VIRTUAL LEARNING

GC06 Database and Information Management System

Team 24

**Yusuke Kawano  
Mohamad Fadhli Ismail  
Izzatul Syazana Samsudin**

Contents

[1 Link to YouTube Video 2](#_Toc414639930)

[2 Entity Relationship Diagram (ERD) 2](#_Toc414639931)

[3 Database Schema 2](#_Toc414639932)

[4 Third Normal Form 2](#_Toc414639933)

[5 List of Queries 3](#_Toc414639934)

[5.1 Part 1 3](#_Toc414639935)

[5.2 Part 2 5](#_Toc414639936)

[**5.3** **Part 3** 8](#_Toc414639937)

# The Video

A video on how our prototype work can be accessed through the following link. However, due to lack of microphone when recording the video results in low sound quality.

Virtual Learning Group 24 - <http://youtu.be/7QQLZy7MdZU>

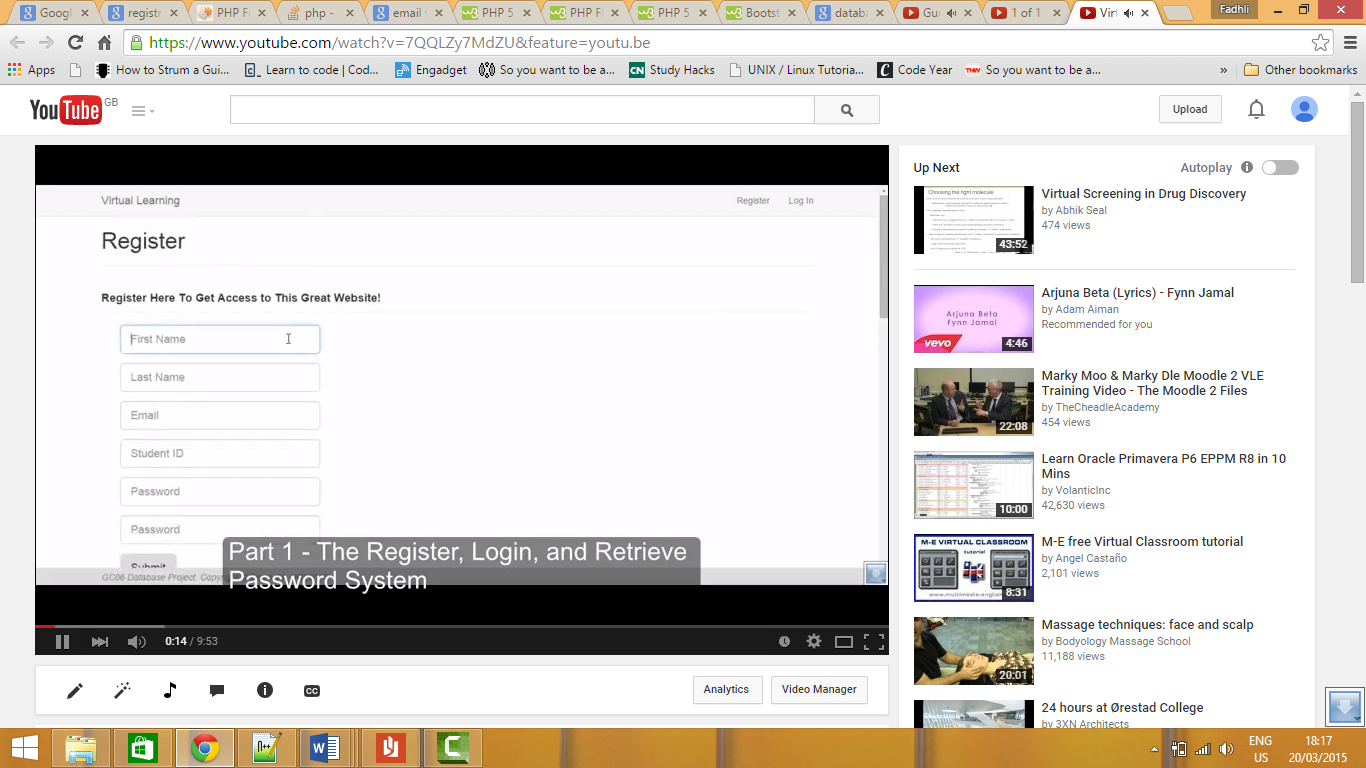


Figure 1 – A screenshot of the video in YouTube

# Entity Relationship Diagram (ERD)

Assumptions made:

# Database Schema

Explanation how it translates the ER diagram

# Third Normal Form

Analysis showing the database schema in 3rd normal form

# List of Queries

## Part 1

1. **Checking whether the student data is already in the database**

$checkStudentId = "SELECT StudentId FROM student WHERE StudentId = ?";

The main purpose of this query is to check whether the user has been registered as to prevent duplication of data in the database. The StudentId entered will be checked and validated with the database first before the user is allowed to register to prevent duplication. If there is a duplication, the user will be notified that the data is already in the database and will be redirected to retrievePasswordPage.php in case the user has forgot their password.

1. **Inserting the user’s information provided into the database**

$insertStudentToDb = "INSERT INTO student (FirstName, LastName, Email, User\_pass, StudentId,

user\_level) VALUES (?, ?, ?, ?, ?, ?)";

After checking whether the user is already registered with the previous query, this query will attempt to insert the data provided by the user into the student table in the database. The FirstName is the first name of the user. The LastName is the last name of the user. The Email, User\_pass, and StudentId are the email, the hashed user’s password, and the user’s student ID respectively.

Before inserting the data into the database, the password will be hashed and salted with password\_hash(). This hashed password will be inserted into the database, and as a safety precaution, the user’s unhashed or original password is not stored anywhere in the database. The user\_level is use to classify whether the user logged in is a student or an admin. At first registration, all the user will be given the student status, however, this can be change by the administrator which have access to the database.

1. **Retrieving information from the database for login purposes**

$queryLogin = "SELECT GroupNo, User\_pass, user\_level, FirstName, LastName

FROM student WHERE StudentId = ?";

This query will be use to retrieve all required information from the database for login purposes. Unlike the registration system, additional query is not needed to check whether the user is registered. Any return of rows from the query or any successful query means that the StudentId data is in the database. The password entered by the user will then be verified with password retrieved from the database using password\_verify.

Upon confirmation that the password and the ID matched, the users will be check whether they are a student or one of the admin based on their $user\_level data retrieved from the database. As said earlier, all users that registered will be given $user\_level of student. However, administrator have the right to change the $user\_level. If the user is a student, they will be redirected to profile.php page, while the admin will be redirected to admin.php page.

1. **Retrieving user password**

$queryPassword = "SELECT User\_pass, FirstName, LastName, Email

FROM student WHERE StudentId = ?";

The above query is to retrieve several data stored in the database for validation and emailing purposes. The FirstName and the LastName from the database will be checked with the data the users entered for added validation. The user can only change their password when they are logged in. The email will be used to send the new password to the users. The email will be sent to the email address the users used upon registration for security measures.

1. **Updating the users password**

$queryNewPswd = "UPDATE student SET User\_pass= ? WHERE StudentId = ?";

The hashed password from the database will be compared to the password entered by the users that will also be hashed. Upon successful comparison, the password in the database will be updated in the database.

1. **Retrieving hashed password and comparing with data provided by user**

$queryPassword = "SELECT User\_pass, FirstName, LastName, Email

FROM student WHERE StudentId = ?";

The above query is to retrieve the hashed password (User\_pass), the FirstName, LastName, Email from the database. As this is for retrieving password, more data is compared from what the user provide with the data in the database for safety measurement. A successful query means that the data is in the database. If not, the user will be prompt and redirected to the registrationPage.php. The first name and the last name which provided by the user will be compared to the FirstName and the LastName in the database respectively.

1. **Updating the password with random generated password**

$queryNewPswd = "UPDATE student SET User\_pass= ? WHERE StudentId = ?";

A random password will be generated and hashed before it is inserted into the database to replace the old hashed password. This is carried by this update query. An email will be sent to the user about the details of their ‘temporary’ password. The $Email (email) used is retrieved directly from the database as a safety measure. This password will be randomly generated and securely hashed with password\_hash(). The user can login using the new password and advised to directly change it as a safety measure.

## Part 2

1. **Displaying Student’s Group No & The Team Members**

SELECT FirstName, LastName, Email FROM `users` WHERE GroupNo = ?

This query is to display team member’s name in the group, which is shown on the home.php page. This is to allow user-student to easily contact their team members for the first meeting in case they do not know each other before.

1. **Displaying Student’s Ranking & Average Score**

SELECT GroupNo, AverageScore, Rank FROM

(SELECT GroupNo, AverageScore,

@curr\_rank := IF(@prev\_score = AverageScore, @curr\_rank, @incr\_rank) AS Rank,

@incr\_rank := @incr\_rank + 1,

@prev\_score := AverageScore

FROM `group` AS g,

(SELECT @curr\_rank :=0, @prev\_score := NULL, @incr\_rank := 1) AS r

ORDER BY AverageScore DESC) AS s WHERE GroupNo = ?

This query is to show user-student’s group ranking as well as the average score they received from the peer assessments. In this query, if the average score of a group is the same as the average score in the previous rank, the rank of the group will remain the same as the current rank. Otherwise, the current rank is incremented by 1. Although the current rank remains the same, the rank will still be incremented in general with this function @incr\_rank := @incr\_rank + 1, hence there will be gap between the rank. For example, if two groups received the same mark, they will receive the same rank, 8 for example. If a subsequent group has a different mark, this group rank will be 10 and not 9.

1. **Displaying Other Peers’ Average Score**

SELECT `Assessor` FROM `assessment` WHERE `Group\_to\_Assess` = ?;

SELECT `AverageScore` FROM `group` WHERE `GroupNo` = ?

The objective of this query is to display the marks of other groups who assess another group in order to provide confidence and reliability of the assessments made by the peers. The first query is to find out the groups who are assessing the specific group. As the results are fetched from the first query in a while-loop, the second query is invoked to fetch the average score of each group of assessors. The results are displayed in the home.php page.

1. **Upload XML File**

INSERT INTO report (GroupNo, File\_Link, File\_Name, File\_Size, File\_Type, Intro, Main, Conclusion) VALUES (?, ?, ?, ?, ?, ?, ?, ?)

ON DUPLICATE KEY UPDATE

File\_Link = ?, File\_Name = ?, File\_Size = ?, File\_Type = ?, Intro = ?, Main = ?, Conclusion = ?

This query runs when user upload an XML type file where the content of the XML file will be extracted to the database based on the element tag in the file. The presence of “ON DUPLICATE KEY UPDATE” in the query means that when the user re-uploads or resubmits their report, the value in the database will be updated if the `GroupNo` already exists. It will not insert a new row in the `report` table.

1. **Show Comment & Score Received**

SELECT score.CriteriaNo, score.Comment, score.Score\_Criteria

FROM score

INNER JOIN assessment

ON assessment.AssessmentNo = score.AssessmentNo

WHERE assessment.AssessmentNo = ?

The above query is executed to display the comments and scores received for each criteria. Each group has been assigned to assess three other groups. To ensure anonymous assessment, `AssessmentNo` notation is used, thus the group will not know which groups they are assessing. The `score` and `assessment` tables are joined where the `AssessmentNo` is the foreign key. The results is fetched by looping through the `AssessmentNo` for that particular group.

SELECT score.AssessmentNo, SUM(score.Score\_Criteria) as OverallScore, ROUND(SUM((score.Score\_Criteria/3)\*2)) AS AverageScore

FROM score

INNER JOIN assessment

ON assessment.AssessmentNo = score.AssessmentNo

WHERE assessment.Group\_to\_Assess = ?

GROUP BY assessment.AssessmentNo

This query calculates the total score for each criteria for each assessment, where the result is used to find the average score for all three assessments.

1. **Show Ranking and Average Score Received by All Groups**

SELECT GroupNo, AverageScore, Rank FROM

(SELECT GroupNo, AverageScore,

@curr\_rank := IF(@prev\_score = AverageScore, @curr\_rank, @incr\_rank) AS Rank,

@incr\_rank := @incr\_rank + 1,

@prev\_score := AverageScore

FROM `group` AS g,

(SELECT @curr\_rank :=0, @prev\_score := NULL, @incr\_rank := 1) AS r

ORDER BY AverageScore DESC) AS s

This query will rank each group based on their average score and list it on the student\_assessment.php page of the admin interface. The logic of this query is explained in details in section 2.2.

1. **Number of Created Groups**

SELECT COUNT(DISTINCT GroupNo) AS `countgroup` FROM users WHERE `GroupNo` !=0

The `users` table consists of list of registered users on the website and they are distinguished by the user level: student or admin. If the `GroupNo` is zero, it means the student has not been assigned a group yet or if the user is an admin. Since the `GroupNo` is not unique for each row, the aim of this query is to count the number of existing groups by utilising the “DISTINCT” function and display it so that the admin knows the number of groups created.

1. **Show Students in Respective Allocated Group**

SELECT `UserId`, `FirstName`, `LastName`, `GroupNo` FROM `users` WHERE `GroupNo` = ?

This query will display students in the group that they have been allocated on. In order to show this, the query is iterated equivalence to the number of groups that have been created, a value which is obtained from the query in section 2.7.

1. **Show List of Unsorted Students**

SELECT `UserId`, `FirstName`, `LastName`, `GroupNo` FROM `users` WHERE `GroupNo` = 0 AND `User\_Level` = 'student' ORDER BY `FirstName`

For users labelled as “student” with `GroupNo` = 0 in the `users` table, this means the student has not been sorted to a group yet. This will allow admin to allocate the group to the student.

1. **Update Student’s Group**

UPDATE `users` SET `GroupNo` = ? WHERE UserId = ?

The group allocation for student in the admin interface is done by dragging the name of the student into a box/container labelled with `GroupNo`. AJAX will processed the parameters, which are `GroupNo` and `UserId` and send to server, where it will update the `GroupNo` according to the specified `UserId`.

1. **Show List of Students**

SELECT `UserId`, `FirstName`, `LastName`, `Email`, `Login\_Id`, `GroupNo` FROM `users` WHERE `User\_Level` = 'student' ORDER BY `FirstName`

This query will list down all registered students in a table and arranged them in order of their first name. Admin will be able to filter `FirstName`, `LastName` and `Login\_Id` to search for a particular student.

1. **Update Student’s Details**

UPDATE `users` SET " .$name. " = ? WHERE UserId = ?

The variable $name represents the column in the `users` table. There are four components of the student details that the admin can amend: `FirstName`, `LastName`, `Email` & `GroupNo`. These are the values for $name. Therefore, this query is to update either of the aforementioned components for a student with the particular UserId.

## Part 3

1. **Assessing Peer’s Report (Review.php)**

$queryReportToAssess = "SELECT `Report\_to\_Assess`,`AssessmentNo` FROM `assessment` WHERE `GroupNo` = ?";

This query is to show report by selecting the value of Report\_to\_Assess number and Assessment number that exists on assessment table. In order to avoid showing all data, user can pass the group number that he would like to show. The Report\_to\_Assess number is used by implying the detail of report number.

1. **Grading criteria, writing comment and opening xml file (View\_report.php)**

$queryGroupNo = "SELECT `GroupNo` FROM `assessment` WHERE `AssessmentNo`= $id";

This query is to show group number, which corresponds with the assessment number of session information from assessment table. Following query ($queryFileName) uses a group number. Each assessment number is allocated a group number. Therefore if user exploits their own group number, it would generate wrong result.

$queryReport = "SELECT `File\_Name`, `GroupNo`, `Intro`, `Main`, `Conclusion` FROM `report` WHERE `GroupNo`= ?";

To specify which group’s file should be open, this query is to show file name and group numebr and three xml components. Group number is derived from former query and intro, main and conclusion are incorporated part of xml file. When user opens view\_report page, the specified xml file is read and shown on the html page. Therefore, user needs to submit their report on xml form.

$queryCriteria = "SELECT `CriteriaNo`,`Score\_Criteria`,`Comment` FROM `score` WHERE `AssessmentNo` = $id AND `CriteriaNo`=$i ";

User must input other team’s score from 0 to 10 marks. The default score is five but when user changes each scale, it automatically reflects and change the score. Also, they need to write down the comment why they give the score. If there is any space of comment, java script alert will be opened.

However, if user changes their mind after submitting their assessment, it would be hard to remember all score and comments. Thus, this query is to show the existing data if database has already given score and comments. Database must have five criteria and comments, so this query will be repeated five times and store the result on array.

1. **Update score and comments of database (Report.php)**

$queryUpdate = "UPDATE `score` SET `Comment`= '".$$comment."',`Score\_Criteria`= '".$$criteria."' WHERE `AssessmentNo` = '".$AssessmentNo."' AND `CriteriaNo`=$j";

$comment = 'comment'.$j;

$$comment = filter\_input(INPUT\_POST, $comment);

$criteria = 'c'.$j;

$$criteria = filter\_input(INPUT\_POST, $criteria);

This query is to update comment and score of score table. When user inputs each score and comments on view\_report page and clicks the submit button, report page is shown in order to confirm them what they inputted. This query also is repeated five times to fulfill all criteria and comments.

Therefore, it is essential to connect criteria and comment. Also assessment number is a key to differentiate the other assessments. Two variables, $$comment and $$ criteria, are used to convert the result of filter\_input function into incorporates the query.

1. **Show score and comments of database**

$queryScore = "SELECT `CriteriaNo`, `Comment`,`Score\_Criteria`FROM `score` WHERE `AssessmentNo` = ?";

On the view\_report page, user can input the score of each criteria and comments. If user inputs wrong data or database shows the wrong value which user intend to input, they can easily check their input on the report page. This query’s purpose is to show the contents of database. If some bug of query or anything that might affect the database contents, user can check the data itself.

1. **Calculate score received for each assessment**

$calc\_score = "UPDATE `assessment` SET `Score`= (SELECT SUM(Score\_Criteria) as OverallScore FROM score WHERE AssessmentNo = ?) WHERE AssessmentNo = ?";

User needs to input all score of criteria and comments. In order to store the sum of five criteria score, this query is to calculate overall score and update it. To specify a row of assessment table, assessment number is used.

1. **Find group according to assessmentno**

$find\_group = "SELECT `GroupNo` FROM `assessment` WHERE `AssessmentNo` = ?";

This query is to show group number. In order to specify the group which needs to calculate average score on the following query, it searches the group number from assessment number.

1. **Calculate average score received by each group**

$calc\_average = "UPDATE `group` SET `AverageScore`=(SELECT ROUND(((SUM(`Score`))/3)\*2) AS AverageScore FROM assessment WHERE Group\_to\_Assess = ?)

In this system, three peer groups assess one group. Each group’s assessment score is stored on assessment table. Therefore, average score of one group is calculated by sum of same group\_to\_assess number divide three times two. The result is stored average score column of assessment table. When a user submits one assessment, the result affects not only score table but also group table and assessment table. Thus report.php must have queries that can update related data.

1. **Update submission time**

$queryTime = "UPDATE `report` SET `Submission\_Timestamp`= NOW(),`Submission\_Updated`= NOW() WHERE `GroupNo`=?";

When user submitted their assessment, they need to store when the decision was decided. This query is to update the time of submission is executed. Database has two columns of time related part. One is time stamp and the other is submission update time.

User can revise their assessment every time they need, therefore database should store not only time stamp but update time. In order to prevent updating other team’s submission time, this query only update indicated group number’s data.

1. **Show threaded discussion board (Discussion.php)**

'SELECT', '`name`, `text`, `time`,`AssessmentID`', 'FROM mini\_board', 'ORDER BY `time` DESC',

When user clicks the discussion link of the header parts, this query will be executed. All information of discussion board is stored mini\_board table and this query is to show contributor’s name and contents of writing and submission time. All messages line up descending order of time, therefore, newest submission will be shown top of the part.

1. **Insert comment into threaded discussion board (Discussion\_title.php)**

'INSERT', 'INTO mini\_board(`name`,`text`,`assessmentID`,`time`)', 'VALUES(?, ?, ?, ?)',

When user would like to discuss specific assessment, they can write down comments about it. The discussion board is enable to set name and text. When User clicks the submit button, both of them will store the database. Also database should distinguish from other assessment discussion, assessmentID will store same table. Discussion board must be shown the contents in chronological orders so submission time is also stored same table.

'SELECT', '`id`,`name`,`text`, `time`,`AssessmentID`', 'FROM mini\_board',

'WHERE AssessmentID =', $id,

When user submits their discussion, this query shows the past submission contents and contributor. In this page, user can’t read other assessmentID discussion.

# Acknowledgement

1. [www.w3schools.com](http://www.w3schools.com) – Great websites for learning lots of basic HTML, CSS, JavaScript and PHP.
2. [www.codecademy.com](http://www.codecademy.com) – Another great website for learning basic websites design.
3. PHPMailer – An open source PHP function to send email. This is used in retrieving the password as an email need to be sent to the user of the generated random password. The code is available at <https://github.com/Synchro/PHPMailer>.
4. Twitter Bootstrap – An open source front-end framework. This is used as the template for the website. Twitter Bootstrap is available at <https://github.com/twbs/bootstrap> and <http://getbootstrap.com/>.