Login Form:

Checking if UserID is valid or not.

SELECT Username, Type  
FROM User  
WHERE Username = $user  
AND Password = $pass

Registration Form:

Checking if UserID has already been picked by another user.

SELECT Count (Username)  
FROM User  
WHERE `Username` = $user

AND `Username` IS NOT NULL;

If not, then these statements insert data into the database.

INSERT INTO User

VALUES ($user, $pass, $type);

INSERT INTO regularuser (Username)

VALUES ($user);

INSERT INTO student (Username)  
VALUES ($user);

Student and Faculty Profile forms:

Student Personal Information

Retrieving existing data from database:

SELECT \* FROM RegularUser WHERE `Username` = $user AND `Username` IS NOT NULL

SELECT \* FROM Student WHERE `Username` = $user AND `Username` IS NOT NULL

//retrieving courses for drop-down menu for tutoring.

SELECT S.Title, R.Grade

FROM Section S, Registers R

WHERE S.CRN = R.CRN AND R.Username = $user AND `Username`

NOT IN (SELECT Username

FROM TutorFor

WHERE Username = $user AND Title = S.Title);

Updating new data into database:

UPDDATE RegularUser SET

Email = $email,

Name = $name,

DOB = $dob,

Address = $address,

PermanentAddress = $permanentAddress,

Gender = $gender,

ContactNumber = $contactNumber

WHERE Username = $user;

UPDATE Student SET

Major = $major,

degree = $degree,

WHERE Username = $user

SELECT Username FROM TutorFor

WHERE Username = $user AND Title = $title;

//if the result of the above set is not null, following query is executed:

INSERT INTO AppliedToTutor

VALUES($user, $title);

INSERT INTO Education\_History

VALUES($user, $institution, $yearGrad, $degree, $major, $gpa);

Faculty Personal Information

Retreiving existing data from database

SELECT \* FROM RegularUser

WHERE Username = $user AND Username IS NOT NULL;

SELECT \* FROM Faculty

WHERE Username = $user AND Username IS NOT NULL;

SELECT \* FROM Department

WHERE Dept\_Id = $deptID AND DeptId IS NOT NULL;

SELECT \* FROM FacultyResearchInterst

WHERE Username = $user;

SELECT T.Title, T.Letter

FROM Section T WHERE T.Instructor = $user;

Updating existing information into database

INSERT INTO FacultyResearchInterst VALUES($user, $research)

UPDDATE RegularUser SET

Email = $email,

Name = $name,

DOB = $dob,

Address = $address,

PermanentAddress = $permanentAddress,

Gender = $gender,

ContactNumber = $contactNumber

WHERE Username = $user;

Average Grade by each professor: student report

Dropping views if they exist already

DROP VIEW IF EXISTS cs4400\_Group32.grade\_view

DROP VIEW IF EXISTS count\_view

Creating the two views with organized data for manipulation

CREATE VIEW grade\_view AS

SELECT F.username AS Instructor,

Code AS Course\_Code,

C.Title AS Course\_Name,

R.Grade, Count(\*) AS Total\_points

FROM Faculty F,

Section S,

Code C,

Registers R

where (F.Username = S.Instructor

AND C.Title = S.Title

AND R.CRN = S.CRN)

GROUP BY F.Username, R.Grade

SELECT \* FROM grade\_view

//Counting the number of A’s , B’s, C’s, D’s and F’s for a specific professor

CREATE VIEW count\_view AS

SELECT Instructor,

SUM(Total\_Points) AS Student\_Grade

FROM grade\_view

GROUP BY Instructor

SELECT \* FROM count\_view

After this, the java program handles the further processing of grades and produces average grades on the screen.

Faculty Report:

DROP VIEW IF EXISTS faculty\_grade\_view

DROP VIEW IF EXISTS faculty\_count\_view

CREATE VIEW faculty\_grade\_view

AS SELECT C.Code , C.Title , COUNT( \* ) AS Number\_of\_meetings, grade  
FROM section S, code C, logsvisitdate L, registers R  
WHERE (

S.Instructor = $instructor   
AND C.Title = S.Title   
AND L.CRN = S.CRN  
AND R.CRN = S.CRN

)  
GROUP BY (C.Code);

//Counts the number of A’s , B’s, C’s, D’s and F’s for a specific course

CREATE VIEW faculty\_count\_view

AS SELECT code, grade , COUNT(\*) AS faculty\_Grade\_count

FROM faculty\_grade\_view

GROUP BY code, grade;

After this the java/php query can handle the further processing of grades and produce average grade on the screen

Admin Report:

Admin Report generates a list of all courses and the average grade in each.

CREATE VIEW admin\_grade\_view

AS SELECT C.Code, C.Title , R.grade , count(\*) AS Total\_number  
FROM Code C, Section S, Registers R  
WHERE (

C.Title = S.Title   
AND S.CRN = R.CRN

)  
GROUP BY C.Code;

CREATE VIEW admin\_count\_view

AS SELECT code, grade , SUM(grade) AS Admin\_Grade\_count

FROM admin\_grade\_view

GROUP BY code, grade;

Furthur processing by the Java program can be used to find the average of different sections of the same course.

Tutor Logbook:

the following query checks if the user signed in is a tutor or not.

SELECT Count(\*)

FROM Tutor T

WHERE T.Username = $user;

the following query retrieves the name of the tutor signed in.

SELECT R.Name

FROM RegularUser R

WHERE R.Username = $user

the following query retrieves the classes for which the tutor can log times, which will fill up the drop-down menu.

SELECT C.Code

FROM TutorFor T, Code C

WHERE (T.Username = $user

AND C.Title = T.Title)

The following query retrieves the CRN of the class for which the tutor is registered

SELECT S.CRN

FROM TutorFor T, Code C, Section S

WHERE C.Title = S.Title AND S.Title = T.Title

The following query checks if the name and studentId written are valid values.

SELECT Username

FROM RegularUser

WHERE Username = $student AND Name = $NAME

GROUP BY Username

The following query inserts the data into the database

INSERT INTO LogsVisitDate

VALUES ($user, crn, $student, $time);

Tutor search:

SELECT C.Code, T.Title, T.Username, R.Email

FROM Code C, TutorFor T, RegularUser R

WHERE (Code = $code OR C.Title LIKE “word”

AND T.Title = C.Title AND T.Username = R.Username)

GROUP BY C.Code

Tutor Assignment:

The following query retrieves data for tutors who have applied

SELECT DISTINCT U.Name, T.Username

FROM AppliedToTutor t, Section S, RegularUser U

WHERE U.Username = T.Username AND S.Instructor = $user

AND T.Title IN (

SELECT Title FROM Section WHERE Instructor = $user)

Retrieves the course for which a tutor has applied

SELECT Title FROM AppliedToTutor WHERE Username = $user;

Insert into database the information

INSERT INTO TutorFor VALUES($user, $title);

Delete from appliedToTutor table the same values

DELETE FROM AppliedToTutor WHERE Username = $user;

Classes Registration form:

Obtain the course which the users has not taken

String query = "SELECT CRN, S.Title, Name, Time, Day, Location, Letter, Code FROM Section S, Code C, RegularUser U WHERE CRN NOT IN " +

Insert that course in database

Insert into Registers (Username, CRN, Grade\_Mode) VALUES ($user, $crn, $grade\_mode);

SELECT CRN, S.Title, Name, Time, Day, Location, Letter, Code FROM Section S, Code C, RegularUser U WHERE CRN NOT IN (SELECT R.CRN FROM Registers R WHERE Username = $user) AND C.Title = S.Title AND U.Username = S.Instructor