[30 marks]

### **General Instructions:**

- You can refer to any offline resources already on your laptop, but you must disable all networking and Bluetooth connections during the test. You must not communicate with anyone via any means during the test.
- You can also refer to your own offline cheat sheets. You may not share your cheat sheets with other students during the test. Please be courteous and keep your cheat sheets in your designated seat area.
- Just before the test, you will be given instructions by the invigilator as to how to obtain resource files required for the lab test and how to submit your solutions.
- No questions will be entertained during the test. If necessary, make your own assumptions.
- You are allowed to use only standard PHP classes and functions in your solutions do not use any third party libraries.
- Use meaningful names for classes, methods, functions and variables, as well as indent your code correctly.
- You **MUST** include your name as author in the comments of all your submitted source files. Failure to do so WILL attract a penalty of up to **20%** of your score for the corresponding question.

For example, if your registered name is "KIM Jong Un" and email ID is kim.jongun.2018, include the following comment at the beginning of each source file you write.

```
/*
    Name: KIM Jong Un
    Email: kim.jongun.2018
*/
```

- You may wish to comment out the parts in your code which cause errors. But commented code will not be marked.
- Unless otherwise stated, you can assume that user inputs are in the correct format.
- Instructions are given for WAMP users considering default setting the setting that we support for this course. If you are using MAMP, you may need to make necessary changes by yourself (e.g., modifying connection information in ConnectionManager.php).

DO NOT TURN OVER UNTIL YOU ARE TOLD TO DO SO

## **Load Database & Configure Connection Manager**

#### Given:

- database/
  - o ConnectionManager.php
  - o LT2.sql

#### **Instructions:**

- Import LT2.sql into your local MySQL database (via PHPMyAdmin, MySQL WorkBench or by other means).
  - o It will create "2019It2" database and the following tables:
    - For **Question 1**: "response"
    - For Question 2: "employee", "spouse", "child"
    - For **Question 3**: no tables
  - o Check the contents of the above-mentioned tables.
    - This may help you in answering Questions 1 and 2 below.
- In **ConnectionManager.php**, verify that the username, password, port number are correct.

Trial Lab Test 2 (Set A)

Page 2 of 16

### Question 1: Survey (Difficulty Level: \*)

[ 9 marks ]

#### Given:

```
    q1/model/
        o common.php (complete)
        o Response.php (complete)
        o ResponseDAO.php (complete)
    q1/images/
        o sis.png (given)
    q1/
        o display.php (partial)
        o survey1.php (complete)
        o survey2.php (partial)
        o process survey.php (partial)
```

o statistics.php (partial)

#### Overview:

In SIS, each term is 16 weeks long and each lesson (of a typical module) is three (3) hours long. Assume that SIS is looking to revamp this based on student preferences. In order to best understand student preferences with regards to **term length** (15 or 16 weeks) and **lesson duration** (2 or 3 hours), you are tasked to develop a web application for collecting student **survey responses**. Upon receiving the response from each student, the web application displays a summary, and it stores the response into a database.

The web application can also display all responses stored in the database and compute some statistics based on those responses.

#### **User Navigation Flow:**

```
    Part A: display.php
    Part B: survey1.php -> survey2.php -> process_survey.php
    Part C: statistics.php
```

Trial Lab Test 2 (Set A) Page 3 of 16

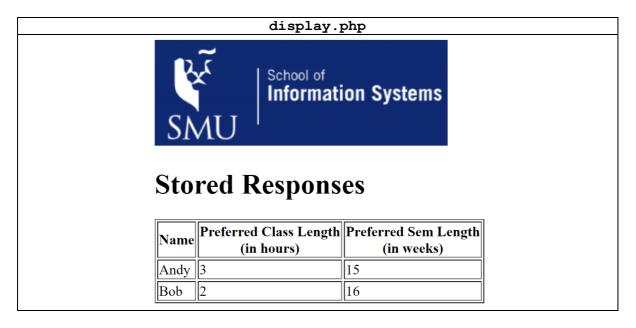
### Part A: Complete display.php (3 marks)

Update **display.php** to display responses that are stored in the database:

- Use retrieveAll() method of ResponseDAO to retrieve all records from table "response" in the
  database as an indexed array of Response objects. Loading LT2.sql given to you results in the creation of
  two records (a.k.a. rows) in table "response".
- Display the array of **Response** objects as a HTML table whose format is shown below:

Name	Preferred Class Length (in hours)	Preferred Sem Length (in weeks)
Andy	3	15
Bob	2	16

If done correctly, display.php should display this:



**Note:** The above content will change as the table "response" gets updated.

Trial Lab Test 2 (Set A) Page 4 of 16

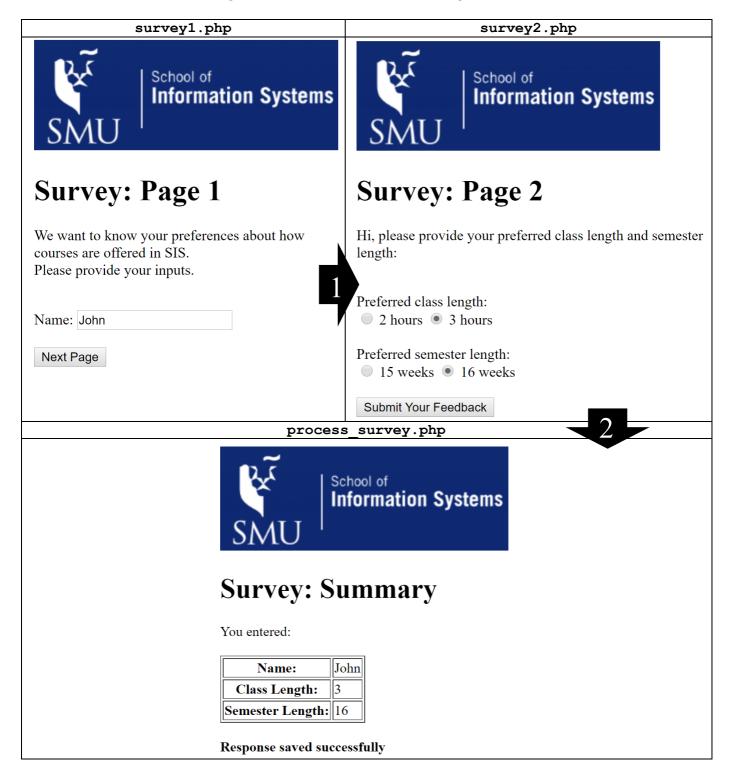
### Part B: Complete survey2.php and process survey.php (4 marks)

The survey has 3 pages:

- 1. **survey1.php** collects the name of a student (if it is provided by the student).
  - a. The student can choose to remain anonymous (by NOT typing in any name in the input text box).
- 2. **survey2.php** collects the preferred class length and semester length of the student.
- 3. **process\_survey.php** displays all the pieces of information entered by the student in an HTML table and stores them into the database.

The following shows the navigation flow with a sample set of user inputs.

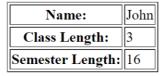
(Name = John; Preferred Class Length = 3 hours; Preferred Semester Length = 16 weeks):



Trial Lab Test 2 (Set A) Page 5 of 16

#### Your tasks are:

- 1. Get data passed through the 3 PHP pages:
  - a. Complete **survey2.php** so that the student name that is entered in **survey1.php** gets passed to **process survey.php**.
    - i. You may use \$\_SESSION or HTML form hidden input field as deemed necessary.
- 2. Display student name and preferences properly in process survey.php as an HTML table.
  - a. An example HTML table matching the navigation flow from above is as follows:



- 3. Add a new response to the database and display status:
  - a. Complete **process\_survey.php** such that user inputs are added as a new record to the table "response" in the database.
    - Use the addResponse(\$name,\$class\_length,\$sem\_length) method of ResponseDAO.
    - ii. The method returns a Boolean flag indicating whether the database insertion was successful or not.
  - b. Display a suitable message depending on the success/failure of the database insertion operation.
    - i. Echo "<strong>Response saved successfully</strong>" if the insertion was successful.
    - ii. Echo "<strong>Response was not saved successfully</strong>" if the insertion was not successful.

#### HINT:

- Use display.php created in Part A to check whether the insertion operation was done properly.
- Otherwise, check the status of the table "response" in your database using either PHPMyAdmin or MySQL Workbench (or by other means).

Trial Lab Test 2 (Set A) Page 6 of 16

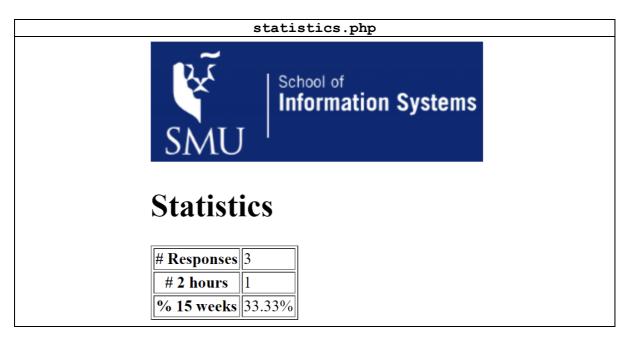
#### Part C: Complete statistics.php (2 marks)

Update **statistics.php** to display some statistics of the responses stored in the database as a HTML table:

- Use **retrieveAll()** method of **ResponseDAO** to retrieve all records from table "response" in the database as an indexed array of **Response** objects.
- Display some statistics of the array of **Response** objects as an HTML table with **THREE** rows whose format is shown below:
  - The first row (# Responses) corresponds to the total number of responses stored in the database table "response".
  - The second row (# 2 hours) corresponds to the total number of responses indicating "2 hours" as the preferred class length.
  - The third row (% 15 weeks) corresponds to the percentage (%) of responses indicating "15 weeks" as the preferred semester length (out of all responses). Use the built-in PHP function number\_format to format the percentage value to TWO decimal places.

# Responses	3
# 2 hours	1
% 15 weeks	33.33%

If done correctly, assuming that you have only added John (preferred class length = 3 hours; preferred semester length = 16 weeks) described in Part B, statistics.php should display this:



Note: The above will change for different contents of table "response".

Trial Lab Test 2 (Set A) Page 7 of 16

### Question 2: Employee Management (Difficulty Level: \*\*)

[15 marks]

#### Given:

#### Overview:

We want to build a web application that 1) allows retrieval of employees' data - employee ID, name of employee, name of spouse (if any), name and age of the children (if any) and 2) allows the update of an employee's password.

The design of the application:

- The application has TWO roles **Admin** and **User**.
- An employee is modelled as an Employee object.
- Spouse is modelled as a Spouse object.
- Children are modelled as an associative array with the <u>name</u> of the child as the key and <u>age</u> of the child as the value.

#### **User Navigation Flow:**

For users with **User role**:

login-view.html -> login.php -> viewDetails.php

This role will be allowed to view only his/her own data from the database.

For users with Admin role

login-view.html -> login.php -> viewDetails.php -> updatePassword.php
 -> process.php

This role will be allowed to a) view all employees' data from the database and b) update password for each employee.

Trial Lab Test 2 (Set A)

Page 8 of 16

### Part A: Complete login.php (3 marks)

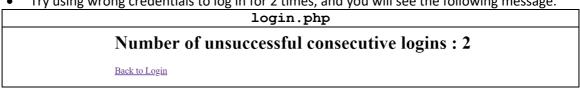
login-view.html page allows the user (employee) to log in using employee id and password.

login-view.html			
Login			
Employee I Password: Log In	[d:		

Upon clicking on "Log In" submit button, the employee is taken to login.php which performs employee id and password authentication.

In login.php, perform the following:

- 1. Retrieve the empId and password from login-view.html
- 2. Use authenticate (\$empId, \$password) method from EmployeeDAO to authenticate the empId and password
  - a. NOTE: For simplicity reasons, the password information IS NOT hashed in this question. It is stored in plain text. Please check the database table using PHPMyAdmin or MySQL WorkBench (or by other means).
- 3. If authenticate (\$empId, \$password) returns null (i.e., authentication is unsuccessful):
  - a. Initialize or update \$ SESSION['countUnsuccessful'] to keep track of the number of unsuccessful consecutive logins.
  - b. For example:
    - Try using wrong credentials to log in for 2 times, and you will see the following message.



- 4. Otherwise, if authenticate (SempId, Spassword) returns a String that denotes the role of the employee (i.e., authentication is successful):
  - a. Store the empId and role as \$ SESSION['empId'] and \$ SESSION['role'] respectively.
  - b. Unset \$ SESSION['countUnsuccessful'].
  - c. After that, redirect the user to viewDetails.php

Please use the following accounts for testing purposes:

Employee Id	Password	Role	
1	а	Admin	
2	b	Admin	
3	С	User	
4	d	User	
5	е	User	

HINT: If you are stuck in this part, hard code \$ SESSION['empId'] and \$ SESSION['role'] following one of the above accounts (e.g., \$ SESSION['empId'] = 3 and \$ SESSION['role'] = "User") and redirect to viewDetails.php.

Trial Lab Test 2 (Set A) Page 9 of 16

### Part B: Complete viewDetails.php and getAllEmployees() of EmployeeDAO.php (7 marks)

When the employee is redirected to **viewDetails.php**, an HTML table with different details will be displayed based on the **role** (Admin or User) of the employee.

- Get the role of the employee from \$ SESSION['role'] created in login.php
- Display the following HTML table for an employee with the User role (e.g. if you login with Employee Id: 3).
  - o For the <u>User</u> **role**, there will be no further functions for him/her to perform.
  - o Thus, a message will be displayed that he/she has been successfully logged out too.

vi	viewDetails.php			
Employee ID	Name	Spouse	Child	
3	Cindy	Charlie	Chloe: 4 Corrine: 8 Charmaine: 3	
You are logged	lout			

- Display the following for an employee with an <u>Admin</u> role (e.g. if you login with Employee Id: 1).
  - The main difference between the two HTML tables is that the employee with the <u>Admin</u> **role** will see the information for all employees and also an additional column (**Password**).

viewDetails.php				
<b>Employee ID</b>	Name	Spouse	Child	Password
1	Andrew	Amanda	Alan: 8 Adeline: 3 Ada: 5 Alison: 10	<u>a</u>
2	Bob	None	None	<u>b</u>
3	Cindy	Charlie	Chloe: 4 Corrine: 8 Charmaine: 3	<u>c</u>
4	Danny	None	None	<u>d</u>
5	Elvis	Elle	None	<u>e</u>

# Your tasks for <u>User</u> role:

- 1. Retrieve empld from \$ SESSION['empld'] created in login.php
- 2. Get an **Employee** object corresponding to **empld** by calling **getEmployee** (**\$empld**) method of **EmployeeDAO**
- 3. Display an HTML table containing information about the employee (see above).
  - a. Use **getSpouse** (**\$empId**) method from **EmployeeDAO** to get spouse information.
    - Use **getSpouseName** () method of **Spouse** to get spouse name.
    - Display "None" if the employee has no spouse.
  - b. Use **getChildren** (\$empId) method from **EmployeeDAO** to get children information.
    - The method returns an associative array with the **childName** as the key and **childAge** as the value.
    - The employee may still have records of children even if he doesn't have a spouse.
    - Display "None" if the employee has no children.
  - c. Use getEmpId() and getName() methods from Employee to get other pieces of information.
- 4. Display "You are logged out message" and clear all data stored in \$\_SESSION.

Trial Lab Test 2 (Set A) Page 10 of 16

### Your tasks for Admin role:

- 1. Complete the getAllEmployees() method in EmployeeDAO.php
- 2. Get an indexed array of Employee objects by calling getAllEmployees() method of EmployeeDAO
- 3. Display a HTML table containing information about all the employees (see above)
  - a. Password column should display the current password as a link that will direct the employee to updatePassword.php by passing in the corresponding empId for the specific employee.

Employee Id	Password Link
1	updatePassword?empId=1
2	updatePassword?empId=2
3	updatePassword?empId=3
4	updatePassword?empId=4
5	updatePassword?empId=5

HINT: If you are stuck in this part, directly use one of the above links, e.g.,

localhost/lt2/q2/updatePassword?empId=1, on the address bar of your web browser to proceed to Part C.

Trial Lab Test 2 (Set A) Page 11 of 16

### Part C: Complete updatePassword.php (2 marks)

When the employee is redirected to **updatePassword.php**, the following will be displayed. It allows the user to update his password, which is a randomly generated string generated from the application.



In updatePassword.php, perform the following:

- 1. Display the details of the employee
- 2. Store the empId as a HTML Form Hidden Field
- 3. Create a function **generateRandomPassword()** to generate and return a string with 8 alphanumeric characters (containing numbers **and** both lower and upper case alphabets), you may assume that there are no special characters.
  - a. The string should be random and not with a specific pattern (e.g. 1<sup>st</sup> character is a random number, 2<sup>nd</sup> character is random alphabet, 3<sup>rd</sup> character is a random number, etc.).
    - o **NOTE**: Document your assumptions (if any) as comments in your PHP file
  - b. Use **generateRandomPassword()** and display the randomly generated password in the **newPassword** text field.
- 4. Clicking on the "Update" submit button will submit the form using **HTTP GET** containing the **newPassword** text field and the **empId** hidden field to **process.php**. We use HTTP GET to allow you to proceed to the next step in case you are stuck (see **HINT** below).

**HINT:** If you are stuck in this part, you can directly type, e.g.,

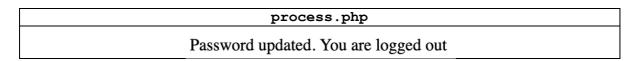
http://localhost/lt2/q2/process.php?empId=1&newPassword=123, on the address bar of your web browser to proceed to Part D.

Trial Lab Test 2 (Set A)

Page 12 of 16

### Part D: Complete process.php and updatePassword method of EmployeeDAO.php (2 marks)

When the employee is directed to **process.php**, the following will be displayed after the password has been updated in the database.



To complete this part, perform the following:

- 1. Code updatePassword(\$empId, \$new\_password) function in EmployeeDAO which returns a Boolean value.
- 2. Make use of updatePassword(\$empId, \$new\_password) to update the new password into the database.
- 3. Clear all data stored in \$\_SESSION.

# Part E: Complete protect.php (1 mark)

When an un-authenticated user (i.e., a user who has NOT successfully logged in or has logged out) visits:

- viewDetails.php
- updatePassword.php
- process.php

the user should automatically be redirected back to login-view.html

**NOTE:** This part can be done independently from the other parts of this question. Do this part as the last step. Otherwise, the hints provided earlier that can enable you to proceed to the next part (in case you are stuck in one part) may no longer work.

Trial Lab Test 2 (Set A)

Page 13 of 16

#### Question 3: Room Allocation (Difficulty Level: \*\*\*)

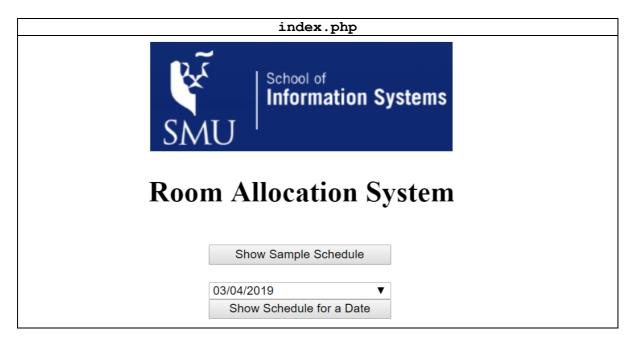
[6 marks]

#### Given:

- q3/classes/
  - o Lecture.php (complete)
  - o LectureDAO.php (complete)
  - o Schedule.php (complete)
- q3/images/
  - o sis.png (given)
- q3/tests/
  - o test-sample.php (given)
  - o test-common.php (given)
- q3/
  - o index.php (complete)
  - o display.php (partial; ONLY this file will be marked)

#### Overview:

This web application simulates a simple SIS Room Allocation System where 1) lectures are assigned to a minimum number of rooms and 2) more than one lecture cannot happen in the same room at the same time. The main web page (shown below) has two main functionalities -1) Show a sample schedule, 2) Show schedule for a given date.



#### **User Navigation Flow:**

• index.php -> display.php

### Part A: Complete display schedule (\$schedule) function in display.php (2 marks)

display\_schedule function displays the room allocation stored in \$schedule as a HTML table. Each column of the table corresponds to a room, and each row corresponds to a 1-hour time slot, starting at 9 to 17 (i.e., 9 AM to 5 PM). \$schedule is a Schedule object. This object has a property \$room\_allocation, which is an associative array containing room names as keys, and indexed arrays of Lecture objects as values.

**Example**: **display.php** should render the following webpage when "Show Sample Schedule" of **index.php** is clicked. It displays a schedule consisting of 10 lectures (L1 to L10) distributed across 3 rooms (Room-1 to Room-3).

Trial Lab Test 2 (Set A)

Page 14 of 16

# display.php School of **Information Systems** Sample Schedule Room-1 Room-2 Room-3 **09-10** L10 L1 10-11 L7 11-12 1.2 **12-13** L9 13-14 L6 ||L3 14-15 **15-16** L8 L5 L4 16-17 17-18

### NOTE:

- Do not change the name and the number of parameters of display schedule function.
- Your code will be tested using different **Schedule** objects and results will be visually inspected for grading.
- Each outcome is a binary one: Pass (i.e., perfect/expected table), or Fail
- Marks will be given based on the number **Pass** outcomes.
- If your code times out while processing a given **Schedule** object, the corresponding outcome will be considered a **Fail** case.

#### Part B: Complete create schedule (\$lectures) method in display.php (4 marks)

**create\_schedule** function takes **\$lectures**, which is an indexed array of **Lecture** objects. It assigns the **Lecture** objects to a number of rooms, such that the following <u>constraints</u> are met:

- [C1] Each Lecture object in \$lectures is assigned to a room
- [C2] The number of rooms is as small as possible
- [C3] Multiple lectures do not happen in the same room at the same time
- [C4] All Lecture objects assigned to the rooms appear in \$lectures

The resultant room allocation is an associative array containing room names as keys, and indexed arrays of **Lecture** objects as values. Room names start with "Room-", e.g., Room-1, Room-2, Room-3, etc. A *sample* room allocation with 3 rooms and 10 lectures is given below:

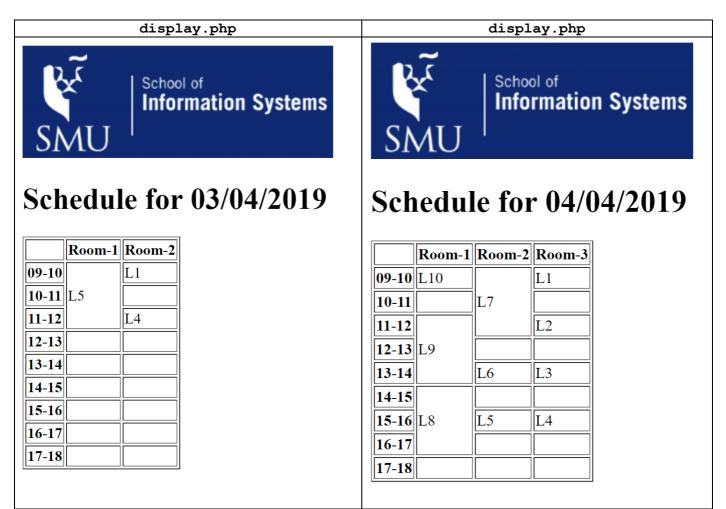
Store a room allocation as the **\$room\_allocation** property of a new **Schedule** object. Return this new **Schedule** object.

Trial Lab Test 2 (Set A)

Page 15 of 16

Page 16 of 16

**Example:** display.php should render the following pages when "Show Schedule for a Date" button is clicked in index.php with "03/04/2019" (*left*) and "04/04/2019" (*right*) selected respectively:



#### NOTE:

Trial Lab Test 2 (Set A)

- There may be **multiple** correct schedules that satisfy the four <u>constraints</u> (C1-C4) given above, and your solution needs to produce just **one** of such schedules.
- It is possible that the number of rooms is larger than 3.
- Do not change the name and the number of parameters of create schedule function.
- We will be using a set of test cases to automatically test **create schedule** function.
- A sample test case is included (q3/tests/test-sample.php). A successful run will produce the following outputs:

test-sample.php

Constraints C1 & C4: OK
Constraint C2: OK
Constraint C3: OK
Test Case Outcome (Sample): PASS

- Test cases that time out will be considered as *failed*.

**HINT:** You may find the built-in PHP function **uasort** useful. You may use other built-in functions as deemed necessary.