COMP3322 Modern Technologies on World Wide Web

Assignment Four

Total 16 points

Deadline: 17:00 April 17, 2023

Overview

You are going to develop a passwordless authentication application using PHP and MySQL database. A user has to use his/her email address to authenticate and access the server. Once authenticated, the server sends an email message to the user's email account with a token that contains the user's identity and a one-time secret. With this token and before the token is expired, the user is allowed to access pages in this Web application.

Objectives

1. A learning activity to support ILO 1 and ILO 2.
2. To practice how to use PHP, cookies, session control, and MySQL to create a passwordless web-based application.

Specification

You develop the application using the course's LAMP docker containers. (Note: another platform you can use for the development is the department's i7 Web server and sophia MySQL server.)

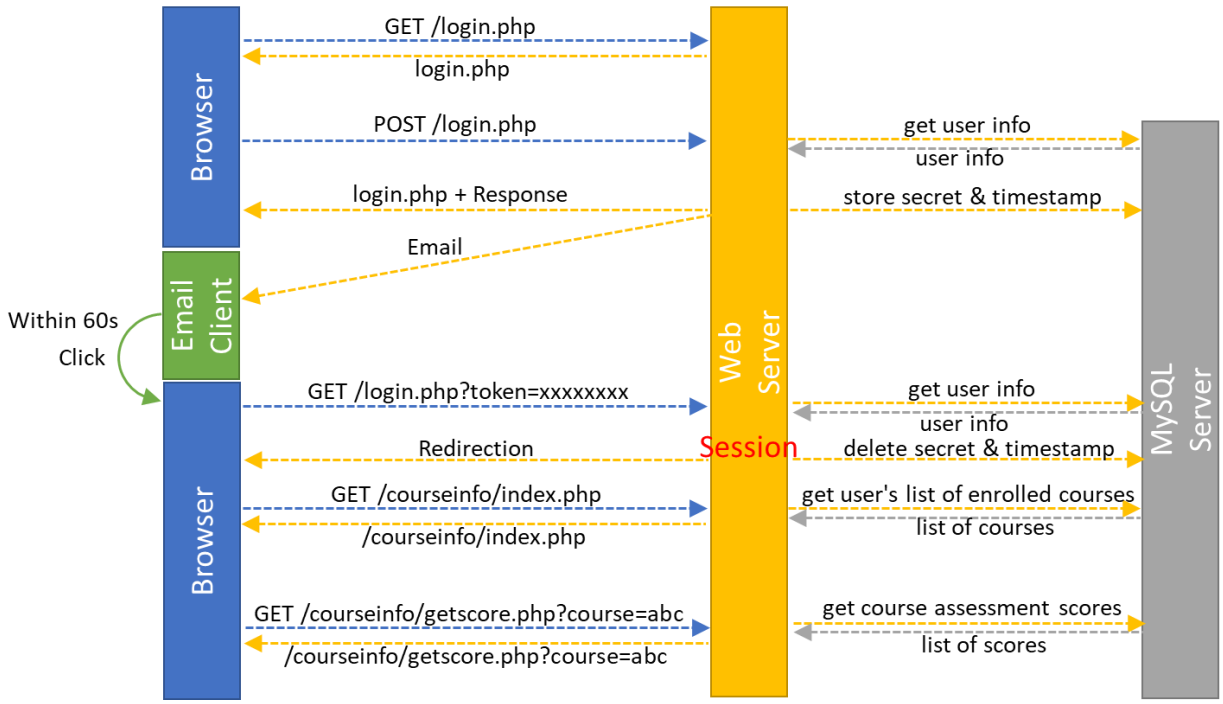
You will develop three PHP programs – login.php, index.php, and getscore.php and the corresponding CSS styling files. The files are placed in the public_html folder of the web server docker container with the following **directory structure**.

```
public_html
│
├── [PHPMailer]
│
├── [courseinfo]
│   ├── getscore.php
│   └── index.php
│
├── [styles]
│
└── login.php
```

This application **requires the PHPMailer library**, which is an external PHP library for sending emails from a web server. The PHPMailer folder contains all the PHP code for sending emails. The styles folder contains all the CSS styling files for this application.

Other than the PHPMailer library, you are **not allowed to use other external libraries** for this assignment.

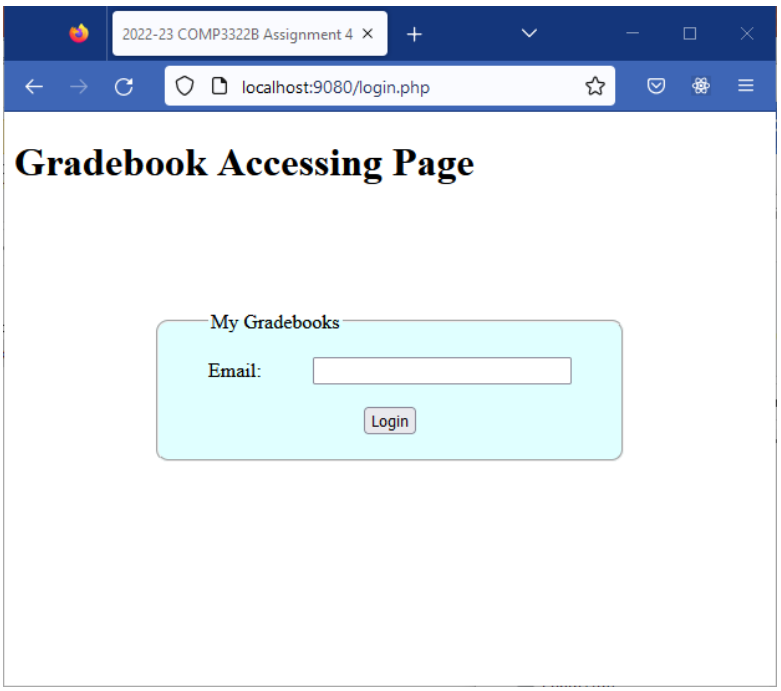
The following diagram represents the data flow between the client, the web server, and the MySQL server.



login.php

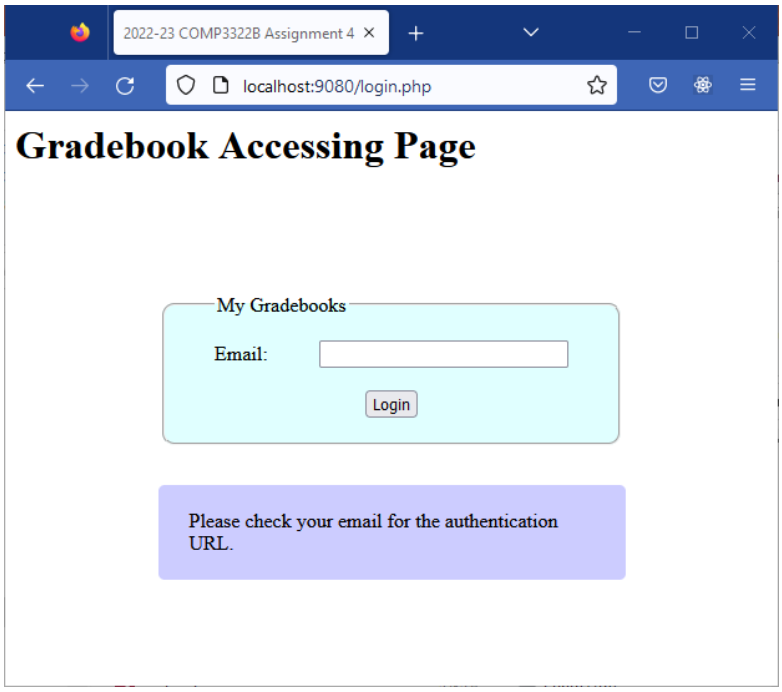
Implement the login.php program to handle the GET /login.php, POST /login.php, and GET /login.php?token=xxxxxxxx requests.

The login.php program is mainly for the users to authenticate and access internal pages. To access the login page, the client sends the GET /login.php request and the server sends back the response as follows.



The user **enters his/her email address** and clicks the 'Login' button to send the POST /login.php request for obtaining the accessing token in the form of URL by email. Based on the submitted email address, **the server checks whether the user has an account in its database**. If the user is in the

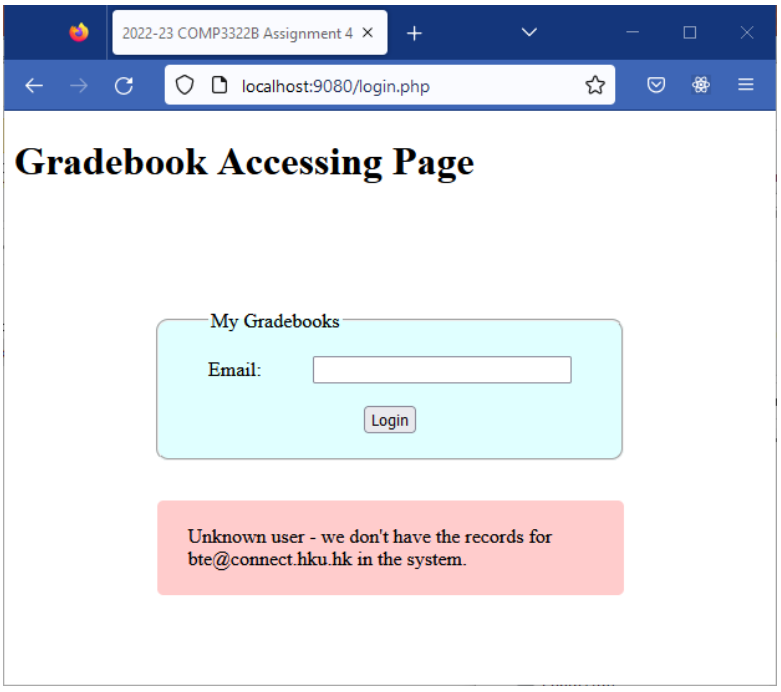
database, the server **responds with the login.php page with a positive message** as well as **sends an email that carries the authentication token** to the user's email account.



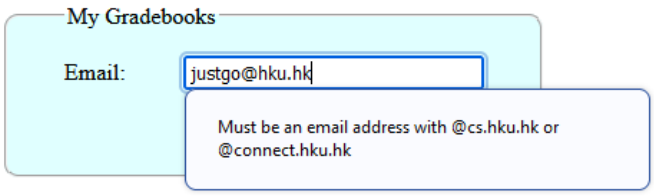
Dear Student,

You can log on to the system via the following link:
http://localhost:9080/login.php?token=7b22756964223a22313033222c22736563726574223a22343663663161366135633337393662652227d

Otherwise, the server responds with the login.php page with a negative response.

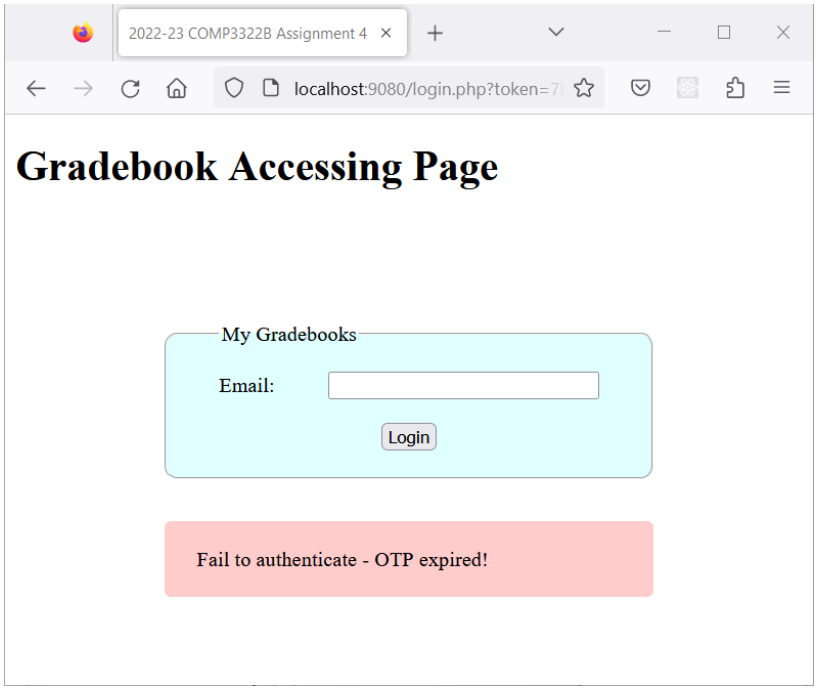


Before submitting the POST request, the page should **check whether the user has entered a valid email address** with the domain must be from **either @cs.hku.hk or @connect.hku.hk**. If the input is not a valid email or domain, it should show the hint to the user.



Once the user clicks on the authentication link, this triggers the browser to send the GET /login.php?token=xxxxxxxx request to the web server. Each authentication token has **a time limit**, which is only **valid for 60 seconds**. The user has to access the system using this URL before the token expires. If the GET request arrives within 60 seconds and the token carries a valid secret for that user, the server should respond with the /courseinfo/index.php page (**by redirection**). (Please read the 'index.php & getscore.php' section for the screenshot of the index.php page.)

When the user accesses the system with an expired authentication token, the server should respond with a negative response.



The login.php page should perform the following checking:

- The user's email is not in the database (user table), the system responds with the negative message: "Unknown user - we don't have the records for ????@#####.hku.hk in the system."
- The token has expired after 60 seconds, the system responds with the negative message: "Fail to authenticate - OTP expired!"
- The token carries an incorrect secret for that user, the system responds with the negative message: "Fail to authenticate - incorrect secret!"
- The token carries a credential for an unknown user, the system responds with the negative message: "Unknown user - cannot identify the student."

User database

You should create a table in the database to store the user's authentication email, a one-time secret, and a timestamp of the secret. **Only users who have their email addresses registered in the system be allowed to access all internal pages**. Here is an example user table (and you can download a copy from the Moodle site).

uid (key)	***email***	***secret***	***timestamp***
101	jummy@cs.hku.hk		
102	nobody@connect.hku.hk		
103	dummy@cs.hku.hk		
104	happy@connect.hku.hk		

To test the program, **you should rename two accounts** in this table with your HKU portal email and CS account email. This allows you to test the program by sending emails to these two accounts.

Authentication token

The authentication token should **contain the user identity and a one-time secret**. Here is a **suggested implementation** of the token for this assignment.

You can generate an one-time secret by using the PHP built-in function **random_bytes()**. This function generates cryptographically secure pseudo-random bytes with the length defined by the input argument. Then convert the random sequence to the hexadecimal format by using the **bin2hex()** function. **Suggestion:** create a 16-byte secret by calling bin2hex(random_bytes(8)).

You can generate the token by creating an associative array that contains the user identity and the one-time secret. **Suggestion:** ['uid': 105, 'secret': '808fc44d325ba361']. Then **encode** the associative array to a JSON string and convert it to the hexadecimal format by using bin2hex(). The token becomes a long random sequence of bytes.

When received an authentication token, the server can use **hex2bin()** to decode the token back to the JSON string and convert it to the associative array for getting the user identity and the one-time secret. Use the user identity to retrieve the stored secret and the timestamp. Then determine whether the token carries a matching secret and the secret hasn't expired.

PHPMailer

Your program should use the PHPMailer library to send emails to users. **Please make sure that you only send emails to your HKU Portal email and CS account email when testing your program.**

To send emails using the department SMTP server - testmail.cs.hku.hk, **your computer must connect to the HKUVPN network**. Otherwise, your computer cannot reach testmail.cs.hku.hk as it is protected inside the CS firewall. The server testmail.cs.hku.hk only accepts emails sending to @cs.hku.hk and @connect.hku.hk email accounts. It rejects other email domains such as @gmail.com.

You should download **a sample program – mailer.php** from the course's Moodle page to test the connection between your computer and the testmail.cs.hku.hk server. Place the mailer.php program in the public_html folder, and type the following URL to the address bar of the browser for sending the email:

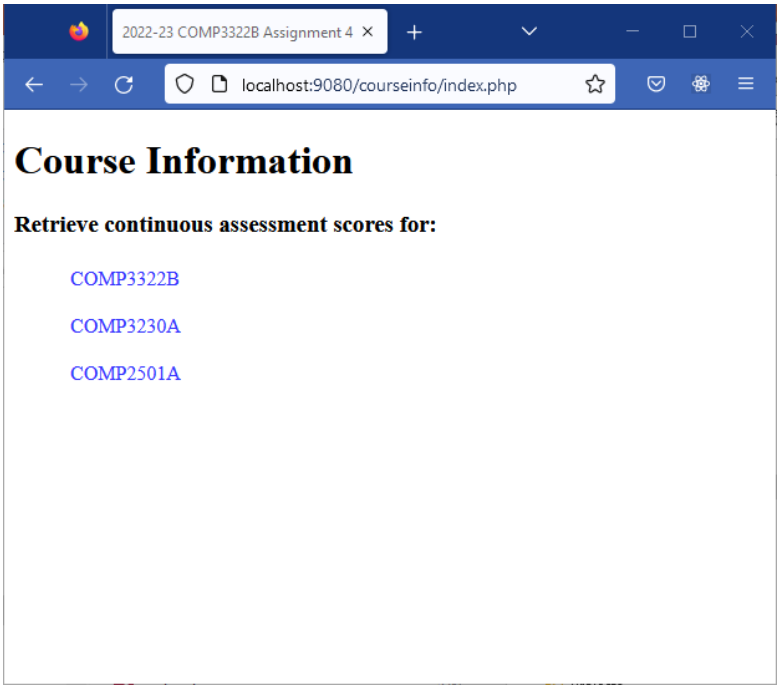
http://localhost:9080/mailer.php?to=tmchan@cs.hku.hk&name=Terry

Remember to change "tmchan@cs.hku.hk" to your email address and "Terry" to your name.

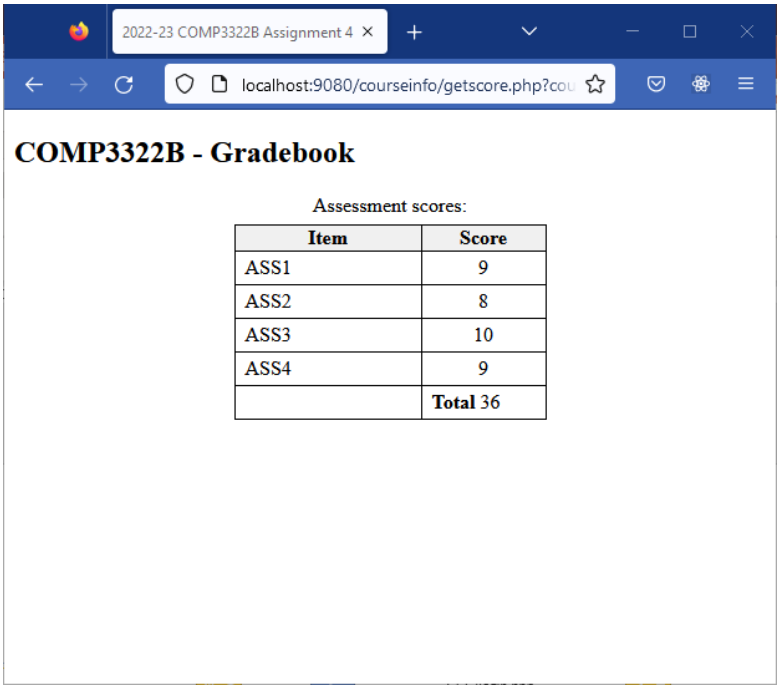
You can reuse most of the code in the mailer.php program for setting up and sending emails in your login.php program. Therefore, please read the source code of mailer.php to learn how it works.

index.php & getscore.php

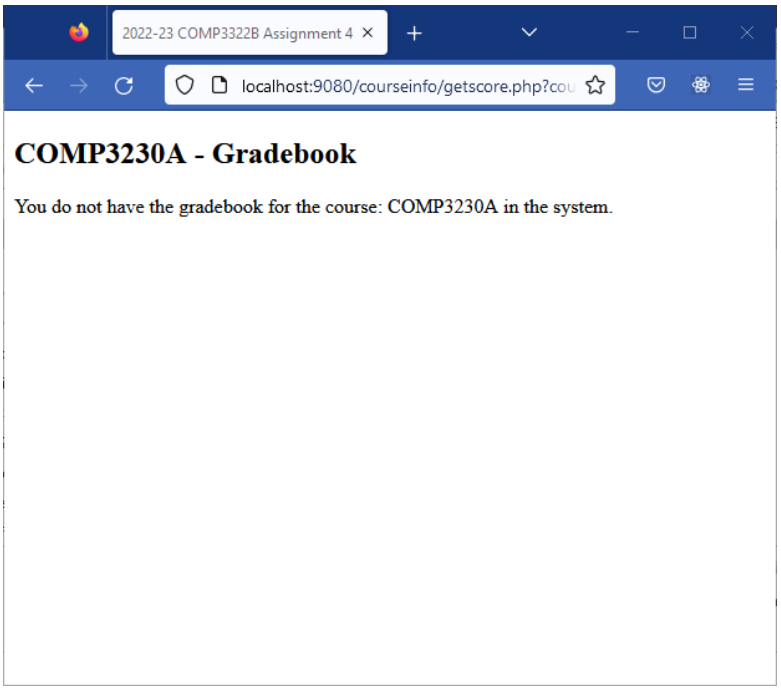
The programs **should apply session control** to check whether the user has successfully authenticated before returning these internal pages. If the user has successfully authenticated and the session hasn't expired, the **index.php page** should return the list of courses that the user has the assessment records in the database (the courseinfo table). Here is an example screenshot of the index.php page for user 101.



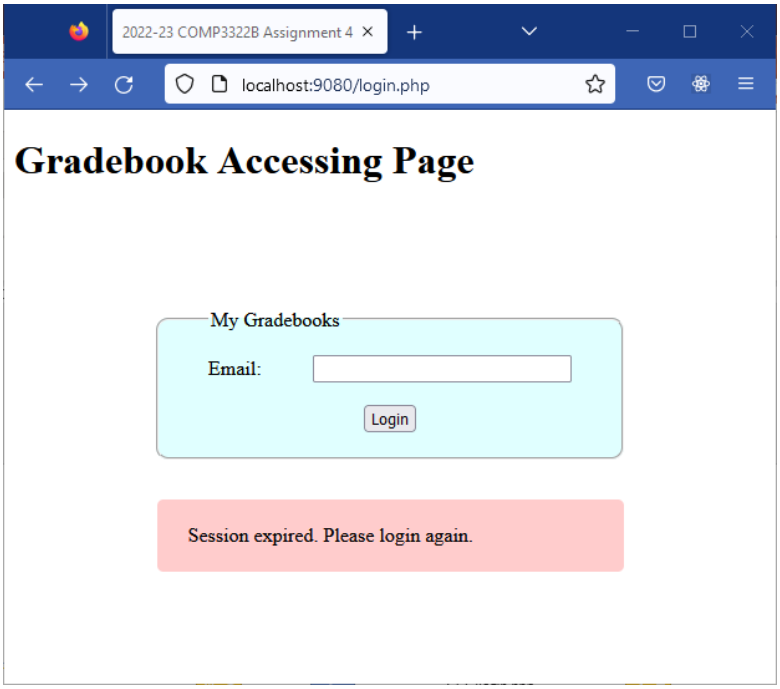
When the user clicks on one of the course links, this triggers the GET /courseinfo/getscore.php?course=abc request. Upon receiving this request, if the session hasn't expired, **the server retrieves all assessment records for the user in the courseinfo table for the course 'abc'**. Then it **returns the data in a tabular format** as follows.



When the system receives a request for a course that the user does not have the record in the database, it should return the following message.



The system **sets a session time limit to 300 seconds**. Within 300 seconds, the user is allowed to access these two pages as many times as the user wants. After 300 seconds, when the user accesses either /courseinfo/index.php or /courseinfo/getscore.php, the system **should redirect to the login.php page with the negative message**: “Session expired. Please login again.”



Courseinfo database

You should create a table in the database to store the assessment records of the courses. Here is an example courseinfo table (and you can download a copy from the Moodle site). The field 'uid' is the foreign key for accessing the user table.

id (key)	***uid***	***course***	***assign***	***score***
1	101	COMP3322B	ASS1	9
2	101	COMP3322B	ASS2	8
3	101	COMP3322B	ASS3	10
4	101	COMP3322B	ASS4	9
5	102	COMP3322B	ASS1	6
6	102	COMP3322B	ASS2	6
7	102	COMP3322B	ASS3	6
8	102	COMP3322B	ASS4	8
9	103	COMP3234B	Prob-set1	7
10	103	COMP3234B	Prob-set2	5
11	103	COMP3234B	Prob-set3	7
12	103	COMP3234B	Prog1	13
13	103	COMP3234B	Midterm	10
14	101	COMP3230A	Prob-set1	4
15	101	COMP3230A	Prob-set2	5
16	101	COMP3230A	Prob-set3	6
17	101	COMP3230A	Prog1	10
18	101	COMP3230A	Prog2	10
19	101	COMP3230A	Midterm	7
20	104	COMP3234B	Prob-set1	2
21	104	COMP3234B	Prob-set2	3
22	104	COMP3234B	Prob-set3	5

23	104	COMP3234B	Prog1	8
24	104	COMP3234B	Midterm	5
25	101	COMP2501A	PASS1	5
26	101	COMP2501A	ESSAY	13
27	101	COMP2501A	PASS2	5
28	101	COMP2501A	Midterm	10
29	101	COMP2501A	PASS3	4
30	104	COMP2501A	PASS1	6
31	104	COMP2501A	ESSAY	10
32	104	COMP2501A	PASS2	6
33	104	COMP2501A	Midterm	12
34	104	COMP2501A	PASS3	4

Resources

You are provided with the following files.

- `PHPMailer.zip` – this zipped file contains all PHPMailer code for sending emails. Place this zip file in the public_html folder and **unzip it** to exact the PHPMailer folder.
- `Mailer.php` – this is the sample PHPMailer program. You should place it in the public_html folder and test the connection between your computer and the testmail.cs.hku.hk server. Remember to connect to HKUVPN first.
- `user.sql` – this sql file is for creating the user table in the db3322 database in the c3322-db server (mysql docker container).
- `courseinfo.sql` – this sql file is for creating the courseinfo table in the db3322 database in the c3322-db server (mysql docker container)

Testing platform

We shall run the server program in the LAMP container set and use curl and Firefox to test the programs.

Submission

Please finish this assignment before Monday April 17 17:00. Submit the following files:

1. login.php
2. index.php
3. getscore.php
4. any other php files
5. any CSS styling files
6. If you use a different user table and/or courseinfo table, export a copy of the table from your mysql database and submit the copies to Moodle.

When working on the grading, we shall place the index.php and getscore.php inside the 'courseinfo' folder and all CSS files inside the 'styles' folder.

Grading Policy

Points	Criteria
8.0	login.php

	▪ Correctly handle the GET /login.php, POST /login.php, and GET /login.php?token=xxxxxxxx request ▪ Check the user input (at client-side and server-side) ▪ Successfully send the authentication emails ▪ Correctly detect and handle those error situations (i.e., display appropriate messages)
3.0	Session control ▪ Correctly set up session and allow authenticated users to access all internal pages ▪ Correctly detect session timeout and redirect the user to login.php page with a negative message. ▪ Correctly reject all requests when no active session
2.5	index.php ▪ Correctly display the course list for the user
2.5	getscore.php ▪ Correctly display the assessment scores and the total for the selected course ▪ Error handling
-4.0	Using any external libraries.

Plagiarism

Plagiarism is a very serious academic offence. Students should understand what constitutes plagiarism, the consequences of committing an offence of plagiarism, and how to avoid it. ***Please note that we may request you to explain to us how your program is functioning as well as we may also make use of software tools to detect software plagiarism.***