|  |
| --- |
| CS2102 Project |
| Module Bidding System |
| Specially Designed For Students To Register Modules… |

|  |
| --- |
| Group 9  Cheong Ke You, A0112707N  Lu Yanning, A0112971J  Tan Teck Li, A0111770R  Chan Jun Wei, A0112084U  Chua Chin Siang, A0112089J |

## **Web server**

Web server used is Zone. (We used local server to test also).

## **Server page language**

Server Page Language: php.

Frontend: Html, CSS

## **Database management system used**

Query language used is SQL.

Database Management System is Oracle.

## **ER diagram**

## **Relational Schema**

### **Modules:**

The table “modules” is created to save the details of a module.

CREATE TABLE modules (  
moduleCode VARCHAR (16),  
moduleName VARCHAR (128),  
PRIMARY KEY (moduleCode)  
);

### **Prerequisite:**

The table “prerequisite” is created to save which module is which module’s prerequisite. (It’s a relation of an instance of “modules” and another instance of “modules”.)

CREATE TABLE prerequisite (  
andId INT,  
module VARCHAR (16),  
requiredModule VARCHAR (16),  
FOREIGN KEY (module) REFERENCES modules(moduleCode) ON DELETE CASCADE,  
FOREIGN KEY (requiredModule) REFERENCES modules(moduleCode) ON DELETE CASCADE,  
PRIMARY KEY (andId, module, requiredModule));

### **Preclusion:**

The table “preclusion” is created to save which module is which module’s preclusion. (It’s a relation of an instance of “modules” and another instance of “modules”.)

CREATE TABLE preclusion(  
module VARCHAR (16),  
excludedModule VARCHAR (16),  
FOREIGN KEY (module) REFERENCES modules(moduleCode) ON DELETE CASCADE,  
FOREIGN KEY (excludedModule) REFERENCES modules(moduleCode) ON DELETE CASCADE,  
PRIMARY KEY (module, excludedModule)  
);

### **ModulesTime:**

The table “modulesTime” is created to save different timeslots of a module that is able to be bided by the student.

CREATE TABLE modulesTime (  
moduleCode VARCHAR (16),  
startTime INT   
CHECK(  
(FLOOR(startTime/100)<=23) AND   
(FLOOR(startTime/100)>=0) AND   
(MOD(startTime,100)>=0) AND   
(MOD(startTime,100)<=59)),  
endTime INT   
CHECK(  
(FLOOR(endTime/100)<=23) AND   
(FLOOR(endTime/100)>=0) AND   
(MOD(endTime,100)>=0) AND   
(MOD(endTime,100)<=59)),  
day CHAR (3)   
CHECK (  
lower(day) LIKE ('mon') OR  
lower(day) LIKE ('tue') OR  
lower(day) LIKE ('wed') OR  
lower(day) LIKE ('thu') OR  
lower(day) LIKE ('fri') OR  
lower(day) LIKE ('sat') OR  
lower(day) LIKE ('sun') ),  
maxVacancy INT NOT NULL,  
FOREIGN KEY (moduleCode) REFERENCES modules (moduleCode) ON DELETE CASCADE,  
PRIMARY KEY (moduleCode, startTime, endTime, day)  
);

### **Users:**

The table “users” is created to save the details of a user.

CREATE TABLE users (  
matricNo VARCHAR(10),  
admin INT DEFAULT '0' CHECK (admin = 0 OR admin = 1),  
name varchar(64) NOT NULL,  
points INT NOT NULL,  
openId INT DEFAULT '1' CHECK (openId = 0 OR openId = 1),  
password CHAR (64),  
PRIMARY KEY (matricNo)  
);

### **Passed:**

The table “passed” is created to record down whether the students have passed the modules or not. (It is the relation between an instance of “users” and an instance of “modules”.)

CREATE TABLE passed (  
matricNo VARCHAR(10),  
moduleCode VARCHAR(16),  
FOREIGN KEY (matricNo) REFERENCES users(matricNo) ON DELETE CASCADE,  
FOREIGN KEY (moduleCode) REFERENCES modules(moduleCode) ON DELETE CASCADE,  
PRIMARY KEY (matricNo, moduleCode)  
);

### **Selected:**

The table “selected” is created to record down which modules are selected by the students during the bidding session. (Relation between an instance of “student” and an instance of “modulesTime”.)

CREATE TABLE selected(  
matricNo VARCHAR(10),  
moduleCode VARCHAR(16),  
startTime INT,  
endTime INT,  
day CHAR(3),  
bidpoints INT NOT NULL,  
bidTime TIMESTAMP NOT NULL,  
success INT DEFAULT '0' NOT NULL CHECK (success = 0 OR success = 1),  
FOREIGN KEY (matricNo) REFERENCES users(matricNo) ON DELETE CASCADE,  
FOREIGN KEY (moduleCode, startTime, endTime, day) REFERENCES modulesTime(moduleCode, startTime, endTime, day) ON DELETE CASCADE,  
PRIMARY KEY (matricNo, moduleCode, startTime, endTime, day)  
);

### **Session Bit:**

The table “session bit” is created to record down the states of the bidding session. It is 0 if the bidding session is close and 1 if the bidding session is opened.

CREATE TABLE sessionBit (

sessionB INTEGER default '0',

PRIMARY KEY (sessionB)

);

## **Sample and representative SQL code**

### **Tracking of Bidding Session:**

To keep track of the bidding session, we actually use the sessionBit table, we insert the value into sessionBit table first then keep updating it:

* ”INSERT INTO sessionBIT VALUES ('0') ”

### **Show all modules data:**

The below statement helps to retrieve all the available modules data and display out at the administrator page:

* ”SELECT \* FROM modules order by moduleCode”

### **Add new modules data:**

The below statement helps to add new module data base on module code and module name into the database:

* "INSERT INTO modules values('$varmoduleCode','$varmoduleName')"

### **Remove a timeslot:**

The below statement removes a data or t-uple from modulesTime, which is a table where module available timeslots are kept, based on the array $piece[], which consists of many values.

* “DELETE FROM modulesTime where moduleCode='$pieces[0]'and startTime='$pieces[1]' and endTime='$pieces[2]' and day='$pieces[3]'"

### **Prerequisite testing:**

The following statement is written to check if the student ($matric) met the requirements to add a module ($moduleCode).

* " SELECT 'True'

FROM prerequisite p

WHERE EXISTS(

SELECT p2.andId

FROM prerequisite p2

WHERE p2.module = p.module

AND p2.andId = p.andId

AND p2.requiredModule IN (

SELECT ps.moduleCode

FROM passed ps

WHERE ps.matricNo = '$matric'

)

GROUP BY p2.andId

HAVING COUNT(p2.requiredModule) = (

SELECT COUNT(p3.requiredModule)

FROM prerequisite p3

WHERE p3.module = p.module

AND p3.andId = p.andId

)

)

AND p.module = '$moduleCode'"

### **Showing bidding statistics:**

The following statement is written to get all the necessary information to show all the bidding statistics.

* " SELECT s.moduleCode, m.moduleName, s.startTime, s.endTime, s.day, s.bidpoints, s.bidTime,   
  COUNT(CASE WHEN s2.bidpoints > 0 AND s2.success = 0 THEN 1 END) AS NoBidder, mt.maxvacancy, MAX(s2.bidpoints) AS highestBidPts

FROM selected s, selected s2, modules m, modulestime mt

WHERE s.matricNo = '$matric'

AND s2.moduleCode = s.moduleCode

AND s2.startTime = s.startTime

AND s2.endTime = s.endTime

AND s2.day = s.day

AND s.success = '0'

AND s.moduleCode = m.moduleCode

AND s.moduleCode = mt.moduleCode

AND s.starttime = mt.starttime

AND s.endtime = mt.endtime

AND s.day = mt.day

GROUP BY s.moduleCode, m.moduleName, s.startTime, s.endTime, s.day, s.bidpoints, s.bidTime, mt.maxvacancy

ORDER BY s.moduleCode"

### **Generating bidding result:**

In order to find out which student has successfully bid the result, we have to compare the value of bidpoints and bidtime: the one who has the higher bidpoints won, and if their bidpoints are the same, the one who bids earlier wons. (The $mc is the module code, $st is the start time, $et is the end time and $d is the day, they are the primary key of moduleTimes.)

* "SELECT s2.matricNo as MN2, s2.moduleCode as MC2,

s2.startTime as ST2, s2.endTime as ET2, s2.day as D2

FROM ( SELECT \*

FROM selected sTemp

WHERE sTemp.bidpoints > 0

AND sTemp.moduleCode = '" . $mc . "'

AND sTemp.startTime = '" . $st . "'

AND sTemp.endTime = '" . $et . "'

AND sTemp.day = '" . $d . "'

ORDER BY sTemp.bidpoints DESC, sTemp.bidTime) s2

WHERE ROWNUM <= (

SELECT mt.maxVacancy

FROM modulesTime mt

WHERE mt.moduleCode = s2.moduleCode

AND mt.startTime = s2.startTime

AND mt.endTime = s2.endTime

AND mt.day = s2.day)"

### **Showing bidding result:**

The following statement is written to get all the necessary information to show all the bidding history.

* " (SELECT m.moduleName, mt.moduleCode, mt.startTime, mt.endTime,

mt.day, mt.maxVacancy, COUNT (s.matricNo), MAX(s.bidpoints), MIN(s.bidpoints)

FROM modules m, modulesTime mt, selected s

WHERE m.moduleCode = mt.moduleCode

AND mt.moduleCode = s.moduleCode

AND mt.startTime = s.startTime

AND mt.endTime = s.endTime

AND mt.day = s.day

GROUP BY m.moduleName, mt.moduleCode, mt.startTime, mt.endTime, mt.day, mt.maxVacancy)

UNION

(SELECT m2.moduleName, mt2.moduleCode, mt2.startTime,

mt2.endTime, mt2.day, mt2.maxVacancy, 0, 0, 0

FROM modules m2, modulesTime mt2

WHERE m2.moduleCode = mt2.moduleCode

AND NOT EXISTS (

SELECT \*

FROM selected s2

WHERE s2.moduleCode = mt2.moduleCode

AND s2.startTime = mt2.startTime

AND s2.endTime = mt2.endTime

AND s2.day = mt2.day

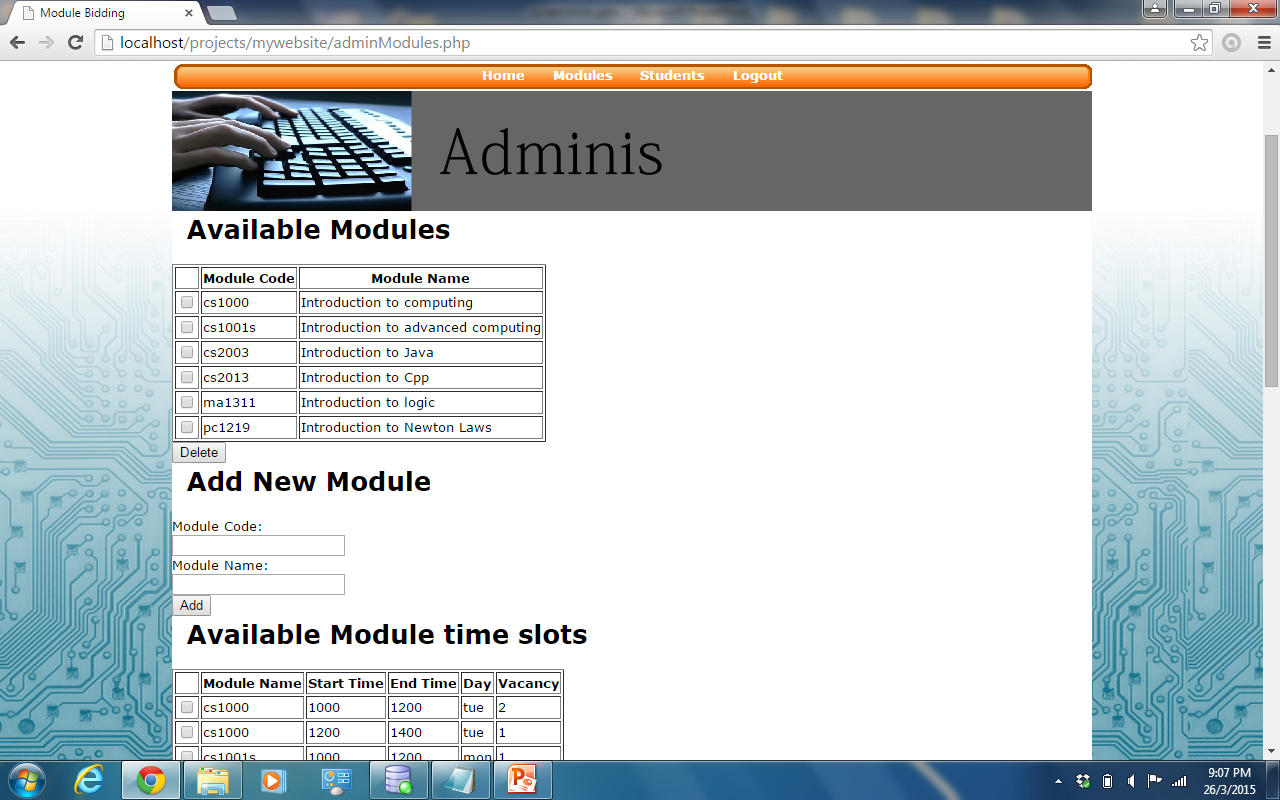
)) "

## **Web interface screenshots**

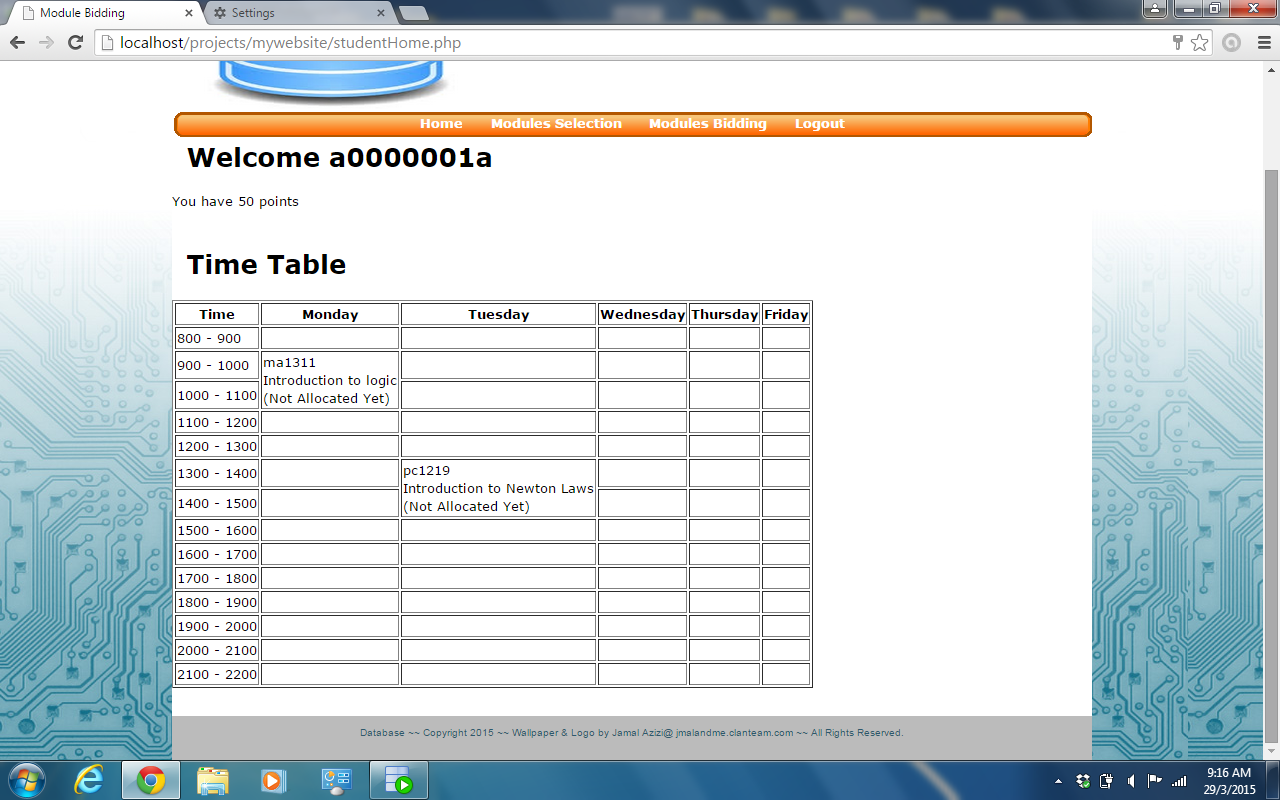
### **Admin: Students Page**

****

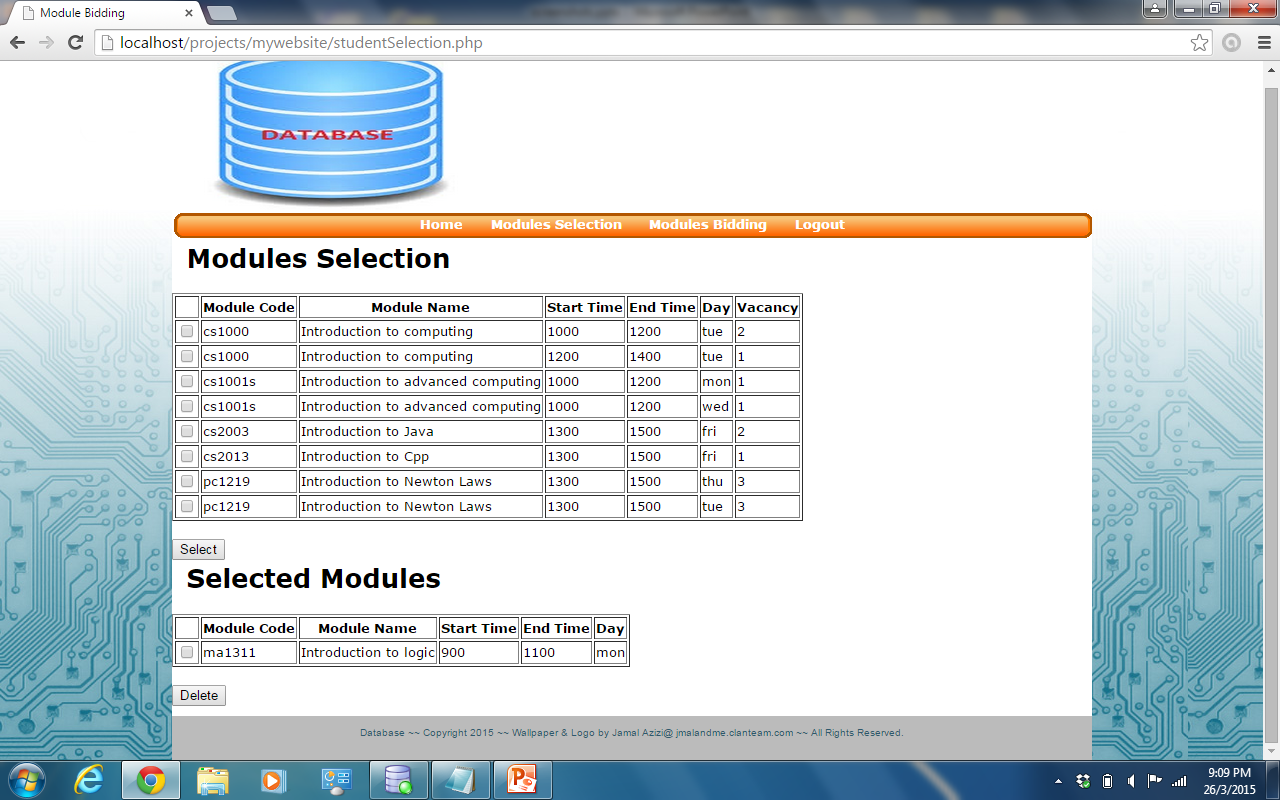
### **Admin: Modules Page**

****

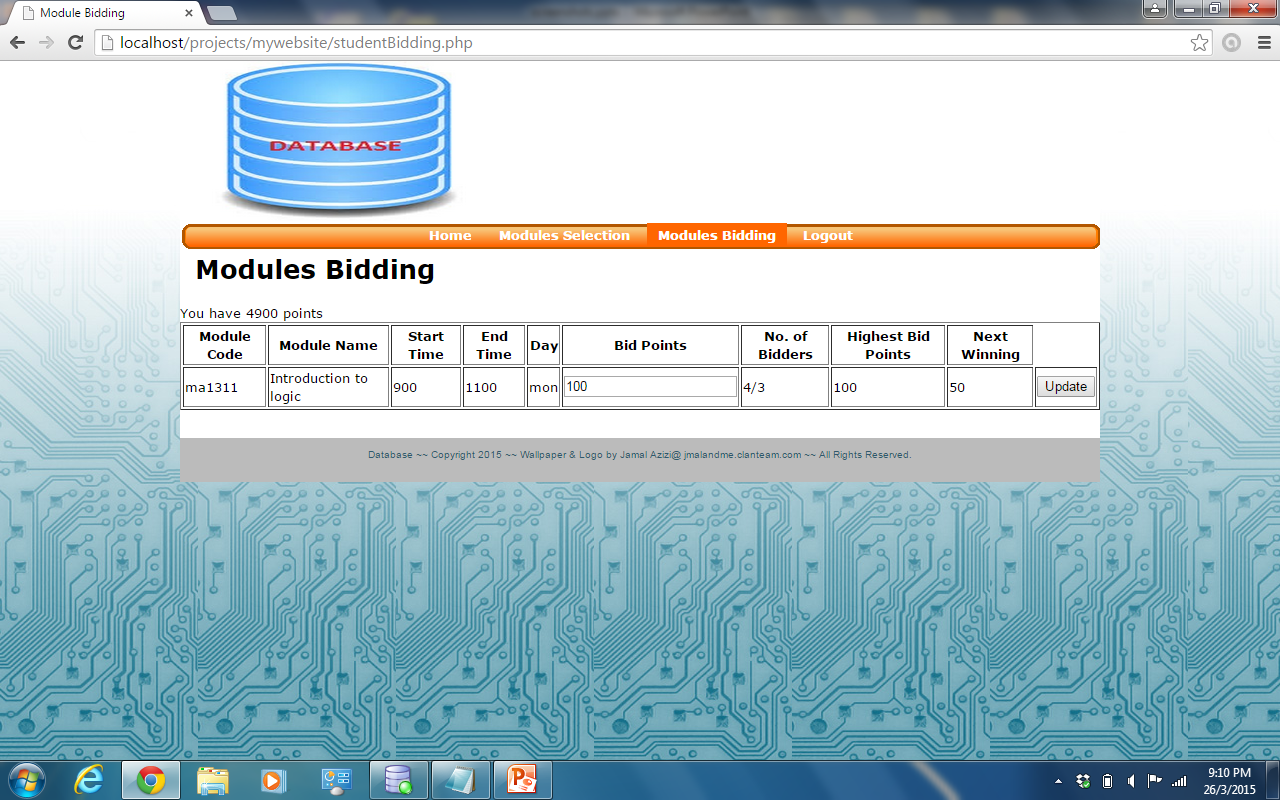
### **Student: Home Page**



### **Student: Modules Selection Page**

****

### **Student Modules Bidding Page**

****

## **Special Thanks To:**

[LightOpenID](https://github.com/iignatov/LightOpenID): We used it to implement Open Id.

<http://www.w3schools.com/>

<http://www.iconshock.com/img/product/IS_clean_database_1.jpg>

<http://php.net/manual/en/language.basic-syntax.php>

<http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/oow10/php_db/php_db.htm>

<http://php.net/manual/en/function.oci-connect.php>

<https://www.youtube.com/watch?v=hMhkD43yUzI>

<https://www.youtube.com/watch?v=IJ10-XCR4H4>

<https://www.youtube.com/watch?v=Gb5QSw9eE2E>

<http://stackoverflow.com/questions/17627827/updating-sql-table-for-selected-check-box-values>

<http://stackoverflow.com/questions/14475096/delete-multiple-rows-by-selecting-checkboxes-using-php>

<http://www.tizag.com/htmlT/forms.php>

<http://stackoverflow.com/questions/18753180/passing-two-values-for-same-checkbox-field-in-get-form>

<http://www.html-form-guide.com/php-form/php-form-select.html>