MODULE NAME:	MODULE CODE:
WEB DEVELOPMENT (INTERMEDIATE)	WEDE6011
WEB DEVELOPMENT (INTERMEDIATE)	WEDE6021

ASSESSMENT TYPE: POE (PAPER)

TOTAL MARK ALLOCATION: 100 MARKS

TOTAL HOURS: A minimum of 15 HOURS is suggested to complete this assessment.

By submitting this assignment, you acknowledge that you have read and understood all the rules as per the terms in the registration contract, in particular the assignment and assessment rules in The IIE Assessment Strategy and Policy (IIE009), the intellectual integrity and plagiarism rules in the Intellectual Integrity Policy (IIE023), as well as any rules and regulations published in the student portal.

INSTRUCTIONS:

1. ***No material may be copied from original sources, even if referenced correctly, unless it is a direct quote indicated with quotation marks. No more than 10% of the assignment may consist of direct quotes.***
2. ***Any assignment with a similarity index of more than 25% will be scrutinised for plagiarism. Please make sure you attach a similarity report to your POE if required.***
3. ***Make a copy of your assignment before handing it in.***
4. *Assignments must be typed unless otherwise specified.*
5. *All work must be adequately and correctly referenced.*
6. *Begin each section on a new page.*
7. *Follow all instructions on the assignment cover sheet.*
8. *This is an individual assignment.*

Referencing Rubric

Providing evidence based on valid and referenced academic sources is a fundamental educational principle and the cornerstone of high-quality academic work. Hence, The IIE considers it essential to develop the referencing skills of our students in our commitment to achieve high academic standards. Part of achieving these high standards is referencing in a way that is consistent, technically correct and congruent. This is not plagiarism, which is handled differently.

Poor quality formatting in your referencing will result in a penalty **of a maximum of ten percent being deducted from the percentage awarded**, according to the following guidelines. Please note, however, that **evidence of plagiarism in the form of copied or uncited work (not referenced), absent reference lists, or exceptionally poor referencing, may result in action being taken in accordance with The IIE's Intellectual Integrity Policy (0023)**.

Markers are required to provide feedback to students by indicating **(circling/underlining) the information that best describes the student's work.**

Minor technical referencing errors: 5% deduction from the overall percentage – the student's work contains **five or more errors** listed in the minor errors column in the table below.

Major technical referencing errors: 10% deduction from the overall percentage – the student's work contains **five or more errors** listed in the major errors column in the table below**.**

If both minor and major errors are indicated, then 10% only (and not 5% or 15%) is deducted from the overall percentage. The examples provided below are not exhaustive but are provided to illustrate the error

Required: **Technically correct referencing style**	**Minor errors in technical correctness of referencing style** **Deduct 5% from percentage awarded**	**Major errors in technical correctness of referencing style** **Deduct 10% from percentage awarded**
Consistency • The same referencing format has been used for all in-text references and in the bibliography/reference list.	Minor inconsistencies. • The referencing style is generally consistent, but there are one or two changes in the format of in-text referencing and/or in the bibliography. • For example, page numbers for direct quotes (in-text) have been provided for one source, but not in another instance. Two book chapters (bibliography) have been referenced in the bibliography in two different formats.	Major inconsistencies. • Poor and inconsistent referencing style used in-text and/or in the bibliography/ reference list. • Multiple formats for the same type of referencing have been used. • For example, the format for direct quotes (in-text) and/or book chapters (bibliography/ reference list) is different across multiple instances.
Technical correctness Referencing format is technically correct throughout the submission. Position of the reference: a reference is directly associated with every concept or idea. For example, quotation marks, page numbers, years, etc. are applied correctly, sources in the bibliography/reference list are correctly presented.	**Generally, technically correct with some minor errors.** • The correct referencing format has been consistently used, but there are one or two errors. • Concepts and ideas are typically referenced, but a reference is missing from one small section of the work. • Position of the references: references are only given at the beginning or end of every paragraph. • For example, the student has incorrectly presented direct quotes (in-text) and/or book chapters (bibliography/reference list).	**Technically incorrect.** • The referencing format is incorrect. • Concepts and ideas are typically referenced, but a reference is missing from small sections of the work. • Position of the references: references are only given at the beginning or end of large sections of work. • For example, incorrect author information is provided, no year of publication is provided, quotation marks and/or page numbers for direct quotes missing, page numbers are provided for paraphrased material, the incorrect punctuation is used (in-text); the bibliography/reference list is not in alphabetical order, the incorrect format for a book chapter/journal article is used, information is missing e.g. no place of publication had been provided (bibliography); repeated sources on the reference list.
Congruence between in-text referencing and bibliography/ reference list • All sources are accurately reflected and are all accurately included in the bibliography/ reference list.	**Generally, congruence between the in-text referencing and the bibliography/ reference list with one or two errors.** • There is largely a match between the sources presented in-text and the bibliography. • For example, a source appears in the text, but not in the bibliography/ reference list or vice versa.	**A lack of congruence between the in-text referencing and the bibliography.** • No relationship/several incongruencies between the in-text referencing and the bibliography/reference list. • For example, sources are included in-text, but not in the bibliography and vice versa, a link, rather than the actual reference is provided in the bibliography.
In summary: the recording of references is accurate and complete.	In summary, at least **80%** of the sources are correctly reflected and included in a reference list.	In summary, at least **60%** of the sources are incorrectly reflected and/or not included in reference list.

Overall Feedback about the consistency, technical correctness and congruence between in-text referencing and bibliography:

.....

.....

Contents of PoE:

This portfolio of evidence is an integration of all four tasks submitted during the semester for WEDE6011 as indicated after a number of sessions are completed. It must include the improvements suggested by your lecturer in the feedback provided for each task. As part of your submission, you must include all documentation of the website and database tables. You may be expected to research topics that were not covered in your lectures for your tasks. Any coding obtained from an external source must be referenced within your script at that point of usage. Every script must contain your student number, name and surname and a declaration or statement that the coding is your own work where not referenced.

Categorise your CD/DVD submission documentation in folders, E.G:

- Root folder with html and php root files and documentation such as the ERD and self-evaluation within a Word document:
  - CSS sub folder;
  - js sub folder;
  - images sub folder.

Details for PoE:

The POE is broken down into three tasks that must function as one web application; for you to demonstrate your PHP scripting abilities and to provide a Portfolio of Evidence demonstrating your learning from the previous tasks. You are required to use Chapters 1 – 10 in the textbook. In other words, ensure that you demonstrate your understanding of:

- PHP scripting
- Functions and control structures
- Manipulation of strings
- Handling of user input
- Manipulating arrays
- Working with databases and MySQL
- Manipulating MySQL Databases with PHP
- Managing State Information
- Object Oriented PHP
- Implementation of Object-Oriented PHP on the eShop

Please note that when completing your three tasks, you are required to use good coding standards. Please refer to the marking rubric which indicates how your coding will be assessed. Marks are specifically allocated in Task 1, but must be followed through within the remaining tasks. If you deviate from coding standards, it will influence your allocated mark for the subsequent tasks.

Before submitting your task ensure that the specific task meets the requirements set out in the marking rubric provided further down.

You are required to document your development by reflecting on:

1. Explanations of shortfalls (lessons learnt) for:
    a. Variable naming;
    b. Comments/code readability throughout PHP code;
    c. PHP file naming (modular coding).

2. User friendly design in:
    a. Error handling and data validation using HTML5;
    b. Integration of member functions and objects;
    c. Shopping cart data structure user experience;
    d. Correct structuring of database scheme with tables.

3. Pros and cons you experienced in working with PHP.

Instructions for Task 1

Assessment/ deliverable	Marks	Weight	Duration
Task 1	100	30%	15 hrs

Task 1: Start during Session 10 and complete after Session 27:

1.1 Complete the sub tasks set out in the Module Outline for LU3 – 4 as needed. The goal is to understand and implement a sticky form, All-in-One form using the $_SERVER ["SCRIPTNAME"] autoglobal. Use that in conjunction with an associative array, read from the database and ensure PHPMyAdmin is used to create tables and export the database as an SQL file. Please consult WEDE5010 and Ullman on creating a template using HTML5 folder structure (images, css, js) to abide by the rules and use HTML5 for validation purposes as set out in https://www.wufoo.com/html5/ when creating your form.

1.2 Style your login page, using a CSS framework. Reference Ullman Quickstart in Chapters 3, 4, and 8 for responsive web design (RWD). You may also use Bootstrap CSS instead of Ullman's Concise CSS (http://concisecss.com), but Chapter 8 (Ullman) covers example code in detail.

1.3 Do sub tasks in LU3 Theme 1 (*refer to Module Outline*) in creating a login form achieving the last sub task when a form calls itself.

1.4 Do sub task 4 from LU4 Theme 2 (*refer to Module Outline*) in reading a text file with proverbs into an array, creating a table within a database and writing those proverbs to the table using PHP and MySQL. There are solution scripts which the lecturer can explain in class if you struggle. Ensure the connection script is embedded in an include file, e.g. dbConn.php.

Task 1 — Login Validation and Table creation (Marks: 100)

Your task must meet the following specifications:

1. Create a table ***tbl_User*** in MySQL using the console or phpMyAdmin, consisting of the following column names:

    a. The table structure is as follows for tbl_User:
    ID (autoincrement - PK), FName (text), LName (text), Email (text), Password (Text).

    b. Create a text file ***userData.tx***t and populate the textfile with at least 30 fictitious entries, e.g. John Doe j.doe@abc.co.za 29ef52e7563626a96cea7f4b4085c124.

    c. Use the console or phpMyAdmin and load the text file manually into the table.

2. Use the existing ***test*** database that is pre-installed with wamp or xamp.

    a. Create a connection within the ***test*** database using PHP and store the code which creates a connection in a file called ***DBConn.php***.

    b. Create a script called ***createTable.php*** that will check if the table ***tbl_User*** exists and if it does, delete the table and (re)create the table and load the data into the table using the ***userData.txt*** file as source file.

    c. Embed the ***DBConn.php*** as an include file within the ***createTable.php*** script. Hint: you may include the connection code in the createTable.php directly and later refactor the code into an include file to modularise your code.

    d. Each time the script is run the table will be deleted if it exists and reloaded afresh with the data stored in the text file.

3. Create a login page for your project (LU2-LU4). The login page must:

    a. Accept a user's name, surname, email address and password.

    b. The password must be compared to a e.g. hash "29ef52e7563626a96cea7f4b4085c124" in the ***tbl_User table***. The correct password is: "P@55w0rd!" Note: you may **not** hard code the hashed value in your PHP code.

    c. When clicking the submit button, use HTML5 to validate. Textboxes and the password from the login details must be compared to the stored hashed password value in the MySQL database.

    d. If the validation satisfies the password, then display the user's data using an associative read approach regarding the column names in a table.

However, if the password is incorrect then use a sticky form and redisplay the details entered allowing the user to edit the fields instead of re-typing all the fields. Display a string at the top of the page that identifies the user and reads: "User John Doe is logged in"

e. If the user does not exist, he/she can register him/herself to create the hash and login.

4. Create a text file ***item.txt*** on items you want to sell and load them into a table ***tbl_Item***. As an example, a computer hardware store will sell items such as PCs, printers, hard drives, flash drives etc. and may have the following table structure: **tbl_Item**(ItemID (text) (PK), Description (Text), Cost Price (numeric(15,2)), Quantity (numeric), Sell Price(numeric(15,2))
   a. Decide on what type of items you want to sell and load 15 items of your business of choice into the text file.
   b. Create a table ***tbl_Item*** and load the data from the textfile into the table.
   c. Ensure that you have a picture for each item, to display the picture on that item within a table.
   d. These pictures must be stored in a folder ***images***. Take care of naming these jpg files for the ItemID must pick up the picture when these items are displayed in a table. You may use another way of storing the name of the image as part of the table.

5. When the user starts the index.php script, the user is prompted to log into the system. He/she must be able to click on a button that will display the table with items, once logged in. The button called "Show Items" will appear next to the user's data. When clicking on "Show Items" button, the table of items must also have a button next to each item line, called ***AddToCart***. The button may have a picture of a cart, instead of the words. When this ***AddToCart*** button is clicked, the item with the item's ***Sell Price*** will be displayed in a popup window. **Please note that you must use associative read from the database and NOT hard code the items in a table in HTML. A table must be built dynamically when reading the items table to display items.** No marks will be given for hard coded items.

Instructions for Task 2

Assessment/ deliverable	Marks	Weight	Duration
Task 2	100	30%	15 hrs.

Task 2: Start during Session 28 and complete after Session 33. Entity Relationship Diagram (ERD)

1. Define or explain the following terminologies in class:

Refer to your database textbook or internet and research terms such as "Composite, Simple, Single-value, Multivalued and Derived attributes", "Connectivity", "Cardinality, "Degree", "Existence dependence and independence", "Weak Entity", "Optional and Mandatory participation", "Unary relationship", "Binary relationship", "Ternary relationship", "Bridge or linking entities", "Recursive or Iterative relationships".

2. Using the CHEN model draw an ERD for the following scenarios. Indicate all Primary keys (PK) and Foreign keys (FK).

2.1. A CAR entity with attributes CAR(Car_Vin, Mod_Code, Car_Year, Car_Colour) where Car_Vin is the PK and Car_Colour may be multiple colours for that car.

2.2. One Professor teaches many classes. A class may be taught by a minimum of one professor and maximum of four professors, but a Professor may only teach minimum and maximum one class.

2.3. An Employee may have many Dependents. Dependent is optional to Employee, but any Dependent will have one Employee associate with it for medical aid purposes.

2.4. A Student may enrol for many Subjects or Modules, but a Module or Subject may have many students. A Class is optional to Student and a Student is optional to Class. Simply this many too many relationships and indicate the cardinalities using CHEN model.

3. Recursive Relationships:

3.1. A Person may play the role of husband or wife, but a husband may only have one wife and vice versa.

3.2. An Employee may act as supervisor over many Employees.

3.3. A Part consists of sub-parts. A Part may play the role of Main part or sub part. E.g. A Car consists of parts such as a gearbox and a gearbox can be broken down into sub-gears.

Task 2 —Design and implementation of the Entity Relationship Diagram (Marks: 100)

Your Task 2 must meet the following specifications:

1. Design a database management system consisting of the following base tables. Simplify the design by analysing the relationships among them: tbl_Customer, tbl_Order, tbl_Item. A customer may place many orders and an order may have many items and one item may appear on many orders. Ensure that you clearly show all primary keys (PK) and foreign keys (FK) and constraints of all attributes or column names. Represent your design using a CHEN model Entity Relationship Diagram (ERD). All many to many relationships must be resolved into one to many relationships. Use Word or Visio and represent your design. Capture as a pdf file and keep towards the POE.

2. Create tables for each entity in your design using MySQL in console or phpMyAdmin mode.
    a. Export your structure of each table to a Word file as part of your POE documentation.
    b. Create a text file for data on each base table and populate the textfile with at least 30 fictitious entries for each base table.
    c. Use the console or phpMyAdmin and load the text file (data) manually into each base table.
    d. Export the database structure to a textfile called ***myShop.sql*** with the DDL statements so the lecturer can use the sql-text file to create your database with 30 entries for each base table.
    e. Ensure that you create the necessary primary keys and foreign keys coding the constraints as dictated by the ERD design.
    f. Create a script loadMyShop.php that will create the tables within the “test” database. Ensure that all tables are dropped before creating them and that a table is created only if it does not exist. Use mysqli or improved mysql to create your connection in an include file. Hint: Export your database to an SQL file and use the exported code in association with PHP code.

Instructions for Task 3

Assessment/ deliverable	Marks	Weight	Duration
Task 3 (PoE)	70	35%	15 hrs.
Task 1 (Revised)	10		
Task 2 (Revised)	10		
Self-Evaluation	10		

Task 3: Start during Session 34 and complete after Session 54

1. Complete the sub tasks set out in the Module Outline for LU5 and LU6 as needed to assist in understanding the object-oriented approach. The goal is to understand how to maintain state, the implication of query strings, sessions and cookies used within sessions. The code within the Gosselin textbook is only a reference and you may deviate from the layout and member functions as your understanding improves. Understand the process which is followed in the textbox in displaying the items to the user and then allow the user to shop.
2. Verify the relationship between a class instance/ object and a member function embedded within. Introduce yourself to objects by doing **sub task 2** at the end of LU5 (*refer to Module Outline)* combining a class (onlineStore.php) with the script myGreeting_Object.php. This will illustrate how to call any function from within an object. There are solution files your lecturer may explain to those who struggle.
3. Do **sub task 3** in LU5 (*refer to Module Outline)* and combine sessions with an instance of a class.
4. Do **sub task 4** in LU5 (*refer to Module Outline)* to move code into a member function inside the class.
5. Create an instance of the class and display the inventory of the coffee data.
6. Do **sub task** 1 within LU6 (*refer to Module Outline)* in building member functions. Move the code to display the inventory file into a member function getProductList(). The shoppingCart array is exactly the same as the inventory array except that it contains only what the client wants to order. So, a suggestion would be to duplicate the code of the getProductList function and call it showCart and use an array shoppingCart instead of inventory array here.
7. Study the code of the GGC.php script and create a webflow of how your website will operate, i.e. displaying the inventory, allowing the user to add items to the shopping cart and provide an option to display the shoppingcart selection.

   Name and study all the special member functions within a PHP class associated with

serialisation and unserialisation of an object. Hint: refer to the textbook Chapter 10 on Serialised Functions on pp.594.

8. ***Checking out*** of your store is most important and this is where the items inside the shopping cart must be recorded into the appropriate database tables.
9. You will not engage with any third party utility e.g. PayFast or PayPal, but merely show the order number as a way that the user may track his/ her purchase.
10. If you accomplished the understanding of objects you can easily show all purchases a user made.
11. Create a webflow of how a purchase takes place and which tables are involved when the shoppingCart items are recorded against a user's name. Hint: the user table places an order which contains many Items or products. The quantity and quoted price must be recorded at the time and the session id may be concatenated with the order id as a tracking string and stored in the database. Criticise Gosselin's approach in the textbook (Exercise 10-2: function checkout()).

    What other tables are involved here, which the author ignored?

```
public function checkout()
{
    $ProductsOrdered = 0;
    foreach($this->shoppingCart as $productID =>$quantity)
    {
     if ($quantity > 0)
     {
        ++$ProductsOrdered;
       $SQLstring = "INSERT INTO orders " ." (orderID, productID, quantity) " ." VALUES('" .
session_id() . "', " .
                "'$productID', $quantity)";
       $QueryResult = $this->DBConnect-> query($SQLstring);
     }
    }
    echo "<p><strong>Your order has been " ."recorded.</strong></p>\n";
 }
```

A report may be extracted to show all the orders placed by a user if the correct tables are used.

Task 3 (Marks: 70)

The Shopping Cart using Member Functions , and Checking out of the eShop – Buy button or the Bye-Bye button

1. Create an include file "aShopCart.php" and script "myShop.php". Inside the aShopCart.php, create a class called ShoppingCart:
   a. The file called "aShopCart.php" contains all member functions that will address serialisation and unserialisation of values.
   b. Add other member functions that will display the contents of the ***tbl_Item*** table. The items must be displayed in an HTML table with the necessary headings showing the quantities available and retail prices. Each line must have a button "AddToCart" or picture of a cart and the picture of the item.
   c. Create a script myShop.php which will include the "aShopCart.php", load the Cascading Stylesheet (CSS) giving a specific look and feel to your pages and display all items of the ***tbl_Item*** table, available for purchasing.
   d. Member function such as emptyCart(), removeItem(), __destruct(), __wakeup(), __sleep(), addItem() and processUserInput() can be studied in the latter part of Chapter 10 and in the exercises at the rear of Chapter 10.

2. Integration and functionality for a normal user and an administrative user:
   a. The user must be confronted with a startup page of the eShop. The startup page must show the name of the eShop and describe the purpose of the eShop to entice the user in buying from the eShop, i.e. functionality and type of business. The startup page must also carry an "Admin" button and a "Shop" button. Note that "Admin" is a backend option and will not normally appear for your users to see, but this is just an academic exercise. You may load this as a separate URL in demonstrating the Admin operations only.
   b. When clicking on the "Shop" button, the items are displayed when running the functions embedded within the "myShop.php" script with the necessary buttons "AddToCart" at the end of each row and "ShowCart" at bottom and top of the table. The items need NOT be displayed in a table only, but you can access or consult other eShops on the Internet as examples on how products within the eShop are displayed.
   c. The user must be able to add items to his/ her shopping cart array, but then also show the contents of the shopping cart array when clicking on the "ShowCart" button and jump to the Shopping Cart array and back to the Items table to continue shopping.

To display all items of the shopping cart, use a table layout without pictures or minimised pictures to make the layout simple.

d. The "ShowCart" option displays the items within the shopping cart array but must also have an option to add more of that specific item, showing the quantities of these items and the total cost of all the items in the cart. Each line of the cart must display the unit price, the quantity ordered and the total per line which is quantity X unit price. **Please note we do not write the shopping cart to database, but maintain it as a data structure in memory. Here those who are adventurous may use a JSON object to maintain the shopping cart.** The following figures illustrate the standard basic example as in the textbook:

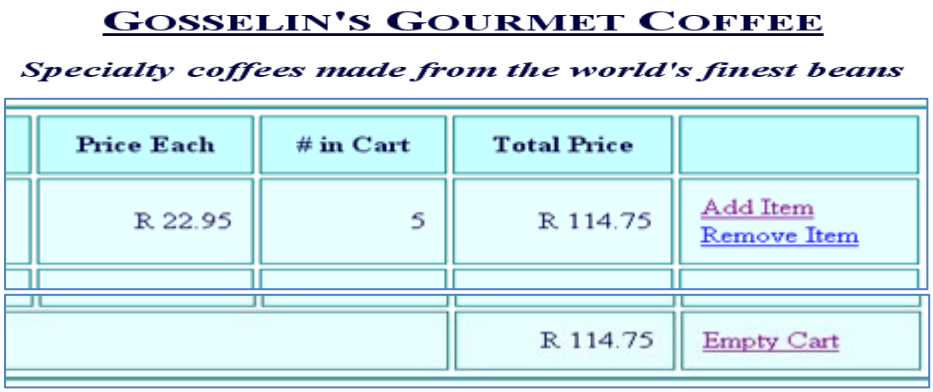

Figure 1: Example Shopping Cart Array with Total per line and Total for Cart

Figure 2: Example Shopping Cart Array with Checkout button

f. When the user clicks the “Admin” button, the user **must** be prompted to login with administrator rights, unless the user with those rights is already logged in. You can also use a URL to load the Items table. The table with items is now displayed, but with the buttons “Edit”, “Delete” on every row. The page must also have an “Add” button where new items can be added to the items table, at the bottom of the page. Pictures/ Files can be uploaded consulting pp.247 – 251 in Chapter 5 for a working example. Ensure every picture is sized correctly in a small and large size image. You can use any utility such as lightbox or carousel to show/ view your items. Validation on quantities in stock is quite important to track quantities and as items are sold the stock quantity must decrement in the items table. Inspect this link on having both on same page: http://visuallightbox.com/rq/jquery-gallery-carousel-and-video-lightbox-on-the-same-page-9u.html.

3. The user must be able to either “Continue” shopping or go to the “CheckOut” hyperlink. When adding a similar item to the shopping cart only the quantity must increment and the item should not be added as a separate entry. When checking out by clicking on the “CheckOut” button, the user **must** be prompted to **login**, unless the user is logged in already or register as a new user. The textboxes must be validated using HTML5. The necessary actions must be taken to place the order (and the items associated with that order) and display the order with its items with a reference number. The actual payment will not be part of the assignment, but assumed when the “CheckOut” button is clicked. You can examine code for the checkout.php script and the function checkout() in Chapter 10 as discussed in the Exercises.
4. Please note that when an Order is created that Order_ID is an autonumber and must be used as the foreign key (part of the primary key) in the OrderLine table. Please reference Chapter 8 (p.466) how to return the ID created with AUTO_INCREMENT after the insert operation and that ID is not visible. This is accomplished by the **mysql_insert_id()** function in mysql.

The mysqli option is `$mysqli->query($query)`

EXAMPLE – Reference: http://php.net/manual/en/mysqli.insert-id.php

```
<?php
    $mysqli = new mysqli("localhost", "my_user", "my_password", "worl
    d");
    /* check connection */
    if (mysqli_connect_errno()) {
        printf("Connect failed: %s\n", mysqli_connect_error());
        exit();
    }
    $mysqli->query("CREATE TABLE myCity LIKE City");
    $query = "INSERT INTO myCity VALUES (NULL, 'Stuttgart', 'DEU', 'S
    tuttgart', 617000)";
    $mysqli->query($query);
    printf ("New Record has id %d.\n", $mysqli->insert_id);
    /* drop table */
    $mysqli->query("DROP TABLE myCity");
    /* close connection */
    $mysqli->close();
    ?>
```

5. Combine Task 1 to Task 3 and deliver an eShop where users can view your items and put items into a shopping cart. Only when ***checking out*** the items will be recorded into the database. You should provide as much freedom as possible without forcing the user to unnecessarily log into your eShop. In Task 1 we first logged in to make the webflow easier, but in the final stages a login will **only** be required if the user wants to purchase items.

Appendix A

Assessment Sheet (Marking Rubric)

Please note: Tear off this section and **attach** it to your work when you submit it.

MODULE NAME:	**MODULE CODE:**
WEB DEVELOPMENT (INTERMEDIATE)	**WEDE6011**

STUDENT NAME:
STUDENT NUMBER:

RUBRIC 1: Login Validation and Table creation	**Levels of Achievement**				**Feedback**
In order to be awarded full marks for these elements of Task 1, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks)				
Good coding standards					
1. Variable naming and User friendly design	**5**	**3—4**	**1—2**	**0**	
2. Comments/ code readability throughout PHP code;	**5**	**3—4**	**1—2**	**0**	
3. PHP file naming	**5**	**3—4**	**1—2**	**0**	
4. modular coding – use of functions where necessary	**5**	**3—4**	**1—2**	**0**	
User Table Structure and Connection					
5. Text file correctly created with 30 names	**5**	**3—4**	**1—2**	**0**	

RUBRIC 1: Login Validation and Table creation	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 1, students need to have:	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ marks)				
6. Data is preloaded into a table tbl_User – check in console.	5	3—4	1—2	0	
7. The script DBConn.php is included within the DBConn.php script with correct include statement.	5	3—4	1—2	0	
8. Code makes a connection.	5	3—4	1—2	0	
9. The script createTable.php deletes table, creates table and load the data (Check in console).	5	3—4	1—2	0	
Login Page					
10. Checks password against hashed password in tbl_User.	5	3—4	1—2	0	
11. Registration of new user.	5	3—4	1—2	0	
12. Textboxes are validated and if password is incorrect the values stay – sticky form works.	5	3—4	1—2	0	
13. Associative columns are fetched from the tbl_User table – **Check code embedded in function.**	5	3—4	1—2	0	

Item Table Structure					
14.	Text Files item.txt contains at least 15 items and tbl_Items structured correctly.	**5**	**3—4**	**1—2**	**0**	
15.	Images folder exists and contains 15 jpg files with significant filenames.	**5**	**3—4**	**1—2**	**0**	
16.	The table tbl_Item is (assoc) read into array and displayed in a table with headers.	**5**	**3—4**	**1—2**	**0**	
17.	Each line contains the picture of the item.	**5**	**3—4**	**1—2**	**0**	
18.	Each line carries the AddToCart/ Cart Picture button.	**5**	**3—4**	**1—2**	**0**	
19.	When AddToCart is clicked the SellPrice is shown in popup and return to table.	**5**	**3—4**	**1—2**	**0**	
20.	CSS gives a professional look and feel to Task as a whole, i.e. Login Page and Tabular display.	**5**	**3—4**	**1—2**	**0**	
TASK 1 SUBTOTAL						**/100**

RUBRIC 2: Design and mapping of ERD	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 2, students need to have:	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ marks)				
ER Diagram (CHEN Model) and Mapping					
1. ER diagram correctly structured with all properties.	**14—15**	**10—13**	**7—9**	**0—6**	
2. Tables are correctly named and structured.	**14—15**	**10—13**	**7—9**	**0—6**	
Populating Items table and PHP-code to create tables with data					
3. Items table populated with 30 data entries.	**14—15**	**10—13**	**7—9**	**0—6**	
4. Key constraints are in order for all tables, i.e. Primary and foreign keys (detail/ child tables).	**14—15**	**10—13**	**7—9**	**0—6**	
5. Database exported to myShop.sql. Clear DB and load.	**18—20**	**12-17**	**8—11**	**0—7**	
6. PHP script that will drop tables, create tables and load data – **loadMyShop.php.**	**18—20**	**12-17**	**8—11**	**0—7**	
TASK 2 SUBTOTAL					**/100**

RUBRIC 3: Shopping Cart using member functions	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 3, students need to have:	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ Mark)				
Shopping Cart Class and Member Functions					
1. Member functions AddItem, RemoveItem, Checkout, EmptyCart, Login, ProcessInput are present. As long as the student illustrates understanding, he/ she may use own functions. These names are used within the book.	**7—8**	**5—6**	**3—4**	**0—2**	
2. Startup page clearly states type of eShop and goals styled on some CSS.	**4**	**3**	**1—2**	**0**	
3. eShop button displays Items table with buttons AddToCart and Show Cart.	**4**	**3**	**1—2**	**0**	
4. AddToCart does perform function - test by clicking on ShowCart to view the shopping cart contents.	**7—8**	**5—6**	**3—4**	**0—2**	
Administrator Option to load Items and Pictures of the Items					
5. Prompt for login when Admin button is clicked or url loaded.	**7—8**	**5—6**	**3—4**	**0—2**	
6. Admin button/ url landing page displays items table with buttons Edit, Delete and Add/ Insert.	**7—8**	**5—6**	**3—4**	**0—2**	
7. Add, Delete and Edit button works for Admin on Items table.	**7—8**	**5—6**	**3—4**	**0—2**	

RUBRIC 3: Shopping Cart using member functions	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 3, students need to have:	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ Mark)				
Shopping Cart Functionality					
8. When adding same item to Cart, quantity increases and not new item added.	**4**	**3**	**1—2**	**0**	
9. When continue shopping Cart remains available with selected Items still intact.	**4**	**3**	**1—2**	**0**	
Shopping Cart Class and Member Functions					
10. Checkout calls login page if not logged in already or option to register.	**4**	**3**	**1—2**	**0**	
11. Checkout shows reference number e.g. ordernumb and sessionid.	**4**	**3**	**1—2**	**0**	
12. Checkout writes entries into orderline table and quantity decremented (check tables in database).	**7—8**	**5—6**	**3—4**	**0—2**	
13. After checkout the Shopping Cart array is empty.	**4**	**3**	**1—2**	**0**	
14. User has option to draw history of purchases. Report must show total of all purchases at bottom of page. **7—8**					

RUBRIC 3: Shopping Cart using member functions	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 3, students need to have:	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ Mark)				
TASK 3 SUBTOTAL					**/70**

[TOTAL MARKS: 100]