VIRTUAL LEARNING

GC06 Database and Information Management System

Team 24

Yusuke Kawano

Mohamad Fadhli Ismail

Izzatul Syazana Samsudin

Contents

1 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

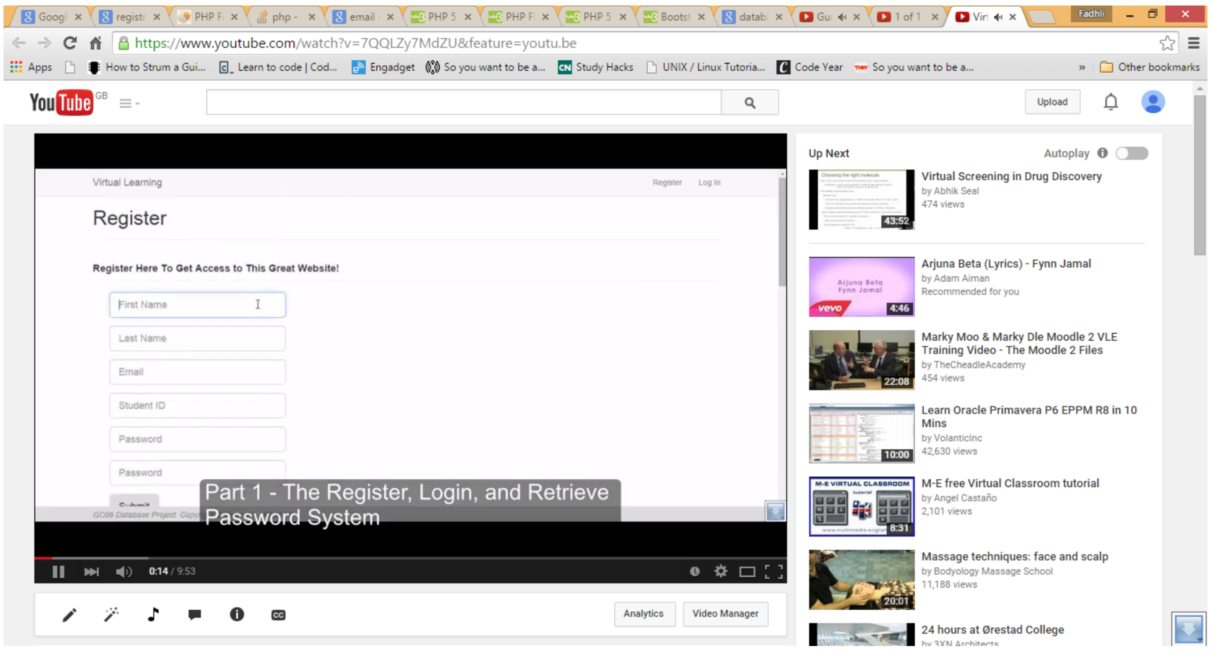
https://www.youtube.com/watch?v=7QQLZy7MdZU&feature=youtu.be
Virtual Learning
Register
Log In
Register
Register Here To Get Access to This Great Website!
First Name
Last Name
Email
Student ID
Password
Password
Part 1 - The Register, Login, and Retrieve Password System
0:14 / 9:53
Analytics
Video Manager
Up Next
Autoplay
Virtual Screening in Drug Discovery
by Abhik Seal
474 views
Arjuna Beta (Lyrics) - Fynn Jamal
by Adam Aiman
Recommended for you
Marky Moo & Marky Dle Moodle 2 VLE Training Video - The Moodle 2 Files
by TheCheadleAcademy
454 views
Learn Oracle Primavera P6 EPPM R8 in 10 Mins
by VolanticInc
42,630 views
M-E free Virtual Classroom tutorial
by Angel Castaño
2,101 views
Massage techniques: face and scalp
by Bodyology Massage School
11,188 views
24 hours at Ørestad College

Figure 1 – A screenshot of the video in YouTube

2 Entity Relationship Diagram (ERD)

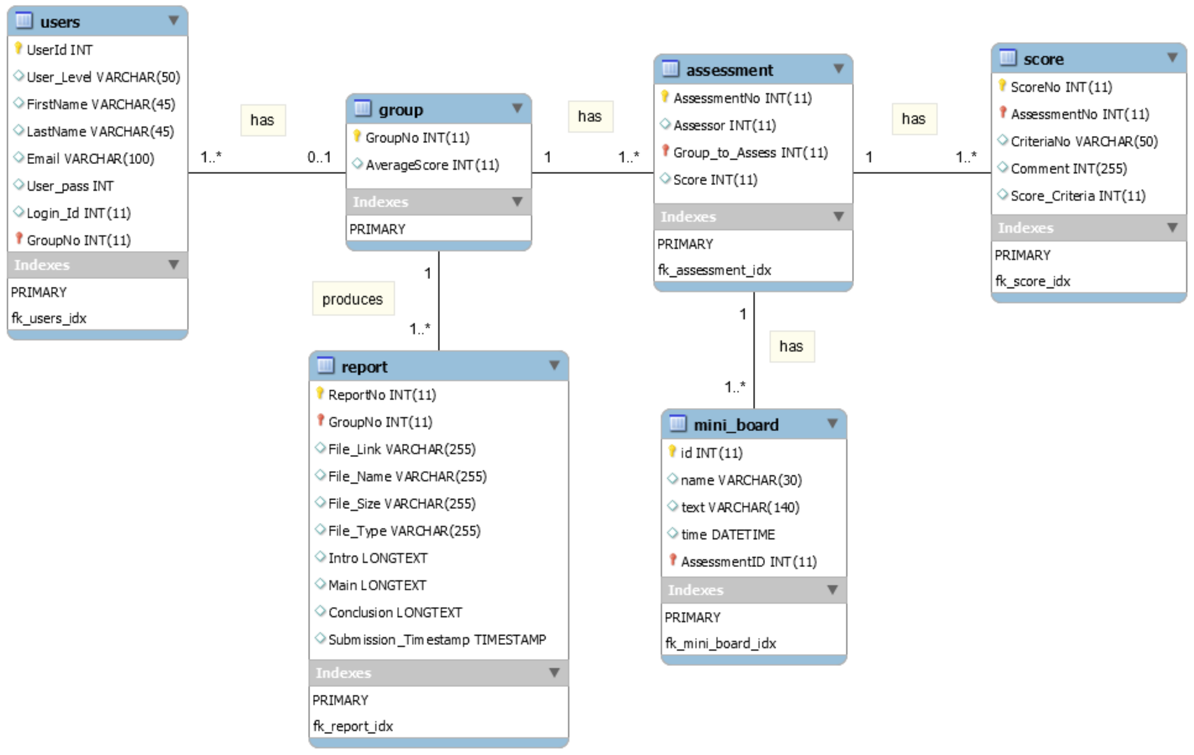

Figure 1: Entity Relationship Diagram for Virtual Learning database

Assumptions made when building this ERD:

1. There can be multiple students in the group. In this project, it is assumed that each group consists of three students. Hence, the relationship between \`group\` and \`users\` is one-to-many.
2. Each group will submit only one XML type report. Therefore, \`group\` and \`report\` have a one-to-one relationship.
3. Users consist of student and admin, which is differentiated by the “User_Level” column in \`users\` table.
4. Each group has to assess multiple other groups. In this case, it is assumed that each group will assess three other groups. To preserve anonymous assessment, “AssessmentNo” is assigned to the group, thus the group will not know which group they will assess. Therefore, the relationship between \`group\` and \`assessment\` is one-to-many.
5. For each assessment, the group has to rate the report based on a few criteria assigned to them. In this case, five criteria are assigned. The group has to give rating for each criteria and provides comment to justify the rating. Therefore, the \`assessment\` and \`score\` has a one-to-many relationship.
6. For each assessment, there is a simple forum setup for members of the group to discuss about each assessment. Every time a student post a comment, a new row is inserted in the \`mini_board\` table where “AssessmentID” will store the “AssesmentNo”. Therefore, it has a one-to-many relationships.

3 Listing of Database Schema

The virtual_learning database consists of six tables:

1. assessment
2. group
3. mini_board
4. score
5. report
6. users

Below is a list of database schema:

Field	Type	Null	Key	Default	Extra
AssessmentNo	int(11)	NO	PRI	*NULL*	auto_increment
Assessor	int(11)	NO		*NULL*	
Group_to_Assess	int(11)	NO	MUL	*NULL*	
Score	int(11)	NO		*NULL*	

Table 1: `assessment` table

In the `assessment` table, "AssessmentNo" is the primary key. Since each group is required to assess three other groups, therefore "Group_to_Assess" is a MUL key, where multiple occurrences can be found in this column. For example, group 1 may occur three times in the "Group_to_Assess" column but it has three different "AssessmentNo". "Group_to_Assess" is also a foreign key which referenced to "GroupNo" in the `group` table. On deletion or update of specific "GroupNo" in the `group` table, it will also delete or update the row which contains the specific "GroupNo" in "Group_to_Assess" column.

Field	Type	Null	Key	Default	Extra
GroupNo	int(11)	NO	PRI	NULL	auto_increment
AverageScore	int(11)	NO		NULL	

Table 2: `group` table

The "GroupNo" is the primary key. There is no foreign key in this table. The average score is the average of the total score received for each assessment.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
name	varchar(30)	NO		*NULL*	
text	varchar(140)	NO		*NULL*	
time	datetime	NO		*NULL*	
AssessmentID	int(11)	NO	MUL	*NULL*	

Table 3: `mini_board` table

"id" column is the primary key whereas the "AssessmentID" is a foreign key, which is referencing to "AssessmentNo" in `assessment` table. If a specific "AssessmentNo" is deleted or updated, the row which has that specific "AssessmentNo" in the "AssessmentID" column will be deleted or updated accordingly. Since every time a user posted a comment in the forum, a new row is inserted into the table, which results in duplicate of "AssessmentNo", hence the MUL key for "AssessmentID" column.

Field	Type	Null	Key	Default	Extra
ScoreNo	int(11)	NO	PRI	*NULL*	auto_increment
AssessmentNo	int(11)	NO	MUL	*NULL*	
CriteriaNo	varchar(50)	NO		*NULL*	
Comment	varchar(255)	NO		*NULL*	
Score_Criteria	int(11)	NO		*NULL*	

Table 4: `score` table

The “ScoreNo” column is the primary key whereas the “AssessmentNo” is a MUL key, which means that it has duplicate “AssessmentNo” in that column. This is because for each assessment, there will be multiple criteria for user to assess on each report. For each criteria, there is a score given as well as a comment to justify the rating. “AssessmentNo” is also a foreign key in which it will be deleted or updated if a specific “AssessmentNo” is deleted or updated in the `assessment` table.

Field	Type	Null	Key	Default	Extra
ReportNo	int(11)	NO	PRI	*NULL*	auto_increment
GroupNo	int(11)	NO	UNI	*NULL*	
File_Link	varchar(255)	NO		*NULL*	
File_Name	varchar(255)	NO		*NULL*	
File_Size	varchar(255)	NO		*NULL*	
File_Type	varchar(255)	NO		*NULL*	
Intro	varchar(1000)	NO		*NULL*	
Main	varchar(1000)	NO		*NULL*	
Conclusion	varchar(1000)	NO		*NULL*	
Submission_Timestamp	timestamp	NO		CURRENT_ TIMESTAMP	

Table 5: `report` table

The “ReportNo” column is the primary key whereas the “GroupNo” column is a unique key as well as a foreign key. This acts as a unique id when user re-upload or re-submit their report where instead of inserting a new row to the table, it will update the row which consists of the unique “GroupNo”. This prevents duplicate entry of submission from the same group.

Since the “GroupNo” is a foreign key too, if a certain “GroupNo” is deleted or updated from the `group` table, the row which consists that particular “GroupNo” in the `report` table will be deleted or updated as well.

Field	Type	Null	Key	Default	Extra
UserId	int(11)	NO	PRI	*NULL*	auto_increment
User_Level	varchar(50)	NO		*NULL*	
FirstName	varchar(45)	NO		*NULL*	
LastName	varchar(45)	NO		*NULL*	
Email	varchar(100)	NO		*NULL*	
User_pass	varchar(255)	NO		*NULL*	
Login_Id	int(11)	NO		*NULL*	
GroupNo	int(11)	YES	MUL	*NULL*	

Table 6: `users` table

In the `users` table, the primary key is “UserId” whereas the foreign key is “GroupNo”. The “GroupNo” column has MUL key because a group consists of a few of students. On update of “GroupNo” in the master table, which is `group`, the “GroupNo” in the `users` table will also be updated. However, on delete, this will set “GroupNo” in the `users` table as NULL. Hence, this column is set to allow null entry. If “GroupNo” has NULL, this means that either the user is an admin or a student with no group allocated.

4 Third Normal Form

Below is the analysis showing the database schema in its 3rd normal form.

1. **‘assessment` Table**

Example of 1NF:

AssessmentNo	Assessor	Group_To_Assess	Score

Example of 2NF:

AssessmentNo	Assessor	Group_To_Assess	Score

Example of 3NF:

AssessmentNo	Assessor	Group_To_Assess	Score

In the `assessment` table, the “AssessmentNo” is the primary key and unique. Although “Assessor” must be allocated same number if the students' group were same, there is no relation between assessor, group_To_assess and score. Eventually, 1nf, 2nf, 3nf have the same entity.

2. **`mini_board` table**

Example of 1NF:

id	name	text	time	AssessmentID

Example of 2NF:

id	name	text	time

id	AssessmentID

Example of 3NF:

id	name	text	time

id	AssessmentID

In the mini_board table, id is primary key and unique. Id determines name, text and time. AssessmentID is independent from id, therefore AssessmentID should be separated by name,text, time. Therfore, there are two entities, id and assessment. However, there is any dependencies between non-key attributes both two entities. Thus, 3NF is same as 2NF.

3. **\`score\` Table**

Example of 1NF:

ScoreNo	AssessmentNO	CriteriaNo	Comment	Score_Criteria

Example of 2NF:

ScoreNo	AssessmentNO	CriteriaNo	Comment	Score_Criteria

Example of 3NF:

ScoreNo	AssessmentNO

AssessmentNO	CriteriaNo	Comment	Score_Criteria

In the score table, ScoreNo is primary key and unique. Non-key attribute is depending on the scoreNo, therefore 2NF is same as 1NF. Each five criteria must have unique comment and score_criteria therefore those attributes should be separated from other attributes.

AssessmentNo determines which group assesses the other group, thus AsssessmentNo is a foreign key of the score entity.

4. **`report` Table**

Example of 1NF:

<u>Report No</u>	Group No	File_ Link	File_N ame	File_Size	File_ Type	Intro	Main	Conclusion	Submission_ Timestamp

Example of 2NF:

<u>Report No</u>	Group No	File_ Link	File_N ame	File_Size	File_ Type	Intro	Main	Conclusion	Submission_ Timestamp

Example of 3NF:

<u>ReportNo</u>	GroupNo

<u>ReportNo</u>	Intro	Main	Conclusion

<u>ReportNo</u>	File_Link	File_Name	File_Size	File_Type	Submission_Timestamp

In the report table, reportNo is primary key and unique. Non-key attribute is depending on the ReportNo, therefore 2NF is same as 1NF. GroupNo is functionally dependent on ReportNo. File_Link, File_Name, File_Size, File_Type and Submission_Timetamp are functionally dependent on ReportNo. Intro, Main,Conclusion means Xml file contents and these are also functionally dependent on ReportNo.

5. **`users` Table**

Example of 1NF:

<u>UserID</u>	User_Level	FirstName	LastName	Email	User_pass	Login_Id	GroupNo

Example of 2NF:

<u>UserID</u>	User_Level	FirstName	LastName	Email	User_pass	Login_Id	GroupNo

Example of 3NF:

<u>UserID</u>	User_Level

<u>UserID</u>	FirstName	LastName	Email	GroupNo

<u>UserID</u>	User_pass	Login_Id

In the user table, UserID is primary key and unique. Non-key attribute is depending on the UserID, therefore 2NF is same as 1NF. User_Level has two kinds of value which is student or administrator. It is determined by independently to other attributes. Thefore, User_Level is functionally dependent on UserID. FirstName,LastName,Email and GroupNo are functionally dependent on UserID. Also, User_Pass and Login_ID are functionally dependent on UserID

6. **`group` Table**

Example of 1NF:

<u>GroupNo</u>	ReportNo	AverageScore

Example of 3NF:

<u>GroupNo</u>	ReportNo	AverageScore

Example of 3NF:

<u>GroupNo</u>	ReportNo	AverageScore

In the group table, GroupNo is primary key and unique. Non-key attribute is depending on the GroupNo, therefore 2NF is same as 1NF. There are no functionally dependent between primary key and non-key attribute. Thus, 1nf, 2nf, 3nf are same entity.

5 List of Queries

5.1 Part 1

7. Checking whether the student data is already in the database

```
SELECT Login_Id FROM users WHERE Login_Id = ?
```

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 Login_Id 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.

8. Inserting the user's information provided into the database

```
INSERT INTO users (FirstName, LastName, Email, User_pass, Login_Id, 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 Login_Id 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.

9. Retrieving information from the database for login purposes

```
SELECT GroupNo, User_pass, User_Level, FirstName, LastName FROM users WHERE
Login_Id = ?
```

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 Login_Id 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.

10. Retrieving user password

```
SELECT User_pass, FirstName, LastName, Email FROM users WHERE Login_Id = ?
```

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.

11. Updating the users password

```
UPDATE users SET User_pass= ? WHERE Login_Id = ?
```

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.

12. Retrieving hashed password and comparing with data provided by user

```
SELECT User_pass, FirstName, LastName, Email FROM users WHERE Login_Id = ?
```

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.

13. Updating the password with random generated password

```
UPDATE users SET User_pass= ? WHERE Login_Id = ?
```

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.

5.2 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.

2. **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 each receive the same rank, 8 for example. If a subsequent group has a different mark, this group's rank will be 10 and not 9.

3. **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 assesor. The results are displayed in the home.php page.

4. **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 (Intro, Main, Conclusion) 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. Otherwise, it will insert a new row to the `report` table.

5. 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 on `AssessmentNo`. 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 of each assessment, where the result is used to find the average score for all three assessments. There are five criteria for each assessment and for each criteria, user can give a max score of 10. Since there are three assessments for each group, the "OverallScore" would be the sum of (Score_Criteria/(3*50)*100.

6. 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.

7. Number of Created Groups

```
SELECT COUNT(DISTINCT GroupNo) AS `countgroup` FROM users
```

The `users` table consists of list of registered users on the website and they are distinguished by the user level: student or 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. Any “GroupNo” that has NULL value will be omitted from the count. The null in here means that the student has not been allocated a group or the user is an admin.

8. 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 7.

9. Show List of Unsorted Students

```
SELECT `UserId`, `FirstName`, `LastName`, `GroupNo` FROM `users` WHERE `GroupNo`
IS NULL AND `User_Level` = 'student' ORDER BY `FirstName`
```

For users labelled as “student” with `GroupNo` = NULL 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.

10. Update Student’s Group

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

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”. There are two sets of query which is executed depending on the “GroupNo” received from AJAX. If the “GroupNo” == ‘0’, the first query is executed where “GroupNo” is set to NULL. Otherwise, second query is executed.

11. 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.

12. Update Student’s Details

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

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

5.3 Part 3

13. Assessing Peer’s Report

```
SELECT  `Group_to_Assess`,`AssessmentNo` FROM `assessment` WHERE `GroupNo` = ?
```

This query is to show report by selecting the value of Group_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 Group_to_Assess number is used by implying the detail of report number.

1. **Grading criteria, writing comment and Displaying Report**

```
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. The following query 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.

```
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.

```
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.

2. **Update score and comments of database**

```
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.

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

```
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.

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

```
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.

5. **Find group according to assessmentno**

```
SELECT `Group_to_Assess` 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.

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

```
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.

7. **Update submission time**

```
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.

8. **Show threaded discussion board**

```
'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.

9. **Insert comment into threaded discussion board**

```
'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.

6 Acknowledgement

1. www.w3schools.com – Great websites for learning lots of basic HTML, CSS, JavaScript and PHP.
2. 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/.