Variables have been assigned a '$' symbol when processed by the PHP code. Variables not denoted with this symbol are field names within the MySQL database tables.

ProcessLoginPage queries

$UserLogin decides the login name of the User within their table. 'User' is substituted with the User category identifier thereafter.
$UserPassword decides the login password of the User within their table. 'User' is substituted with the User category identifier thereafter and only used during the login process. It is not used elsewhere.

$UserType decides type of User. This was selected manually by the User in LoginPage.

Queries for searching correct User table for required fields

If $UserType is Student:

SELECT StudentFirstName, StudentSurname, StudentForm, StudentEmail, StudentPhoneNumber, StudentForm, StudentID FROM StudentTable WHERE StudentLogin = '$StudentLogin' AND StudentPassword = '$StudentPassword';

If $UserType is Parent:

SELECT ParentFirstName, ParentSurname, ParentEmail, ParentPhoneNumber, ParentID where ParentLogin = '$ParentLogin' AND ParentPassword = '$ParentPassword';

If $UserType is Mentor:

SELECT MentorFirstName, MentorSurname, MentorEmail, MentorPhoneNumber, MentorID WHERE MentorLogin = '$MentorLogin' AND MentorPassword = '$MentorPassword';

ViewMeetingPage queries

$UserType decides the type of User.
$StudentID is loaded into memory before the queries if $UserType is a Student.
$ParentID is loaded into memory before the queries if $UserType is a Parent.
$MentorID is loaded into memory before the queries if $UserType is a Mentor.

The table fields StudentFirstName, StudentSurname, StudentForm, StudentID, ParentFirstname, ParentSurname, ParentID, MentorFirstName, MentorSurname, MentorID, MeetingDate and MeetingID are also assigned to PHP variables as they are implemented into an array from the sql queries in order to be output by the program code.

Queries for other relevant Users

If $UserType is Student:

SELECT ParentID FROM StudentTable WHERE StudentID = '$StudentID';
SELECT ParentFirstName, ParentSurname FROM ParentTable WHERE ParentID = '$ParentID';
and
SELECT MentorID FROM StudentTable WHERE StudentID = '$StudentID';
SELECT MentorFirstName, MentorSurname FROM MentorTable WHERE MentorID = '$MentorID';

If $UserType is Parent:

SELECT StudentFirstName, StudentSurname, StudentForm, StudentID FROM StudentTable WHERE ParentID = '$ParentID';
SELECT MentorID FROM StudentTable WHERE StudentID = '$StudentID';
SELECT MentorFirstName, MentorSurname, MentorID FROM MentorTable WHERE StudentID = '$StudentID';

If $UserType is Mentor:

SELECT StudentFirstName, StudentSurname, StudentForm, StudentID FROM StudentTable WHERE MentorID = '$MentorID';
SELECT ParentFirstName, ParentSurname, ParentID FROM ParentTable WHERE StudentID = '$StudentID';

Queries for existing meetings for User

Students:
SELECT MentorID FROM StudentTable WHERE StudentID = '$StudentID';

Parents:
SELECT StudentID, MentorID FROM StudentTable WHERE ParentID = '$ParentID';

Mentors:
SELECT StudentID FROM StudentTable WHERE MentorID = '$MentorID';

Query for obtaining individual meeting records:

SELECT MeetingDate, MeetingID FROM MeetingTable WHERE StudentID = '$StudentID' AND MentorID = '$MentorID';

AddMeetingPage Queries

Only Mentors may access this page. They may choose any of their relevant Students to assign the meeting to and set a date. Because a Mentor may set a meeting earlier or later than the actual meeting date, there is no validation for when the date is set to, except for qithin the academic year. MentorID and MeetingID are automatically assigned to the new meeting.

Query for obtaining Student information:

SELECT StudentFirstName, StudentSurname, StudentID FROM StudentTable WHERE MentorID = '$MentorID';

Query for finding MeetingID number; the program uses iteration to find an unoccupied value for MeetingID in the table (in the case automatic incrementing of ID numbers may fail):

SELECT MeetingID FROM MeetingTable;

Query for inserting a new meeting record:

INSERT INTO MeetingTable (MeetingID, StudentID, MentorID, MeetingDate) VALUES ('$MeetingID', '$StudentID', '$MentorID', '$MeetingDate');

EditMeetingPage queries

Only a Mentor should be able to access this page with a given meeting ID. To check that the Mentor ID and meeting ID are both correct, the current Mentor ID is checked against the record's Mentor ID and the current meeting ID is checked against that of the record. The system will then check for existing meeting information and display it to the Mentor.

Query for validating that MentorID and MeetingID are correct:

SELECT FROM MeetingTable WHERE MeetingID = '$MeetingID' AND MentorID = '$MentorID';

Query for displaying meeting information:

SELECT StudentID, MeetingDate FROM MeetingTable WHERE MeetingID = '$MeetingID';

DeleteMeetingPage queries

This page is accessible only by Mentors. It checks the current data continued from the previous page against the record data in the database. If MentorID and MeetingID are correct, The system will output the details of the meeting and enable the User to delete the meeting if they wish. StudentTable's details will require several SQL statements. One for obtaining the StudentID held in MeetingTable and another for retrieving StudentFirstName, StudentSurname and StudentForm using StudentID's value, '$StudentID'.

Query for validating that correct meeting is selected and correct Mentor requests deletion of record:

SELECT FROM MeetingTable WHERE MeetingID = '$MeetingID' AND MentorID = '$MentorID';

Query for meeting information:

SELECT MeetingDate, StudentID FROM MeetingTable WHERE MeetingID = '$MeetingID';

SELECT StudentFirstName, StudentSurname, StudentForm FROM StudentTable WHERE StudentID = '$StudentID';

Query for deleting record from MeetingTable:

DELETE FROM MeetingTable WHERE MeetingID = '$MeetingID';

Queries for deleting other related elements:

DELETE FROM TargetTable WHERE MeetingID = '$MeetingID';
DELETE FROM FileTable WHERE MeetingID = '$MeetingID';

ViewTargetPage queries

$UserType is decided by the type of User.
$MeetingID is loaded into memory so that only targets belonging to the meeting will be displayed. This is used in a check to confirm the correct User is viewing data.
$StudentID is loaded into memory so that only targets belonging to relevant Users may be displayed. This is used in a check to confirm the correct User is viewing data. StudentID value may feature only in content for that particular Student.
Table fields StudentTarget, MentorTarget, StudentComment, ParentComment, MentorComment, DateDue, TargetMetYet, TargetID are also used as variables by the program code.

Query for validating that meeting and Student's ID are correct:

SELECT FROM MeetingTable WHERE MeetingID = '$MeetingID' AND StudentID = '$StudentID';

Query for viewing information on targets:

SELECT StudentTarget, MentorTarget, DateDue, StudentComment, ParentComment, MentorComment, TargetMetYet FROM TargetsTable WHERE MeetingID = '$MeetingID';

AddTargetPage queries

$UserType is decided by the type of User.
$MeetingID is set based on the meeting Mentor is currently adding a target to.
$MentorID is loaded into memory after the $UserType validation.
$TargetID is searched for numerous times until an unused value is found. This is then the proposed value for the new record's TargetID.
$TargetMetYet is set to '0' because it is assumed that the target is not completed before it is set through the computer system. A '1' may replace the '0' should the target be completed after setting the new target.

Accessible only to Mentors, $UserType is set and checked to verify whether the current User may continue. The Mentor may then enter details for StudentTarget, MentorTarget, DateDue and MentorComment. TargetID, MeetingID and TargetMetYet are automatically asssigned a value. StudentComment and ParentComment are left blank for their respective Users to add to.

Query for validating that Mentor and meeting are correct:

SELECT FROM MeetingTable WHERE MeetingID = '$MeetingID' AND MentorID = '$MentorID';

Query for finding TargetID, used to find lowest number for non-existent record through iteration:

SELECT TargetID FROM TargetTable;

Query for inserting record into TargetTable:

INSERT INTO TargetTable (StudentTarget, MentorTarget, DateDue, MentorComment, TargetMetYet, MeetingID, TargetID) VALUES ('$StudentTarget', '$MentorTarget', '$DateDue', '$MentorComment', '$TargetMetYet', '$MeetingID', '$TargetID');

EditTargetPage queries

$UserType is used to verify if User is a Mentor.
$MeetingID is used to verify if Meeting is correct.
$MentorID is used to verify if Mentor is correct.
$TargetID is used to retrieve details for correct target and replace any changed fields in the same target.

Query for validating if Mentor and meeting are correct:

SELECT FROM MeetingTable WHERE MeetingID = '$MeetingID' AND MentorID = '$MentorID';
SELECT FROM TargetTable WHERE TargetID = '$TargetID';

Query for retrieving existing information from table TargetTable:

SELECT StudentTarget, MentorTarget, StudentComment, ParentComment, MentorComment, DateDue, TargetMetYet, FROM TargetTable WHERE TargetID = '$TargetID';

Query for replacing fields in record:

INSERT INTO TargetTable (StudentTarget, MentorTarget, StudentComment, ParentComment, MentorComment, DateDue, TargetMetYet) VALUES ('$StudentTarget', '$MentorTarget', '$StudentComment', '$ParentComment', '$MentorComment', '$DateDue', '$TargetMetYet');

DeleteTargetPage queries

$MeetingID is the meeting ID.
$UserType decides the type of User.
$MeetingID is used to verify that the target has ben correctly selected.
$MentorID is used to check if the Mentor is correct.
$FileID is set for specific file.

Query for validating if Mentor and meeting are correct:

SELECT FROM MeetingTable WHERE MeetingID = '$MeetingID' AND MentorID = '$MentorID';
SELECT FROM FileTable WHERE FileID = '$FileID';

Query for retrieveing existing information:

SELECT FileName, FileExtension, FileSize, FileResourceLocation, UserType, UserID, FileID FROM

FIleTable WHERE FileID = '$FileID';

Query for deleting record:

DELETE FROM FileTable WHERE FileID = '$FileID';

ViewFilePage queries

$MeetingID is set for displaying only files relevant to particular meeting.
$UserType is set because different Users may have different priviledges.
$StudentID is set for checking that Meeting is relevant to particular Student.
$ParentID is set if $UserType is Parent.
$MentorID is set if $UserType is Mentor.

Query for verifying that relevant Meeting and Student are correct:

SELECT FROM MeetingTable WHERE MeetingID = '$MeetingID' AND StudentID = '$StudentID';

Query for obtaining and displaying current file data:

SELECT FileName, FileExtension, FileSize, FileResourceLocation, FileID, MeetingID FROM FileTable WHERE MeetingID = '$MeetingID';

DeleteFilePage queries

Query for displaying current file information:

SELECT FileName, FileExtension, FileSize, FileResourceLocation, FileID, MeetingID FROM FileTable WHERE MeetingID = '$MeetingID';

Query for deleting a file:

DELETE FROM FileTable WHERE FileID = '$FileID';

AddFilePage queries

$UserType is decided based on type of User.
$StudentID is set if $UserType is Student.
$ParentID is set if $UserType is Parent.
$MentorID is set if $UserType is Mentor.

Query for fetching all files for a user:

SELECT FileSize FROM FileTable WHERE UserType = '$UserType' AND UserID =

Query for inserting record:

INSERT INTO TargetTable FIELDS (StudentTarget, MentorTarget, DateDue, MentorComment, TargetMetYet, MeetingID, TargetID) VALUES ('$StudentTarget', '$MentorTarget', '$DateDue', '$MentorComment', '$TargetMetYet', '$MeetingID', '$TargetID');

ViewContactPage queries

$UserType is decided based on the type of user.
$UserID is set for the ID of the current user, regardless of type.

$StudentID is set if $UserType is Student.
$ParentID is set if $UserType is Parent.
$MentorID is set if $UserType is Mentor.
$RequesterUserID is set for each User ID found to request current User.
$RequesterUserType is set for each User found to request current User.
$RequestedUserID is the current UserID if loaded.
$RequestedUserType is the current UserType if loaded.

Query for retrieving StudentForm options:

```
SELECT DISTINCT StudentForm FROM StudentTable;
```

Queries if $UserType is Student:

```
SELECT ParentID FROM StudentTable WHERE StudentID = '$StudentID';
SELECT ParentFirstName, ParentSurname FROM ParentTable WHERE ParentID = '$ParentID';

SELECT MentorID FROM MentorTable WHERE MentorID = '$MentorID';
SELECT MentorFirstName, MentorSurname FROM MentorTable WHERE MentorID = '$MentorID';
```

Queries if $UserType is Parent:

```
SELECT StudentID FROM StudentTable WHERE ParentID = '$ParentID';
SELECT StudentFirstName, StudentSurname, StudentForm FROM StudentTable WHERE StudentID =
'$StudentID';

SELECT MentorID FROM StudentTable WHERE StudentID = '$StudentID';
SELECT MentorFirstName, MentorSurname FROM MentorTable WHERE MentorID = '$MentorID';
```

Queries if $UserType is Mentor:

```
SELECT StudentID FROM StudentTable WHERE MentorID = '$MentorID';
SELECT StudentFirstName, StudentSurname, StudentForm FROM StudentTable WHERE StudentID =
'$StudentID';

SELECT ParentID FROM StudentTable WHERE StudentID = '$StudentID';
SELECT ParentFirstName, ParentSurname FROM ParentTable WHERE ParentID = '$ParentID';
```

Deleting relation with Parent from StudentTable as a Student or Parent:

```
UPDATE StudentTable SET ParentID = NULL WHERE ParentID = '$ParentID';
```

Deleting relation with Mentor from StudentTable as a Student or Mentor:

```
UPDATE StudentTable SET MentorID = NULL WHERE MentorID = '$MentorID';
```

Query for checking if current User has been added:

```
SELECT RequesterUserID, RequesterUserType FROM RequestTable WHERE RequestedUserID =
'$UserID' AND RequestedUserType = '$UserType';
```

Query for displaying any Parent who has added current Student:

```
SELECT ParentFirstName, ParentSurname FROM ParentTable WHERE ParentID = '$RequesterUserID';
```

Query for displaying any Mentor who has added current Student:

```
SELECT MentorFirstName, MentorSurname FROM MentorTable WHERE MentorID = '$RequesterUserID';
```

Query for displaying any Student who has added current Parent/Mentor:

```
SELECT StudentFirstName, StudentSurname, StudentForm FROM StudentTable WHERE StudentID =
'$RequesterUserID';
```

For adding the current Parent's/Mentor's ID to the StudentTable entry for current Parent/Mentor User:

UPDATE StudentTable SET ParentID/MentorID = '$UserID' WHERE StudentID = '$RequesterUserID';

For adding the current Parent's/Mentor's ID to the StudentTable entry for current Student User:

UPDATE StudentTable SET ParentID/MentorID = '$RequesterUserID' WHERE StudentID = '$UserID';

For rejecting (and after adding the requester) the request:

DELETE FROM RequestTable WHERE RequestedUserID = '$UserID' AND RequestedUserType = '$UserType' AND RequesterUserID = '$RequesterUserID' AND RequesterUserType = '$RequesterUserType';

Query for adding a Parent to StudentTable entry if $UserType is Student or Parent:

UPDATE StudentTable SET ParentID = '$ParentID' WHERE StudentID = '$StudentID';

Query for adding a Mentor to StudentTable entry if $UserType is Student or Mentor:

UPDATE StudentTable SET MentorID = '$MentorID' WHERE StudentID = '$StudentID';

SearchContact queries

$UserType is set based on type of User.
$StudentID is set if $UserType is Student.
$ParentID is set if $UserType is Parent.
$MentorID is set if $UserType is Mentor.
$RequesterUserID is the current UserID.
$RequesterUserType is the current UserType.
$RequestedUserID is the UserID of the target User to add.
$RequestedUserType is the UserType of the target User to add.

Query for finding existing Parent using $StudentID:

SELECT ParentID FROM StudentTable WHERE StudentID = '$StudentID';
SELECT ParentFirstName, ParentSurname FROM ParentTable WHERE ParentID = '$ParentID';

Query for finding existing Mentor using $StudentID:

SELECT MentorID FROM StudentTable WHERE StudentID = '$StudentID';
SELECT MentorFirstName, MentorSurname FROM MentorTable WHERE MentorID = '$MentorID';

Query for finding existing Student using $ParentID if $UserType is Parent:

SELECT StudentID FROM StudentTable WHERE ParentID = '$ParentID';

Query for finding existing Student using $MentorID if $UserType is Mentor:

SELECT StudentID FROM StudentTable WHERE MentorID = '$MentorID';

Query for finding Student information from StudentTable if StudentID retrieved:

SELECT StudentFirstName, StudentSurname, StudentForm FROM StudentTable WHERE StudentID = '$StudentID';

Query for finding a Student using any or all of the fields from StudentFirstName, StudentSurname and StudentForm:

SELECT StudentFirstName, StudentSurname, StudentID StudentForm FROM StudentTable WHERE

StudentfirstName = '*$StudentFirstName*' OR StudentSurname = '*$StudentSurname*' OR StudentForm = '*$StudentForm*';

Query for finding a Parent using any or all of the fields from ParentFirstName and ParentSurname:

SELECT ParentFirstName, ParentSurname, ParentID FROM ParentTable WHERE ParentfirstName = '*$ParentFirstName*' OR ParentSurname = '*$ParentSurname*';

Query for finding a Mentor using any or all of the fields from MentorFirstName and MentorSurname:

SELECT MentorFirstName, MentorSurname, MentorID FROM MentorTable WHERE MentorfirstName = '*$MentorFirstName*' OR MentorSurname = '*$MentorSurname*';

Query for adding a user request to the database so that the other user can confirm the relation:

INSERT INTO RequestTable FIELDS (`RequesterUserID`, `RequesterUserType`, `RequestedUserID`, `RequestedUserType`) VALUES ('$UserID', '$UserType', '$RequestedUserID', '$RequestedUserType');