseudo code for taking the order

Display menu (in: menu file; out: meal array, price array)

Read contents of menu file to array

Open menu file

Read menu file to string array

FOR every section separated by a ‘|’ character in the menu file

Add that section as item to array

END FOR

Split array data to price array and meal array

FOR every alternate item in array starting at index 0

Add current item to meal array

END FOR

FOR every alternate item in array starting at index 1

Add current item to meal array

END FOR

Output menu

FOR number of items in meal array

Output item of meal array at current index, and price array at same index to a row in a table

END FOR

Get meal order data (in: POST data from webpage about order; out: 2D array of meals and quantities)

Prompt user to input quantity of meals into a menu

Assign meal order data to 2D-array

FOR each meal in menu

IF quantity ordered is not less than 1

Add meal and quantity to 2D array

END IF

END FOR

Display meal order data

FOR number of meals in 2D array

Display meal at index, quantity at index and price at the meal of the menu

END FOR

Calculate grand total

FOR number of meals in 2D array

Multiply quantity of meal by its price

Add to grand total

END FOR

Display grand total

Give customer a reference number (out: reference number)

Randomly generate a 10-digit number

Randomly choose a number between 1000000000 and 9999999999

Validate the number by checking it against all other reference numbers

Connect to database

Get all customer records from customer table

FOR all customer records

IF new reference ID equals any ID stored

Generate new 10-digit number

END IF

END FOR

Close connection to database

Display number

Get customer details (in: POST data from a webpage about customer details; out: name, address, phone number)

Prompt user to enter name

Prompt user to enter phone number

Validate phone number as it is being typed

FOR each character being entered

IF number of characters typed > 11

Do not allow character to be entered

END IF

IF character entered is not a number

Do not allow character to be entered

END IF

Write meal order data and customer details to record in a table in a database (in: 2D array of meals, name, address, phone number, reference number; out: record of meal in database, record of customer details in database)

Connect to database

FOR each meal in 2D array of meals

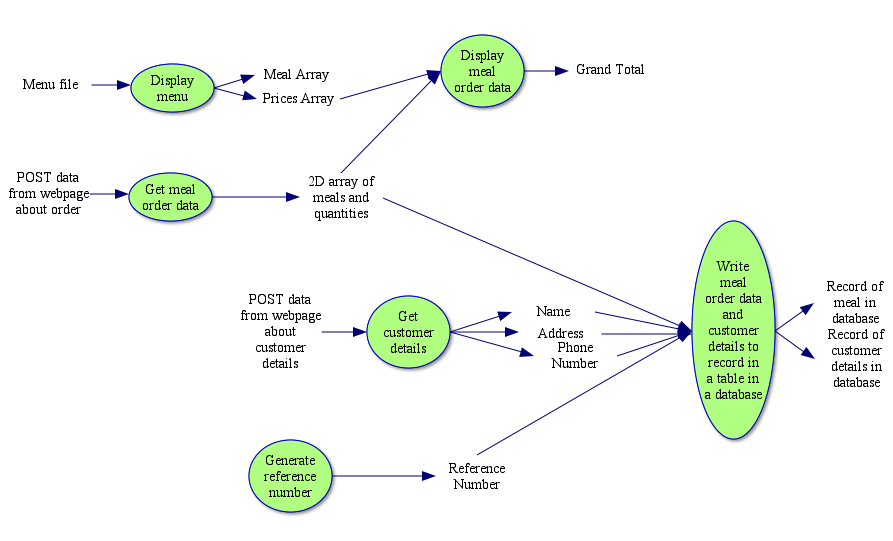
Insert the reference number, name of meal and quantity of meal to a record in the orders table in the database

END FOR

Insert reference number, name, phone number and address to a record in the customer table in the database

Insert the state ‘pending’ into a record in the order states table in the database.

Close connection to database

****

## Pseudo code for checking order

Prompt user to enter their reference number (out: customer reference number)

Check the order details (in: customer reference number; out: customer details record, customer orders record)

Connect to database

Select the correct details from the customer table and the orders table

FOR number of records in customer table

IF reference number in table = customer reference number

Select current record

END IF  
END FOR

FOR number of records in orders table

IF reference number in table = customer reference number

Select current record

END IF  
END FOR

Close connection to database

Display order and details (in: customer details record, customer orders records)

Display name, address and phone number and state from customer details record

Display order from customer orders records

FOR EACH customer order record

Output quantity, item name and total price

Add total price to grand total

Output grand total

END FOR

****

## User Interface Design

The user interface will be delivered via a web browser, and written in HTML. In a normal case, the web pages for a site that provides such a service should usually be visually attractive, however the aim of this specific project is to develop the underlying system that runs the order taking and due to time constraints, not much attention will be given to a particularly pleasing user interface.

The user interface will be a menu driven interface consisting of text, buttons and links, and input text boxes for all inputs. Some data will be displayed in a tabular format, and specific layouts will be designed for the displaying of orders and customer details to the company.

### Menu displayed in tabular form

|  |  |
| --- | --- |
| **Item** | **Price** |
| Meal | £1.00 |
| Meal | £1.00 |
| Meal | £1.00 |
| Meal | £1.00 |
| Meal | £1.00 |
| Meal | £1.00 |
| Meal | £1.00 |
| Meal | £1.00 |
| Meal | £1.00 |

### Design for order output

**Name:** John Smith  
**Phone:** 01234567890  
**Address:** 1 Where I live  
**ReferenceID:** 9876543210

|  |  |  |
| --- | --- | --- |
| **Quantity** | **Item** | **Total Price** |
| 2 | Meal1 | £2.00 |
| 1 | Meal2 | £1.00 |
| 2 | Meal3 | £2.00 |
|  | **Grand Total:** | £5.00 |

## Top down design

****

# Implementation

## Taking Order

### Display Menu

Problem: Data about the menu items and prices needs to be read from a text file and displayed dynamically in a tabular format.

Considerations: The format of the text in the text file is important, how will the program differentiate between items and prices from the data in the file, and separate them into different variables or store them in different arrays.

Solution: It was decided to use a text file to store the menu data so it could be easily updated or changed, instead of hard coding the data into the web pages. The format of the text file will be the name of a meal followed by a ‘|’ character, followed by the meal’s price, followed by another ‘|’ and repeated for all items. This is due to the use of the explode function in PHP, which uses a separator (in this case the ‘|’ character) to split text into separate items in an array. This will allow the meal names and prices to be read and used separately.

This part was decided to be created as an ‘inc.php’ file as many different areas of the system required the data in the menu to be read and used, so using a separate file for it meant that it could be easily included in any other php file in the system.

Format of menu.txt:

|  |
| --- |
| MENU  |Chips|1.40  |Chips and Curry Sauce|6.10  |Chips and Sweet and Sour Sauce|4.00  |Chips and Cheese|3.90  |Fried Rice|1.00  |Fried Rice with Curry Sauce|6.10  |Fried Rice and Sweet and Sour Sauce|4.00  |Fried Rice and Cheese|3.80  |Pizza Slice|3.10  |Pizza Slice and Chips|5.10  |Burger|2.30  |Burger and Chips|3.20 |

*Listing*

<?php

//Read menu file and place contents in an item array and a price array

$testFile = file\_get\_contents("menu.txt");

$testStringArray = explode('|', $testFile);

$mealsCounter = 0;

$pricesCounter = 0;

for($loop1 = 1; $loop1 < count($testStringArray)-1; $loop1+=2) {

$mealsArray[$mealsCounter] = $testStringArray[$loop1];

$mealsCounter++;

}

for($loop2 = 2; $loop2 < count($testStringArray); $loop2+=2) {

$pricesArray[$pricesCounter] = $testStringArray[$loop2];

$pricesCounter++;

}

?>

Once the menu was read to the two arrays it was simple to display dynamically to the browser. Using a mix of PHP and HTML we can output the menu into a neat tabular format.

*Listing*

<?php

include("readmenu.inc.php");

function input($meals, $prices) {

for($loop3 = 0; $loop3 < sizeof($prices); $loop3++) {

?>

<tr>

<td><?php echo $meals[$loop3]?></td>

<td>&pound;<?php echo $prices[$loop3]?></td>

</tr>

<?php }

}?>

<body>

<table width="300" border="1" cellspacing="2" cellpadding="0">

<tr>

<td><b>Item</b></td>

<td><b>Price</b></td>

</tr>

<?php input($mealsArray, $pricesArray); ?>

</table>



**Get meal order data**

Problem: To receive an input from the user for the number of items of each meal that they want, and redisplay the order for confirmation to the customer.

Considerations: The input for the number of items of meals the customer wants must be validated to be numeric only or else it will allow wrong orders. There is no pure numeric text input in HTML and handling events is more difficult in PHP so it has been decided to code the numeric only input in JavaScript for ease. Additionally the order details need to be ‘saved’ like a shopping basket for use later on to write to a database.

Solution: To achieve numeric only input we will create a function that can be added to any HTML text input box to check the input whenever there is a key pressed event. If the input is non-numeric, the input will not be allowed. To save the order details for later, a session variable will be used, which stores the data until the session is ended, or until the variable is cleared.

Numeric only input:

*Listing*

<SCRIPT TYPE="text/javascript">

<!--

function numbersonly(field, input)

{

var key;

var keychar;

if (input)

key = input.which;

else

return true;

keychar = String.fromCharCode(key);

// control keys

if ((key==null) || (key==0) || (key==8) ||

(key==9) || (key==13) || (key==27) )

return true;

// numbers

else if ((("1234567890").indexOf(keychar) > -1))

return true;

else

return false;

}

//-->

All that needs to be done to allow the customer to make the order is to display the menu with the numeric input boxes so that the customer can enter a quantity of the meal that they want.

*Listing*

<?php

include("readmenu.inc.php");

function input($meals, $prices) {

for($loop3 = 0; $loop3 < count($prices); $loop3++) { ?>

<tr>

<td><input type="text" maxlength="2" onKeyPress="return numbersonly(this, event)" name="<?php echo $loop3 ?>" size="3"></td>

<td><?php echo $meals[$loop3]?></td>

<td><?php echo '£'.$prices[$loop3] ?></td>

</tr>

<?php }

}

?>

<table width="300" border="1" cellspacing="2" cellpadding="0">

<tr>

<td><b>Quantity</b></td>

<td><b>Item</b></td>

<td><b>Price</b></td>

<?php

input($mealsArray, $pricesArray);

?>

</table>



Now that the order has been made, the order should be validated with the customer to make sure that it is what the customer actually wants, so we redisplay the order back, before writing it to the database (sending company the order). Additionally as previously stated the order must be written to a session variable in order to be used in later web pages.

*Listing*

<?php

//Declaring session variables and starting session

session\_start();

$\_SESSION['order'];

?>

<?php

//Redisplaying the order back to the customer and writing the order to a 2D-array session variable

}

elseif ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$itemCounter = 0;

$numberOfCounter = 0;

for($loop3 = 0; $loop3 < count($mealsArray); $loop3++) {

$numberOfItems[$loop3] = $\_POST[$loop3];

}

echo "You have ordered:<br />";

for($loop4 = 0; $loop4 < count($pricesArray); $loop4++) {

$total += ($pricesArray[$loop4])\*($numberOfItems[$loop4]);

if($numberOfItems[$loop4] != 0) {

$output = "<br />".$mealsArray[$loop4]." x ".$numberOfItems[$loop4];

$orderItems[$itemCounter][$numberOfCounter]=$mealsArray[$loop4];

$orderItems[$itemCounter][$numberOfCounter+1]=$numberOfItems[$loop4];

$orderItems[$itemCounter][$numberOfCounter+2]=($pricesArray[$loop4])\*($numberOfItems[$loop4]);

$itemCounter++;

echo $output;

}

}

$\_SESSION['order'] = $orderItems;

echo "<br /><br />The total price of your purchases is: £".$total;

?>



### Get customer details

Problem: To get the customer name, address and phone number, and to give the customer a reference number which can be used to check the state of their order.

Considerations: The reference number must be a number that is different from any other given reference number and also non-guessable as important customer details can be obtained from the number. The phone number must be numeric only and have a maximum length of 11 characters.

Solution: The phone number problem is exactly the same as the input quantities problem earlier on, so the same JavaScript input validation method can be used to create a numeric only text input box, additionally for the maximum length, only an extra HTML part is needed to define the text box.

*Listing*

Phone: <input type="text" onKeyPress="return numbersonly(this, event)" name="phone" maxlength="11"><br>

The reference number was decided to be a randomly generated 10-digit number, which would be checked against the customer details database to make sure it was not the same as any other reference number before it was allowed.

However a bug was found where when the reference number was input into the database, it automatically changed itself to 2147483648, making all reference numbers entered the same. This was discovered to be due to the maximum number in the table possible was this, which was represented by 31 bits and some numbers were exceeding this so it was changed to the largest possible number.

Changing the reference number to a 9-digit number so that it would never exceed the max solved this problem and still maintained its non guessable nature.

*Listing*

<?php

include("dbinfo.inc.php");

mysql\_connect(localhost,$username,$password);

@mysql\_select\_db($database) or die( "Unable to select database");

$query="SELECT \* FROM customers";

$result=mysql\_query($query);

$num=mysql\_numrows($result);

$i=0;

$isSame = false;

do {

//Generate random 9-digit number

$randID = rand(100000000,999999999);

while ($i < $num) {

$id=mysql\_result($result,$i,"id");

if($randID == $id) {

$isSame = true;

}

$i++;

}

}while($isSame == true);

mysql\_close();

?>

The other inputs required will just be created with standard HTML text input boxes and will not be validated.



### Write meal order data to databases

Problem: To write the meal order data and customer details to records in tables in the database, and create a state record for the order.

Considerations: At first it was thought to have one record for all the details, however this was discovered to be complicated and unachievable due to the different number of fields required dependent on the meals ordered. So it was then decided to have a table for the customer details, a table for the meals, where a record was made for each item type ordered which could be more than 1 per order (i.e. 10 chips and 3 rice would make a record for chips and a record for rice), and a table for the state of the order. Additionally to work with databases, I had to learn basic MySQL but this did not take very long.

Solution: The order data was taken from the 2D-Array saved to the session and the customer details were received from the POST data sent from the webpage that received the customer input. All of this data was organized into records using standard MySQL queries and thus the order would be completed and able to view by the company.

*Listing*

<?php

session\_start();

$order = $\_SESSION['order'];

?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title></title>

</head>

<body>

Thank you for your order <br>

<?php

include("readmenu.inc.php");

include("dbinfo.inc.php");

$orderStuff;

$orderNumbers;

for($loopForStuff = 0; $loopForStuff < sizeOf($order); $loopForStuff++) {

$orderStuff[$loopForStuff] = $order[$loopForStuff][0];

}

for($loopForNums = 0; $loopForNums < sizeOf($order); $loopForNums++) {

$orderNumbers[$loopForNums] = $order[$loopForNums][1];

}

for($loopForPrices = 0; $loopForPrices < sizeOf($order); $loopForPrices++) {

$orderPrices[$loopForPrices] = $order[$loopForPrices][2];

}

$id=$\_POST['id'];

$name=$\_POST['name'];

$phone=$\_POST['phone'];

$address=$\_POST['address'];

mysql\_connect(localhost,$username,$password);

@mysql\_select\_db($database) or die( "Unable to select database");

$query = "INSERT INTO customers VALUES ('$id','$name','$phone','$address')";

mysql\_query($query);

$i=0;

while ($i < sizeof($order)) {

$query2 = "INSERT INTO theorders VALUES ('', '$id','$orderStuff[$i]','$orderNumbers[$i]','$orderPrices[$i]')";

mysql\_query($query2);

$i++;

}

$pending = "pending";

$query3 = "INSERT INTO orderStates VALUES ('$id','$pending')";

mysql\_query($query3);

mysql\_close();

?>

## Checking Order

### Display order details

Problem: To retrieve all of the details of a specific order with a valid reference number.

Solution: Simple MySQL to get the records and then map them to variables and output them.

*Listing*

<?php

$customerID=$\_POST['idd'];

include("dbinfo.inc.php");

mysql\_connect(localhost,$username,$password);

@mysql\_select\_db($database) or die( "Unable to select database");

$query="SELECT \* FROM customers WHERE id=$customerID";

$result=mysql\_query($query);

$num=mysql\_numrows($result);

echo "<b><center>Your details</center></b><br><br>";

$i=0;

while ($i < $num) {

$id=mysql\_result($result,$i,"id");

$name=mysql\_result($result,$i,"name");

$phone=mysql\_result($result,$i,"phone");

$address=mysql\_result($result,$i,"address");

echo "<b>$name</b><br>Phone: $phone<br>Address: $address<br>ReferenceID: $id<hr><br>";

$i++;

}

?>

<table width="300" border="1" cellspacing="2" cellpadding="0">

<tr>

<td><b>Quantity</b></td>

<td><b>Item</b></td>

<td><b>Price</b></td>

<?php

$query2="SELECT \* FROM theorders WHERE refid=$customerID";

$result2=mysql\_query($query2) or die('Query failed: ' . mysql\_error() . "<br />\n$sql");

;

$num2=mysql\_numrows($result2);

$grand\_total = 0;

$i=0;

while ($i < $num2) {

$item=mysql\_result($result2,$i,"item\_name");

$quantity=mysql\_result($result2,$i,"quantity");

$total\_price=mysql\_result($result2,$i,"total\_price");

$grand\_total += $total\_price;

?>

<tr>

<td><?php echo $quantity ?></td>

<td><?php echo $item ?></td>

<td><?php echo '£'.$total\_price ?></td>

</tr>

<?php

$i++;

}

?>

</table>

<br>

<?php

$query3="SELECT \* FROM orderStates WHERE id=$customerID";

$result3 = mysql\_query($query3);

$state = mysql\_result($result3, 0, "state") or die('Query failed: ' . mysql\_error() . "<br />\n$sql");

echo "Grand Total: £".$grand\_total;

?>

<br><br>

<?

echo "Order State: ".$state;

mysql\_close();

?>





# Testing

## Test Plan

### Sub-programs testing

|  |  |  |
| --- | --- | --- |
| **Sub Program description** | **Tested** | **Comment** |
| Read and display menu | ✔ |  |
| Quantity input validation | ✔ | Good, only allows numeric input |
| Get meal order | ✔ |  |
| Reference ID generation | ✔ | Had a slight glitch where the reference ID generated would not be input properly into the database but that was fixed by setting the generation to a 9 digit number |
| Get customer details | ✔ | Phone number validation but not address |
| Write meal order data to databases | ✔ |  |
| Display order details list | ✔ |  |
| Display customer details list | ✔ |  |
| Change state of order | ✔ |  |
| Check order | ✔ | Has to enter a valid reference number otherwise an error occurs |
| Company login | ✔ | Works but the username and password are hard coded. |
| Delete all data from tables | ✔ |  |

### Functional requirements testing

|  |  |  |
| --- | --- | --- |
| **Functional Requirement** | **Tested** | **Comment** |
| Accessible website | ✔ | Only accessible on school network |
| All options on website available as clickable buttons or links on screen | ✔ |  |
| User input for the quantities of orders, and customer details | ✔ |  |
| Validation of some inputs | ✔ | Quantities of meals, and phone number validated |
| User ability to track their order on website | ✔ |  |
| Relational database to store orders | ✔ |  |
| Company ability to print these orders | ✔ | The software itself does not allow the automatic printing of the order however it does arrange the order into a printable layout which then can be printed by the browser. |
| Company can update the state of an order on a database which will be able to be checked on the website | ✔ |  |

## Systematic and Comprehensive Testing

### Making an order

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Testing input of meal quantities** | | | | | |
| **Type** | **Test Data** | **Reason** | **Expected** | **Actual** | **Comment** |
| Normal | 5 Chips, 3 Fried Rice | Standard Order | 5 x Chips, 3 x Fried Rice, £10 | 1 x Chips, 1x Fried Rice, £10 | Good |
| Extreme | 99 Pizzas | Testing the limits | 99 x Pizzas, £306.9 | 99 x Pizzas, £306.9 | Good |
| Exceptional | 100 Chips and Curry Sauce | Testing if validation works | 10 x Chips and Curry Sauce, £61 | 10 x Chips and Curry Sauce, £61 | Good |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Testing of entering customer details** | | | | | |
| **Type** | **Test Data** | **Reason** | **Expected** | **Actual** | **Comment** |
| Normal | Name: Kenneth  Phone: 0123456789  Address: 1 Wiernotta Mews | Standard Order | Name: Kenneth  Phone: 0123456789  Address: 1 Wiernotta Mews | Name: Kenneth  Phone: 0123456789  Address: 1 Wiernotta Mews |  |
| Extreme | Name:  Phone: 00000000000  Address: !”£$%^&\*() | Testing the limits | Name:  Phone: 00000000000  Address: !”£$%^&\*() | Name:  Phone: 00000000000  Address: !”£$%^&\*() |  |
| Exceptional | Name: Kenneth  Phone: £123A12345” 2345  Address: 1 Street | Testing phone number validation | Name: Kenneth  Phone: 12312345234  Address: 1 Street | Name: Kenneth  Phone: 12312345234  Address: 1 Street |  |

## 

## Beta Testing

Three different classes of pupils were asked to beta test this system and report back any observations. One pupil in each class was given the company username and password and told to act as the company and the rest of the pupils were told to be customers and place orders. The ‘customers’ were asked to firstly make a normal order and comment on the good and bad areas of the system, and afterwards to act as hackers and try to find bugs in the system and report any errors that they found.

### The Customers

* “The system was good, and making the order was easy but more colour and pictures are needed to make it look better”
* “Everything was good apart from the displaying of the [reference] number and I didn’t take it down and forgot it because it was not clearly pointed out”
* “It is not very clear at the start how to actually order your meals but once it is found the rest of the system is easy to use”
* “It needs to look a lot better by adding colour and multimedia”
* “Good for what it does but needs more functions such as online payment”
* “It is not very aesthetically pleasing”

### The Company

* “Easy to navigate and update the order state of orders”
* “It is too easy to delete all of the orders, there should be a warning prompt after the button is pressed

### Errors Discovered

* If a wrong reference number is entered and tried to be checked, a MySQL error is returned.
* Sometimes an order can be received where the state cannot be changed and a MySQL error is returned

# Documentation

## User Guide

### Software features

The software includes:

* A website
* Ability to read and display a menu text file in a particular format
* Customer input and validation for quantities of meals and their details
* Creation of random reference IDs and validation of IDs
* Ability to write records containing these details to tables in a database
* Order checking
* Order state updating
* Administrator features protected by username and password
* Customer details lookup
* Order details lookup
* Delete lists function

### Using the system

To make an order:

* Open the index page of the website in your web browser by accessing this web page - <http://www-bhs.balernochs.edin.sch.uk/Pupils/2006/062761059/OrderTakerPHP/index.php>
* Click on the ‘Go to Menu’ link
* Click on the ‘Order Online’ button
* Enter the quantities of the meals you want in the text boxes and once you are finished click on the ‘Submit Order ->’ button
* Check your order and make sure that it is correct before clicking the ‘Continue’ link. If your order is incorrect or if you decide that you do not want to make an order then click on the links ‘Change your Order’ or ‘Back to Menu’ respectively.
* If you continued then on the next page firstly make a note of your reference number, then enter your name in the name input box, phone number in the phone number input box and your address in the address input box then click on the add to database button
* That’s your order completed! Now click on the ‘Back to Homepage’ button and wait for your meal to arrive (hypothetically).
* Should you wish to check your order or its state, then use our check order feature with your reference number.

To check your order:

* Open the index page of the website in your web browser.
* Click on the ‘Check Order’ link
* Enter your reference number into the reference number input box, then click on the ‘Check’ button
* View your order.

Company features:

* Open the index page of the website in your web browser.
* Enter the username ‘ken’ into the username input field
* Enter the password ‘wong’ into the password input field
* Click the ‘Company Login’ button
* To view a list of customers, click the ‘Show Customer List’ link
* To view a list of orders click the ‘Show Orders List’ link
* To view the menu click the ‘Show Menu List’ link
* To delete all data from database click the ‘Delete all Lists’ link
* To recreate menu from text file click the ‘Recreate Menu’ link
* To change order states for specific orders, first of all you must view the order list, find the order that you want to change, and click on its ‘Select Order’ button.
* In the blank text box underneath the reference ID, enter the state of the order that you would like and click on the ‘Update State’ button.

### FAQ and Troubleshooting

**Q**. How do I get onto the website?

**A**. If you are on a computer in the school network then the easiest way of accessing the website is to go to “www-bhs”, click on the Pupils Internal Home Pages link, find “Kenneth Wong” and click on the link, and click on the link that is shown there to gain access to the index page of the website. Alternatively here is the link: “<http://www-bhs.balernochs.edin.sch.uk/Pupils/2006/062761059/OrderTakerPHP/index.php>”

However if you are not on the school website then currently there is no way of accessing the website as it is hosted on the school’s internal server.

**Q**: Why can I not order more than 99 of each meal?

**A**: It has been predicted that no one will order more than 99 of any meal and if they do, then that kind of order would not be suitable for a small takeaway company therefore the input quantities fields will not allow more than 2 digits.

**Q**: Why is my reference ID so long and complicated? I cannot remember it!

**A**: The reference ID has been designed so that someone else cannot easily guess what IDs have been allocated so that they cannot access your important customer details. If you have problems remembering your ID then write it down somewhere.

**Q**: Why can I not access the company login on the homepage?

**A**: This is probably due to the fact that you do not represent the company and therefore have not been given the username and password required to access the company administrator features.

**Q**: Why can I not pay for my order online?

**A**: This feature has not been implemented yet due to the time constraints of the Advanced Higher course, it has been considered as a future addition to the system.

## Technical Guide

### Software requirements

* A web browser that supports HTML 4
* Any operating system to access the website
* Any server operating system with webserve software such as apache to host the website and database

### Hardware requirements

* A computer with enough RAM, processor speed and storage to handle normal web browser operations to access website i.e. 512MB RAM, Pentium 4 or newer processor that supports SSE2, 200MB free hard drive space (taken from recommended hardware requirements for Mozilla Firefox 11)
* A server computer with at least 4GB RAM, 3GHz clock speed, 20GB storage and with a permanent high speed connection to the World Wide Web

### List of files

* addtoDatabases.php
* admin.php
* checkOrder.php
* CreateTable.php
* dbinfo.inc.php
* deleteFromCustomers.php
* DisplayCustomers.php
* DisplayMenu.php
* DisplayOrders.php
* DropTable.php
* enterRefer.php
* getAddress.php
* index.php
* menu.php
* menu.txt
* numberonly.php
* OrderMenu.php
* paymentPage.php
* printOrder.php
* propertiesFile.txt
* readmenu.inc.php

# Evaluation

## Evaluation of system

### Fitness for purpose

The system performs all of the functions that are stated in the functional requirements and matches specifications exactly. It is a solution to the problem stated in the proposal and thus is fit for purpose. In order to be even more effective as a solution extra features such as online payment and making the user interface more attractive could be implemented however within the time frame of this project they were unachievable.

### User interface

Due to the nature of this project, which was to design and implement the underlying system for an online order taking service, not much time went into designing and creating a particularly good user interface. However the user interface was designed so that customers found it relatively simple to place an order and check their order through the use of a series of buttons and links that clearly stated their purpose. This was effective as shown in the beta testing reports; the ease of placement of an order was reported. On the other hand through the same beta tests, many more reports of the need to enhance the user interface was made. This could be done by including multimedia such as pictures of meals and adding colour to the website. The easiest way to implement a more attractive user interface would be to use cascading style sheets. This is something that I could do outside the scope of the project to enhance my system and also my web development skills.

### Robustness

Overall the system copes as normal with unexpected inputs apart from the checking of an order where if a wrong reference number is entered there is no validation and the wrong reference number is looked up in the table returning a MySQL error. This could be easily fixed by adding input validation to check whether the entered reference number is in the database before returning any output. Robustness is aided by the fact that PHP is run server-side so that it is not easy to access the code and exploit it. However due to the fact that my system is run on the school server the code is actually available to read for anyone on the network (though not write). This can be exploited, as if someone who wanted to maliciously invade my system, they would be able to see my PHP code, and know the security details to my database and be able to pass potentially harmful data into my database and affect the system. Ideally I would run this system on a protected server that no-one else had access to.

### Reliability

From the comprehensive testing we can see that the program is reliable as it gives the correct predicted results for all inputs tested. To make the system more reliable to the company, validation could be implemented for every single input such as the address, checking that it is a valid address within the area before allowing the order to be made.

### Portability

The program is very easily portable to other operating systems and platforms. This is due to the fact that both PHP and MySQL are supported on Unix and Windows based operating systems. Currently the system including database is run on a MAC OS X server. Given the right configurations and installations, the entire system could be ported to a Windows based system.

### Efficiency

For the most part the system has not been optimized for efficiency and is not as efficient as it could be. Some storage space is wasted on the database as the order state records could be combined with the customer records instead of being in a separate table. Additionally the menu need not be read more than once if saved to a data structure saving processing power, however this does not actually take much processing power and thus can be ignored. Overall the system does not require much processing power and RAM, and it depends on the number of orders made how much storage space is needed but each record requires a very small amount of storage so again not much is needed.

### Maintainability

The system is easily maintained as it has internal commentary describing the critical methods and code that will help with all maintenance. Perfective maintenance would be easily achieved, as it is all in web page parts and new web pages can be added for new features. Corrective maintenance is also easily done due to the level of separation given by the web pages; errors can be located to specific web pages and fixed. Adaptive maintenance does not require any changes to code, as it is just the server configurations that need to be changed.

## Evaluation of process skills

### Time management

Although the project was completed within the 40 hours my time management was not very good. Far too much time was spent on the implementation before I realized that I had not analyzed or designed the project very well, which made the implementation difficult. This meant that I had to go back to the design stage and spend more time on that before returning to implementation. However this has taught me the importance of analysis and design and I now know that they are pivotal to a project’s success.

### Alternative Strategies

Many alternatives could have been taken to achieve what I have created, both for the entire system and also for parts of it. For example the system could have been created using ASP.NET as the main development language instead of PHP. This would have been appropriate had I been operating on a Windows based system and had an ASP Apache server instead. Another alternative would be to have the menu as another table in the database instead of as a separate text file and read the data from the database, however for the requirements of this project I had to include file handling and this allowed me to do it simply. A different way of storing the order for use in other web pages would have been to use cookies, which would have worked equally as well. The reason why I used sessions instead was because I would not have to manage the expiration of the session, as it is when the user closes their browser but cookies need an expiry set which is difficult to gauge. The username and password for administrator privileges is hard coded into the program, these could be put into encrypted text files instead

# Appendix

## Full Program Listing