Department of Information and Technology

**(CampusName)**

Higher Diploma in Software Engineering (IT114105)

Module Name : Internet & Multimedia Applications Development

Module Code : ITP4523M

Submission Deadline : Phase 1: 12th teaching week of Semester 2

Phase 2: 7th teaching week of Semester 3

Hand in Methods : *To be announced by the lecturer*

This Group Project : **30% of total module marks** (*also it is part of EA components)*

**The result of EA will not be counted if you do not meet the minimum 70% attendance requirement (if any) governed by the *General Academic Regulations (GARs)* of your programme/course unless approval of the campus principal has been granted.**

1. Objectives

In this project, students are asked to:

* build a web application which provide different functions for *Sales Manager* and *Dealer.*
* apply software development skills to develop a website which is user-friendly, interactive, robust and easy to maintain.
* apply the knowledge that you learned in this module to solve the tasks. (i.e. HTML, CSS, JavaScript, PHP, SQL commands and Python)

1. A simple description to show how the web application will be used

There are *two user roles* for the web-based management system:

* 1. *A Dealer* can make the orders and retrieve the order records.
  2. *A Sales Manager* can manage items and update the order records.

1. Driving Question

You are the CEO of a *Smart & Luxury Motor Company*. Briefly explain how a centralized management system could help the company to promote sales and increase profit.

|  |
| --- |
|  |

1. Design for *Dealer* (Interface Design: 15 marks; Function: 30 marks)

|  |  |
| --- | --- |
|  | Done By |

* 1. Make the orders

Managed to make orders for *spare parts* (also known as *items*) supplied by SLMC.

View product information

Let the user easily select spare parts available from the distributor (only show items with

stock quantity greater than zero)

\* Sort the list of items *wisely* to facilitate smart purchasing

Required information for creating an orderas below:

1. Order ID
2. Order Date & Time
3. Delivery Address
4. Delivery Date
5. Spare Part Number
6. Spare Part Image
7. Spare Part Name
8. Order Quantity
9. Order Price
10. Total Order Amount

\* *Order ID* should be generated automatically by the system (thus it is a primary key)

\* Update the *stock item quantity* after an order is created

* 1. View the order records

Managed to view any order records created by the dealer. The system should display the following necessary information for each order record.

Required information for order record page:

* 1. Order ID
  2. Sales Manager ID
  3. Manager’s Contact Name
  4. Manager’s Contact Number
  5. Order Date & Time
  6. Delivery Address
  7. Delivery Date
  8. Order Status
  9. Spare Part Image
  10. Spare Part Name
  11. Order Quantity
  12. Order Price
  13. Total Order Amount

Function requirement:

\* Group ordered spare parts by part categories   
(e.g. spart part no. 200005, first digit '2' is the category "Major Assemblies")

\* List the items in ascending or descending order by at least TWO columns selectable by the user

* 1. Update dealer’s information

Managed to update the dealer’s information.

**Only allow the dealer to update the following information:**

1. Password
2. Contact Number
3. Fax Number
4. Delivery Address
   1. Delete Order record

Delete an order record from the *Orders* table and related records in *OrdersItem* table. Update the *stock item quantity* when an order is deleted.

Function requirement:

* A confirmation message should be displayed to let the user decide whether the selected order should be deleted or not.

\* An order can only be deleted at least two days before the delivery date.

1. Python Plug-in: Shipping Cost Calculator (Function: 10 marks)

Develop a simple *Python Flask* application running on *port 8080* and IP address *127.0.0.1* to determine the **shipping cost** based on the mode, the total weight (kg) or item quantity of the order. The **RESTful API** should accept the HTTP GET request and send the *required response* (in the *required* JSON format) from the Python program.

|  |  |  |  |
| --- | --- | --- | --- |
| **URL Request** | **/ship\_cost\_api/*<mode>*/*<value>*** | | |
| **Input Parameters** | ***<mode>*** is either "quantity" or "weight"  ***<value>*** is a numeric value which either represents the item quantity or total weight in kg | | |
| **Response in JSON** | {  "result" : "accepted",  "cost" : \_\_\_\_\_\_\_\_  } | *or* | {  "result" : "rejected",  "reason" : \_\_\_\_\_\_\_\_  } |

The shipping cost and limitation is determined by the following decision table:

|  |  |
| --- | --- |
| **Weight Mode** | |
| *Initial cost* per package- First 1 kg | $300 |
| Per 1 kg (starts from the second kg) | $50 |
| Maximum weight | 70 kg per package |

|  |  |
| --- | --- |
| **Quantity Mode** | |
| *Initial cost* per package - First unit of item | $300 |
| Per unit of item (starts from the second unit) | $60 |
| Maximum item quantity | 30 units of item per package |

Below are some example requests and the corresponding responses :

|  |
| --- |
| **HTTP GET request**  **Response in JSON format** |
| http://127.0.0.1:8080/ship\_cost\_api/special/200  { "result": "rejected", "reason": "Error : mode must be 'quantity' or 'weight'" } |
| http://127.0.0.1:8080/ship\_cost\_api/quantity/1  { "result": "accepted", "cost": 300 } |
| http://127.0.0.1:8080/ship\_cost\_api/quantity/3  { "result": "accepted", "cost": 420 } |
| http://127.0.0.1:8080/ship\_cost\_api/quantity/40  { "result": "rejected", "reason": "Maximum number of units per package is 30" } |
| http://127.0.0.1:8080/ship\_cost\_api/weight/0.5  { "result": "accepted", "cost": 300 } |
| http://127.0.0.1:8080/ship\_cost\_api/weight/4  { "result": "accepted", "cost": 450 } |
| http://127.0.0.1:8080/ship\_cost\_api/weight/80  { "result": "rejected", "reason": "Maximum weight per package is 70kg" } |

1. Design for *Sales Manager* (Interface Design: 15 marks; Function: 30 marks)

|  |  |
| --- | --- |
|  | Done By |

* 1. Insert items’ information

Managed to insert the spare part item with the following information.

Required Item information:

1. Spare Part Number
2. Spare Part Category
3. Spare Part Name
4. Spare Part Image
5. Spare Part Description
6. Weight
7. Stock Item Quantity
8. Price

\**Item ID* should be generated automatically by the system (thus it is a primary key).

* 1. Edit items’ information

Managed to update the spare part item created by the sales manager. The system should allow editing of the following information.

Required Item information:

1. Spare Part Description
2. Spare Part Image
3. Stock Item Quantity
4. Price
   1. Delete Item

Delete any spare part items created by the sales manager.

Function requirement:

* A confirmation message should be displayed to let the user decide whether the selected item should be deleted or not.

\* An item can be deleted only when the item does not relate to any order record.

* 1. Update order records  
     Managed to update any order not yet assigned to any sales manager. The Sales Manager should be able to:

(1) View and select any orders not yet assigned to other sales manager

(2) Update the selected order by setting the sales manager ID to the sales manager ID of the logged in sales manager

(3) Update the status of the selected order to "accepted" or "rejected"

Required information for managing an orderas below:

1. Order ID
2. Sales Manager ID
3. Manager’s Contact Name
4. Manager’s Contact Number
5. Order Date & Time
6. Delivery Address
7. Delivery Date
8. Spare Part Image
9. Spare Part Name
10. Order Quantity
11. Order Price
12. Total Order Amount
13. Order Status
14. Shipping Cost

\* Call the Python RESTful API to calculate the *shipping cost* and the *result* ("accepted" or "rejected" or "Error ...").

* 1. Generate report

Managed to view the statistics which can help a Sales Manager to wisely manage its stock spare part item.

Required information for report page:

1. Spare Part Number
2. Spare Part Name
3. Spare Part Image
4. Total number for each order item
5. Total sales amount ($) for each order item \*

\*total sales amount = price × order quantity

1. Form your project group

Each student needs to form a project group, **the maximum number of students in each group is 2.** Strongly recommend you to form a group to complete this project as you can benefit from sharing skills/codes amongst your members, and you can learn to plan, coordinate, and integrate work done by each member. Study carefully the given ERD and table structures before you start the implementation.

1. Additional requirements of your project
   1. Your web site should only use PHP as the server-side programming language (i.e. not ASP, ASP.NET, JSP, servlet etc.), however, you may use JavaScript and CSS for specific purposes. The database server used must be mySQL (version 5.0 or above).
   2. In your PHP code, you must ensure to use the following *parameter values* for the following mySQL database functions :  
       $conn = *mysqli\_connect*($hostname, $username, $password, $database);  
      set to the values below in a PHP script which is *shared by* the web pages :  
       $hostname = "**127.0.0.1**";  
       $database = "**projectDB**";  
       $username = "**root**";  
       $password = "";
2. Items to submit (Phase 1) (30% of total project marks)

Submit all UI design using CSS and HTML. Submission deadline will be announced by the

lecturer.

1. Items to submit (Phase 2) (70% of total project marks)
   1. A ***CD-ROM*** or ***DVD-ROM*** which stores a ***softcopy of all files*** for the whole web site. All files must be stored in non-compressed format (no .zip or .rar files please!)
   2. provide a SQL script file ***CreateProjectDB.sql*** to let the lecturer to re-create the database and test data
   3. for the SQL script file ***CreateProjectDB.sql***, it must contain *CREATE TABLE* commands to setup the database tables in **projectDB** database. Include necessary *INSERT* statements to add additional sample records you want to provide. The following is a sample SQL script :

drop database IF EXISTS **projectDB**;

create database **projectDB** character set utf8;

use **projectDB**;

You must specify the **InnoDB** engine for a database table :

ENGINE = **InnoDB**

Full explanation of different ***mySQL database engines*** :

<http://dev.mysql.com/doc/refman/5.0/en/storage-engines.html>

drop table IF EXISTS **Users**;

Create table **Users** (

userName Varchar(30) NOT NULL,

userPswd Varchar(10),

Primary Key (userName)) **ENGINE = InnoDB**;

INSERT INTO **Users** (userName, userPswd) VALUES

('admin1', 'secret1'),

('admin2', 'secret2');

* 1. a ***demonstration*** of your completed web site should be recorded by   
     a *30-day free-trial software Camtasia Studi*o 8  
     (<http://discover.techsmith.com/try-camtasia/clkn/https/www.techsmith.com/download/camtasia/>).   
     You should save different parts of your demonstration into different **.mp4** files. In a *Word* document named **video\_list.docx**, briefly describe the main content of each demo video file you have created. The video files will facilitate the lecturer to have in-depth evaluation of your web application. Here are some online tutorials for **Camtasia Studio 8** <http://www.techsmith.com/tutorial-camtasia-current.html> :  
     ***Getting Started: 1 - Record Full Screen*** :   
     <http://www.techsmith.com/tutorial-camtasia-record-full-screen.html>   
     ***Produce and Share an MP4 Video*** *:*   
     <http://www.techsmith.com/tutorial-camtasia-produce-and-share-mp4-video.html>

1. Assessment criteria of your project
   1. The functions implemented can perform correctly in *general* and *special* situations
   2. *Enough detail* of database records and extensive *data validation*
   3. Techniques used to promote *code reusability* (e.g. share common PHP/JavaScript/CSS files amongst different web pages) and *standardize the user-interface* of the web pages
   4. Coding style (e.g. indentation, meaningful variable names, modularity by user-defined functions etc.) and meaningful *comment* is added to program codes
   5. *Creativity* to enhance implemented functions so that they become easy to use, more interactive to the users or can handle some problems in real life situation
   6. Screen design and overall *quality of the integration* of different functions in the web site
2. A guideline for web development

It is a step-by-step approach I suggested for inexperienced web developers to develop the web site easily:

* decide what information to be displayed and design a number of web pages in HTML code (not PHP code at this stage) to display the information
* think about the site structure by creating different sub-folders to store files of different purposes (e.g. **images** folder to store image files, **style** folder to store CSS files, **Connections** folder to store files which define the settings for database connection) and design the linkages between the pages. You can easily view the site structure using DW8's site map view
* create HTML web pages (do not add JavaScript so soon) and design the layout with HTML codes and CSS rules. It is a good practice to check your .html files can pass the XHTML validation after completing a .html file
* when using CSS, it is preferred to create *external CSS files* (stylesheets) which can be reused in other web pages, so that other pages can have consistent formatting
* use DW CS6's template features which can help you to create a new page with a standard layout and also it provides common editable regions for web pages created from the same template.
* define frameset(s) and navigation bar or menu to link up different pages
* add JavaScript code to produce more interactive behaviors (such as validate data in the form, highlight a table row with different background color when the mouse moves over a table row). It is preferred to use *external JavaScript file* which will be reused in other web pages
* replace hyperlink text with image / button to beautify the links. Dreamweaver can help you to create nice Flash buttons easily
* finally, it comes to the hardest work, that is to convert some of the HTML codes into PHP codes in order to generate dynamic contents from data extracted from database, cookie and PHP pre-defined arrays ($\_POST, $\_GET, $\_COOKIE, $\_SESSION, $\_FILES, $\_SERVER etc.)

1. Penalty for plagiarism

* Each student needs to submit his/her own work. Plagiarism (抄襲) will be treated seriously.
* All group projects that have been found involved wholly or partly in plagiarism (no matter these projects are from the original authors or from the plagiarists) will score ZERO marks. Furthermore, disciplinary action will be followed.

**Late submission will receive ZERO marks**