Course Project Phase 2 Report

Tyler Neal

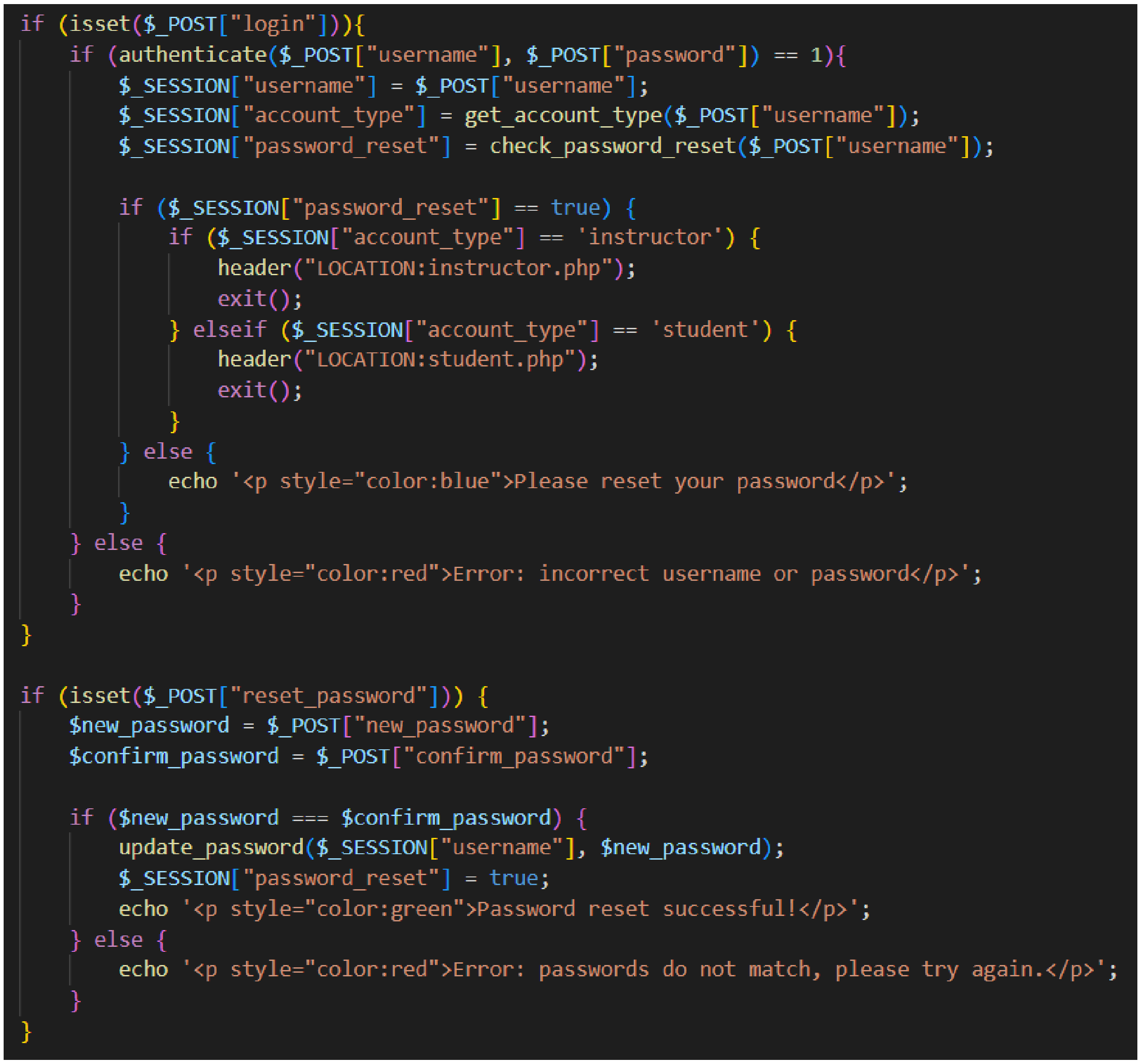
TABLE OF CONTENTS:

Instructor Functions:

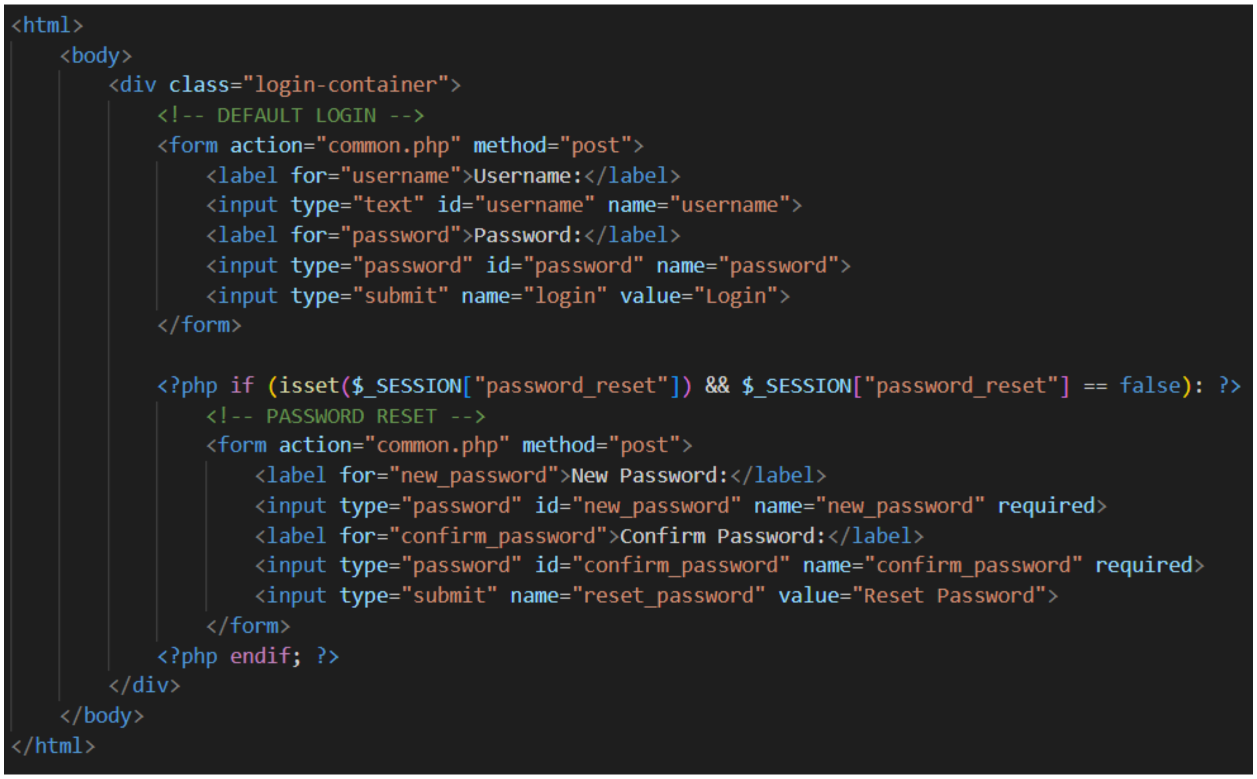
(database functions can be found at end of document)

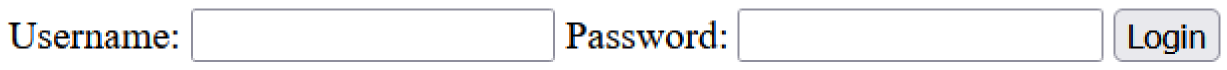
Instructor Page)

login & logout:

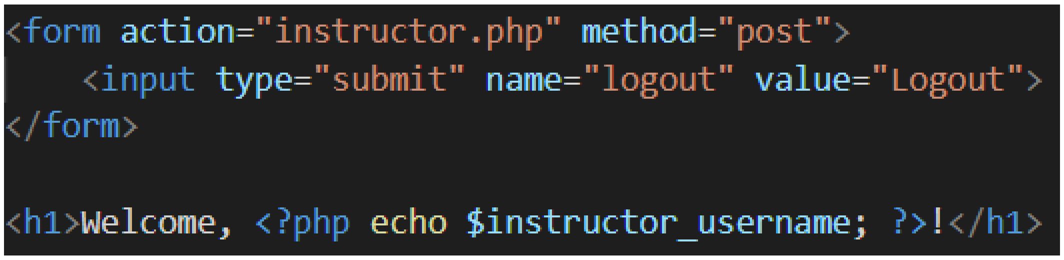


```php
if (isset($_POST["login"])){
    if (authenticate($_POST["username"], $_POST["password"]) == 1){
        $_SESSION["username"] = $_POST["username"];
        $_SESSION["account_type"] = get_account_type($_POST["username"]);
        $_SESSION["password_reset"] = check_password_reset($_POST["username"]);

        if ($_SESSION["password_reset"] == true) {
            if ($_SESSION["account_type"] == 'instructor') {
                header("LOCATION:instructor.php");
                exit();
            } elseif ($_SESSION["account_type"] == 'student') {
                header("LOCATION:student.php");
                exit();
            }
        } else {
            echo '<p style="color:blue">Please reset your password</p>';
        }
    } else {
        echo '<p style="color:red">Error: incorrect username or password</p>';
    }
}

if (isset($_POST["reset_password"])) {
    $new_password = $_POST["new_password"];
    $confirm_password = $_POST["confirm_password"];

    if ($new_password === $confirm_password) {
        update_password($_SESSION["username"], $new_password);
        $_SESSION["password_reset"] = true;
        echo '<p style="color:green">Password reset successful!</p>';
    } else {
        echo '<p style="color:red">Error: passwords do not match, please try again.</p>';
    }
}
```

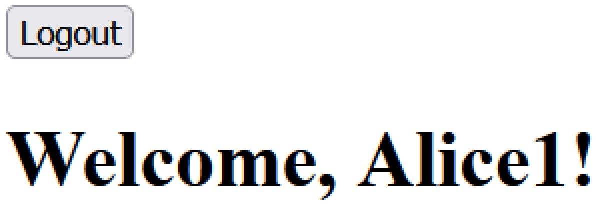

```html
<html>
    <body>
        <div class="login-container">
            <!-- DEFAULT LOGIN -->
            <form action="common.php" method="post">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username">
                <label for="password">Password:</label>
                <input type="password" id="password" name="password">
                <input type="submit" name="login" value="Login">
            </form>

            <?php if (isset($_SESSION["password_reset"]) && $_SESSION["password_reset"] == false): ?>
                <!-- PASSWORD RESET -->
                <form action="common.php" method="post">
                    <label for="new_password">New Password:</label>
                    <input type="password" id="new_password" name="new_password" required>
                    <label for="confirm_password">Confirm Password:</label>
                    <input type="password" id="confirm_password" name="confirm_password" required>
                    <input type="submit" name="reset_password" value="Reset Password">
                </form>
            <?php endif; ?>
        </div>
    </body>
</html>
```



welcome and logout button:



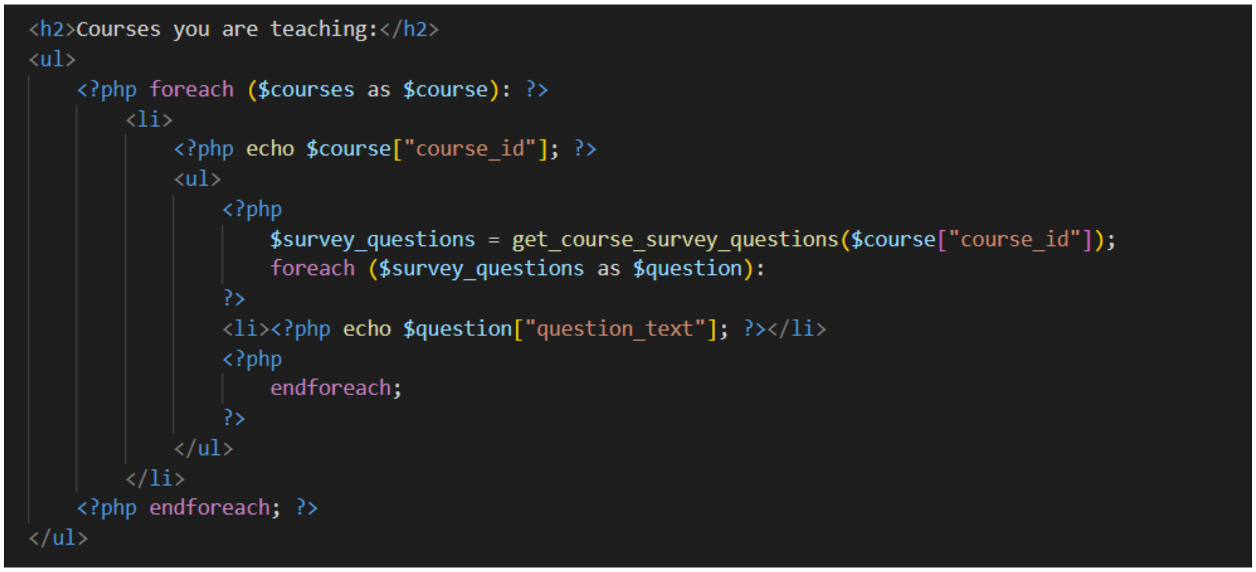
```
<form action="instructor.php" method="post">
    <input type="submit" name="logout" value="Logout">
</form>

<h1>Welcome, <?php echo $instructor_username; ?>!</h1>
```



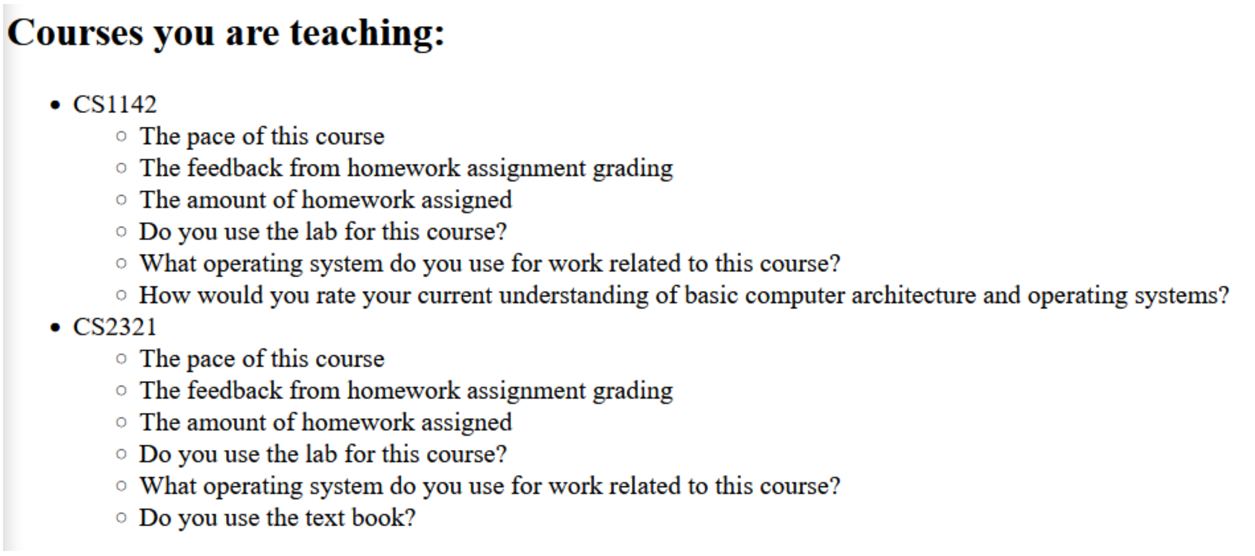
Logout

Welcome, Alice1!

list all assigned courses & associated survey questions:



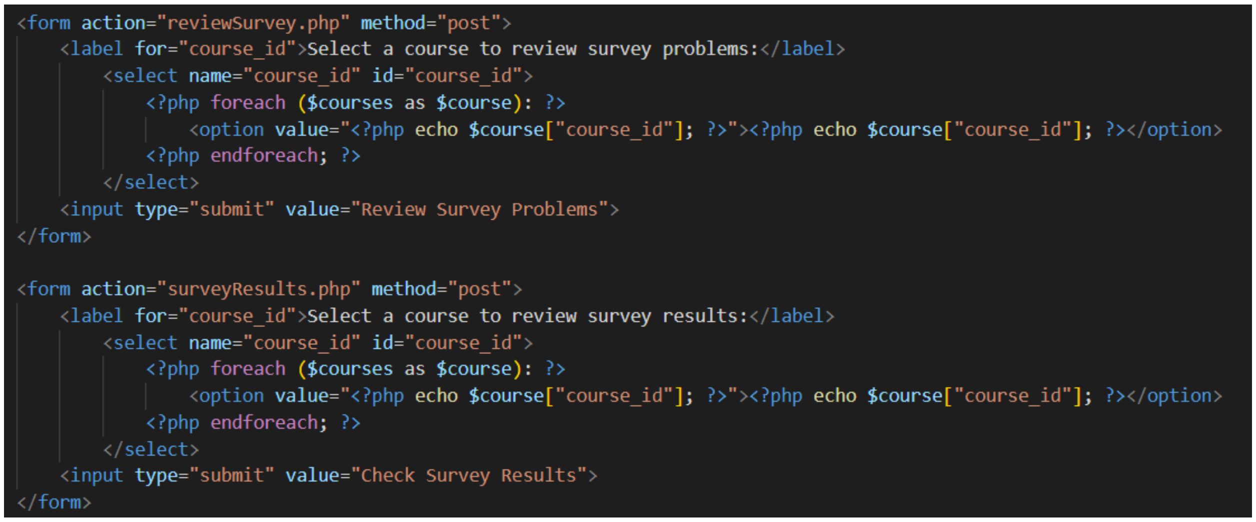
```
<h2>Courses you are teaching:</h2>
<ul>
    <?php foreach ($courses as $course): ?>
        <li>
            <?php echo $course["course_id"]; ?>
            <ul>
                <?php
                    $survey_questions = get_course_survey_questions($course["course_id"]);
                    foreach ($survey_questions as $question):
                ?>
                <li><?php echo $question["question_text"]; ?></li>
                <?php
                    endforeach;
                ?>
            </ul>
        </li>
    <?php endforeach; ?>
</ul>
```



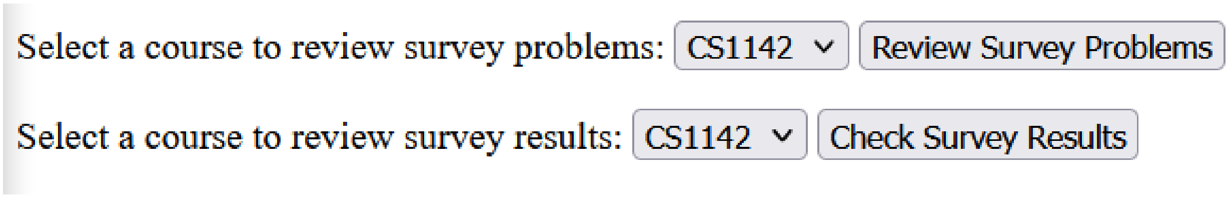
Courses you are teaching:

- CS1142
    - The pace of this course
    - The feedback from homework assignment grading
    - The amount of homework assigned
    - Do you use the lab for this course?
    - What operating system do you use for work related to this course?
    - How would you rate your current understanding of basic computer architecture and operating systems?
- CS2321
    - The pace of this course
    - The feedback from homework assignment grading
    - The amount of homework assigned
    - Do you use the lab for this course?
    - What operating system do you use for work related to this course?
    - Do you use the text book?

survey results and result buttons:



```
<form action="reviewSurvey.php" method="post">
    <label for="course_id">Select a course to review survey problems:</label>
        <select name="course_id" id="course_id">
            <?php foreach ($courses as $course): ?>
                <option value="<?php echo $course["course_id"]; ?>"><?php echo $course["course_id"]; ?></option>
            <?php endforeach; ?>
        </select>
    <input type="submit" value="Review Survey Problems">
</form>

<form action="surveyResults.php" method="post">
    <label for="course_id">Select a course to review survey results:</label>
        <select name="course_id" id="course_id">
            <?php foreach ($courses as $course): ?>
                <option value="<?php echo $course["course_id"]; ?>"><?php echo $course["course_id"]; ?></option>
            <?php endforeach; ?>
        </select>
    <input type="submit" value="Check Survey Results">
</form>
```

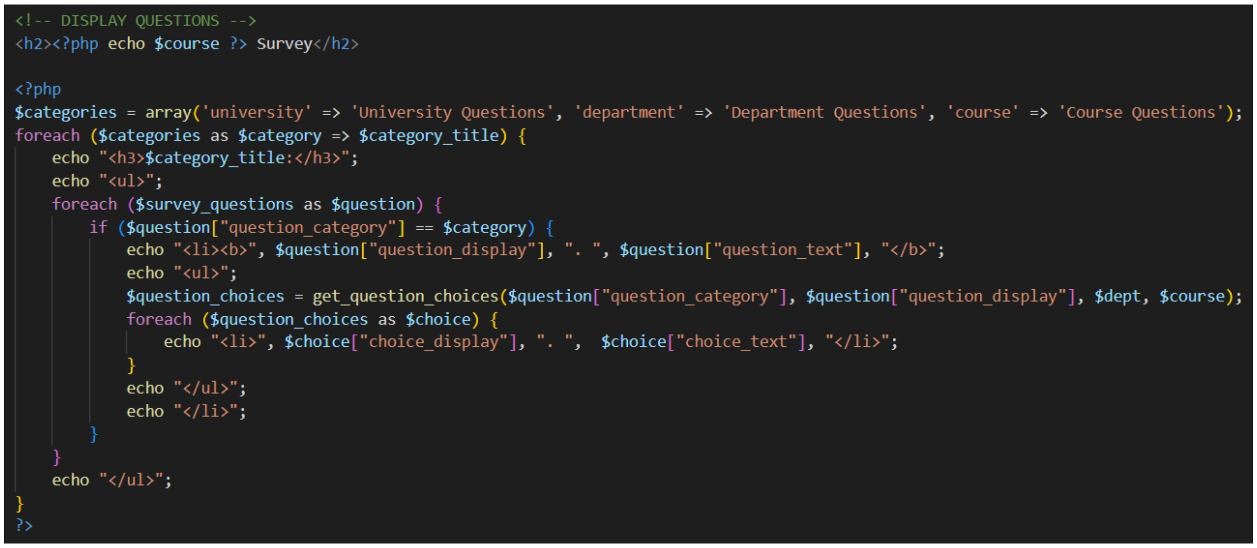


Select a course to review survey problems: CS1142 ⌄ Review Survey Problems

Select a course to review survey results: CS1142 ⌄ Check Survey Results

Review Survey Page)

display questions by category:



```
<!-- DISPLAY QUESTIONS -->
<h2><?php echo $course ?> Survey</h2>

<?php
$categories = array('university' => 'University Questions', 'department' => 'Department Questions', 'course' => 'Course Questions');
foreach ($categories as $category => $category_title) {
    echo "<h3>$category_title:</h3>";
    echo "<ul>";
    foreach ($survey_questions as $question) {
        if ($question["question_category"] == $category) {
            echo "<li><b>", $question["question_display"], ". ", $question["question_text"], "</b>";
            echo "<ul>";
            $question_choices = get_question_choices($question["question_category"], $question["question_display"], $dept, $course);
            foreach ($question_choices as $choice) {
                echo "<li>", $choice["choice_display"], ". ",  $choice["choice_text"], "</li>";
            }
            echo "</ul>";
            echo "</li>";
        }
    }
    echo "</ul>";
}
?>
```

CS1142 Survey

University Questions:

- **Q1. The pace of this course**
  - A. is too slow
  - B. is just right
  - C. is too fast
  - D. I dont know
- **Q2. The feedback from homework assignment grading**
  - A. Too few
  - B. Sufficient
  - C. I dont know
- **Q3. The amount of homework assigned**
  - A. Not enough
  - B. Just enough
  - C. A bit too much
  - D. Way too much
  - E. Indifferent

Department Questions:

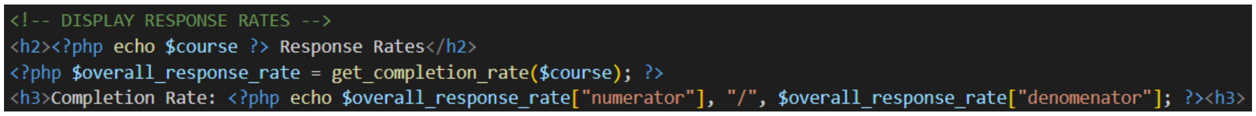
- **Q1. Do you use the lab for this course?**
  - A. Not at all
  - B. Occasionally
  - C. Sometimes
  - D. All the time
- **Q2. What operating system do you use for work related to this course?**
  - A. Mac
  - B. Windows
  - C. Linux

Course Questions:

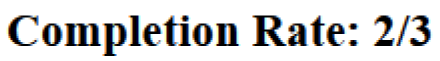
- **Q1. How would you rate your current understanding of basic computer architecture and operating systems?**
  - A. No knowledge
  - B. Beginner
  - C. Intermediate
  - D. Expert

Review Survey Results Page)

overall response rate:

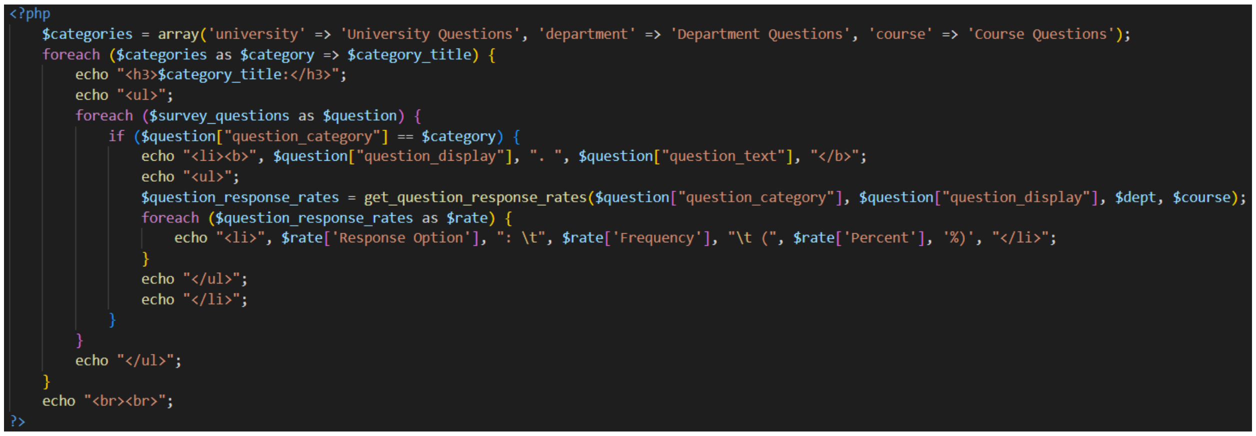


```
<!-- DISPLAY RESPONSE RATES -->
<h2><?php echo $course ?> Response Rates</h2>
<?php $overall_response_rate = get_completion_rate($course); ?>
<h3>Completion Rate: <?php echo $overall_response_rate["numerator"], "/", $overall_response_rate["denomenator"]; ?><h3>
```

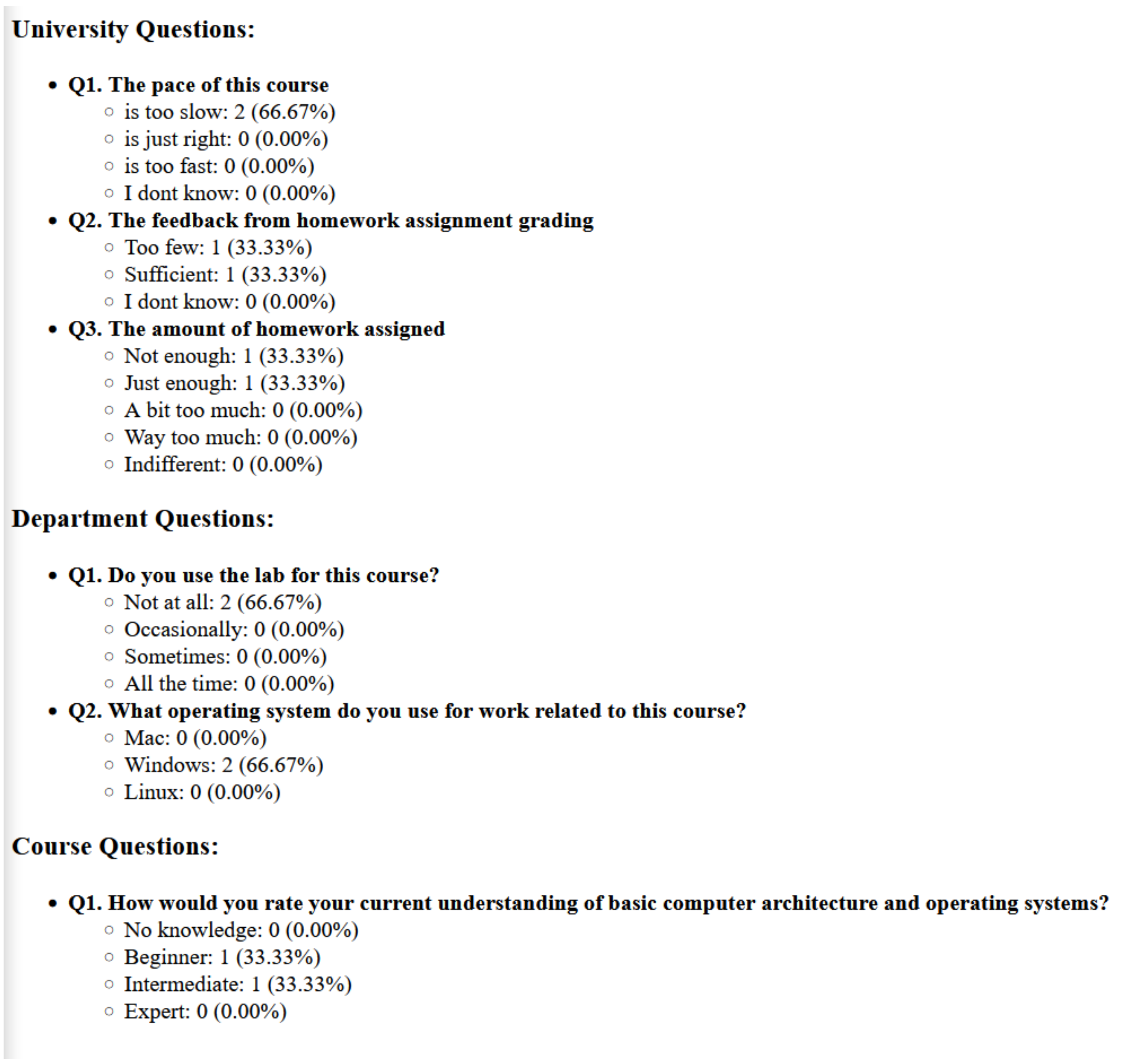


Completion Rate: 2/3

response frequencies:



```
<?php
    $categories = array('university' => 'University Questions', 'department' => 'Department Questions', 'course' => 'Course Questions');
    foreach ($categories as $category => $category_title) {
        echo "<h3>$category_title:</h3>";
        echo "<ul>";
        foreach ($survey_questions as $question) {
            if ($question["question_category"] == $category) {
                echo "<li><b>", $question["question_display"], ". ", $question["question_text"], "</b>";
                echo "<ul>";
                $question_response_rates = get_question_response_rates($question["question_category"], $question["question_display"], $dept, $course);
                foreach ($question_response_rates as $rate) {
                    echo "<li>", $rate['Response Option'], ": \t", $rate['Frequency'], "\t (", $rate['Percent'], '%)', "</li>";
                }
                echo "</ul>";
                echo "</li>";
            }
        }
        echo "</ul>";
    }
    echo "<br><br>";
?>
```



University Questions:

- **Q1. The pace of this course**
  - is too slow: 2 (66.67%)
  - is just right: 0 (0.00%)
  - is too fast: 0 (0.00%)
  - I dont know: 0 (0.00%)
- **Q2. The feedback from homework assignment grading**
  - Too few: 1 (33.33%)
  - Sufficient: 1 (33.33%)
  - I dont know: 0 (0.00%)
- **Q3. The amount of homework assigned**
  - Not enough: 1 (33.33%)
  - Just enough: 1 (33.33%)
  - A bit too much: 0 (0.00%)
  - Way too much: 0 (0.00%)
  - Indifferent: 0 (0.00%)

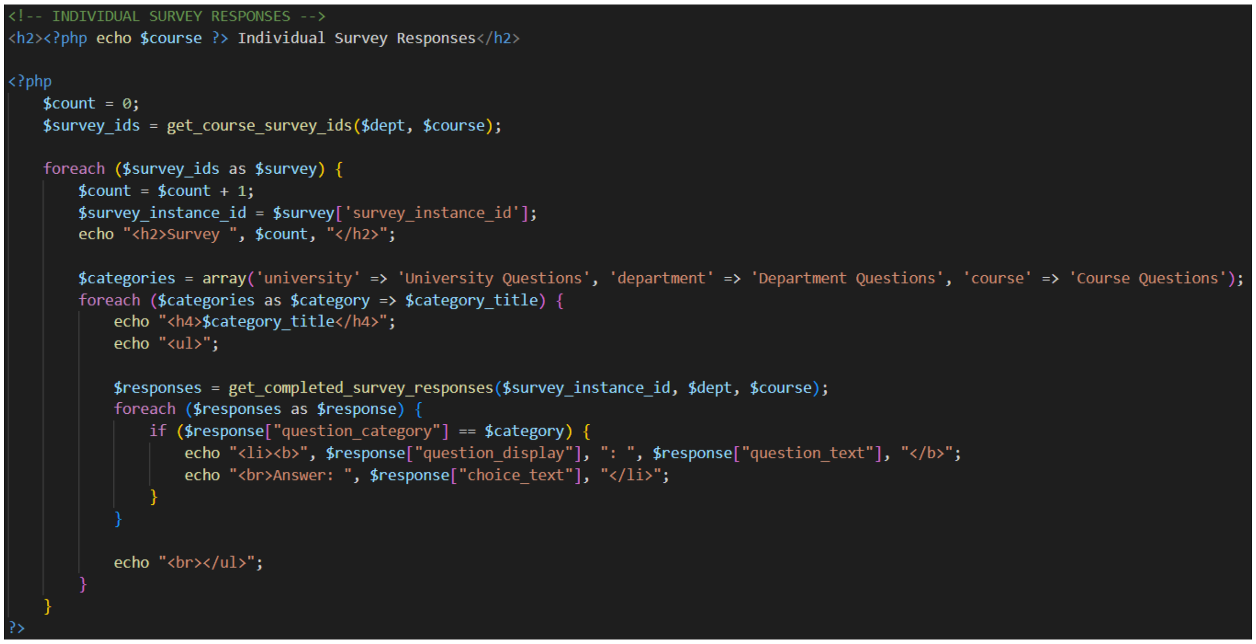
Department Questions:

- **Q1. Do you use the lab for this course?**
  - Not at all: 2 (66.67%)
  - Occasionally: 0 (0.00%)
  - Sometimes: 0 (0.00%)
  - All the time: 0 (0.00%)
- **Q2. What operating system do you use for work related to this course?**
  - Mac: 0 (0.00%)
  - Windows: 2 (66.67%)
  - Linux: 0 (0.00%)

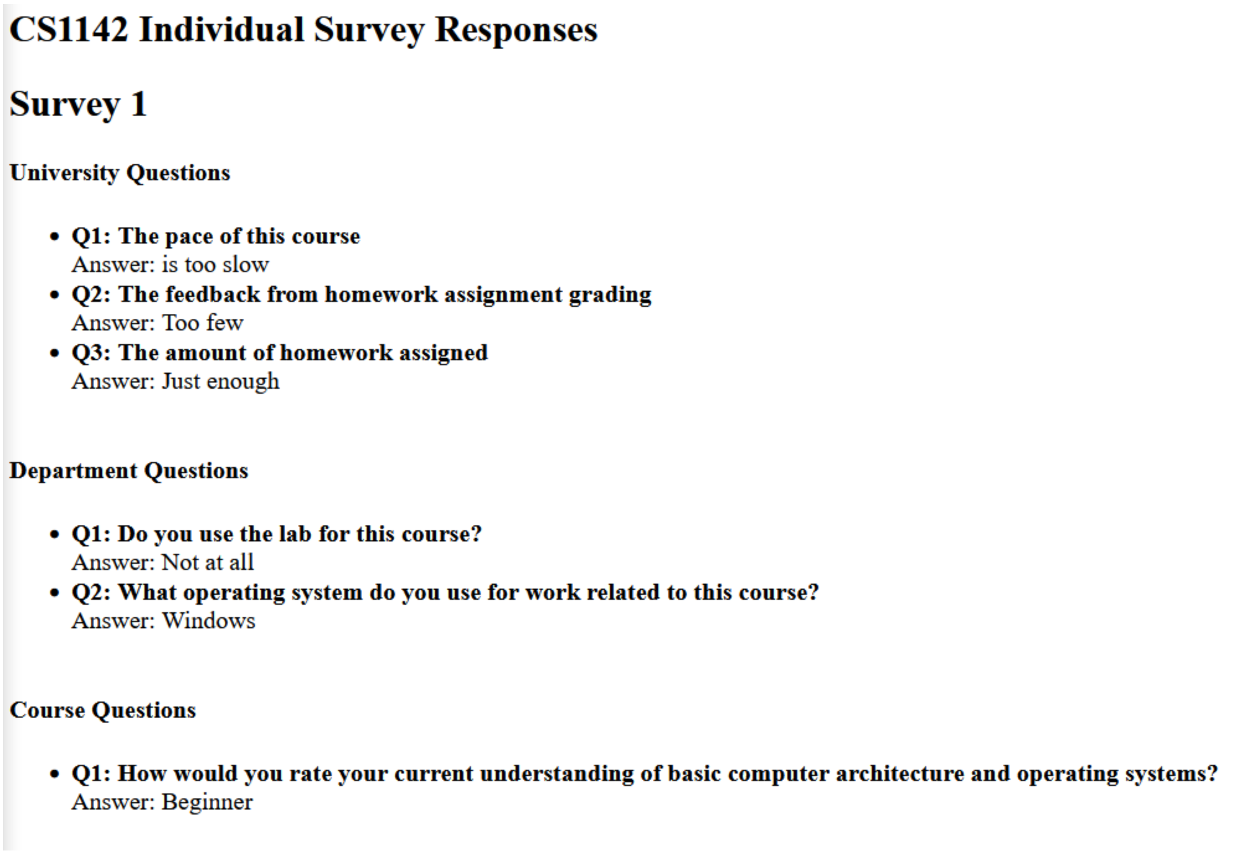
Course Questions:

- **Q1. How would you rate your current understanding of basic computer architecture and operating systems?**
  - No knowledge: 0 (0.00%)
  - Beginner: 1 (33.33%)
  - Intermediate: 1 (33.33%)
  - Expert: 0 (0.00%)

individual surveys:



```
<!-- INDIVIDUAL SURVEY RESPONSES -->
<h2><?php echo $course ?> Individual Survey Responses</h2>

<?php
    $count = 0;
    $survey_ids = get_course_survey_ids($dept, $course);

    foreach ($survey_ids as $survey) {
        $count = $count + 1;
        $survey_instance_id = $survey['survey_instance_id'];
        echo "<h2>Survey ", $count, "</h2>";

        $categories = array('university' => 'University Questions', 'department' => 'Department Questions', 'course' => 'Course Questions');
        foreach ($categories as $category => $category_title) {
            echo "<h4>$category_title</h4>";
            echo "<ul>";

            $responses = get_completed_survey_responses($survey_instance_id, $dept, $course);
            foreach ($responses as $response) {
                if ($response["question_category"] == $category) {
                    echo "<li><b>", $response["question_display"], ": ", $response["question_text"], "</b>";
                    echo "<br>Answer: ", $response["choice_text"], "</li>";
                }
            }

            echo "<br></ul>";
        }
    }
?>
```



CS1142 Individual Survey Responses

Survey 1

University Questions

- **Q1: The pace of this course**
  Answer: is too slow
- **Q2: The feedback from homework assignment grading**
  Answer: Too few
- **Q3: The amount of homework assigned**
  Answer: Just enough

Department Questions

- **Q1: Do you use the lab for this course?**
  Answer: Not at all
- **Q2: What operating system do you use for work related to this course?**
  Answer: Windows

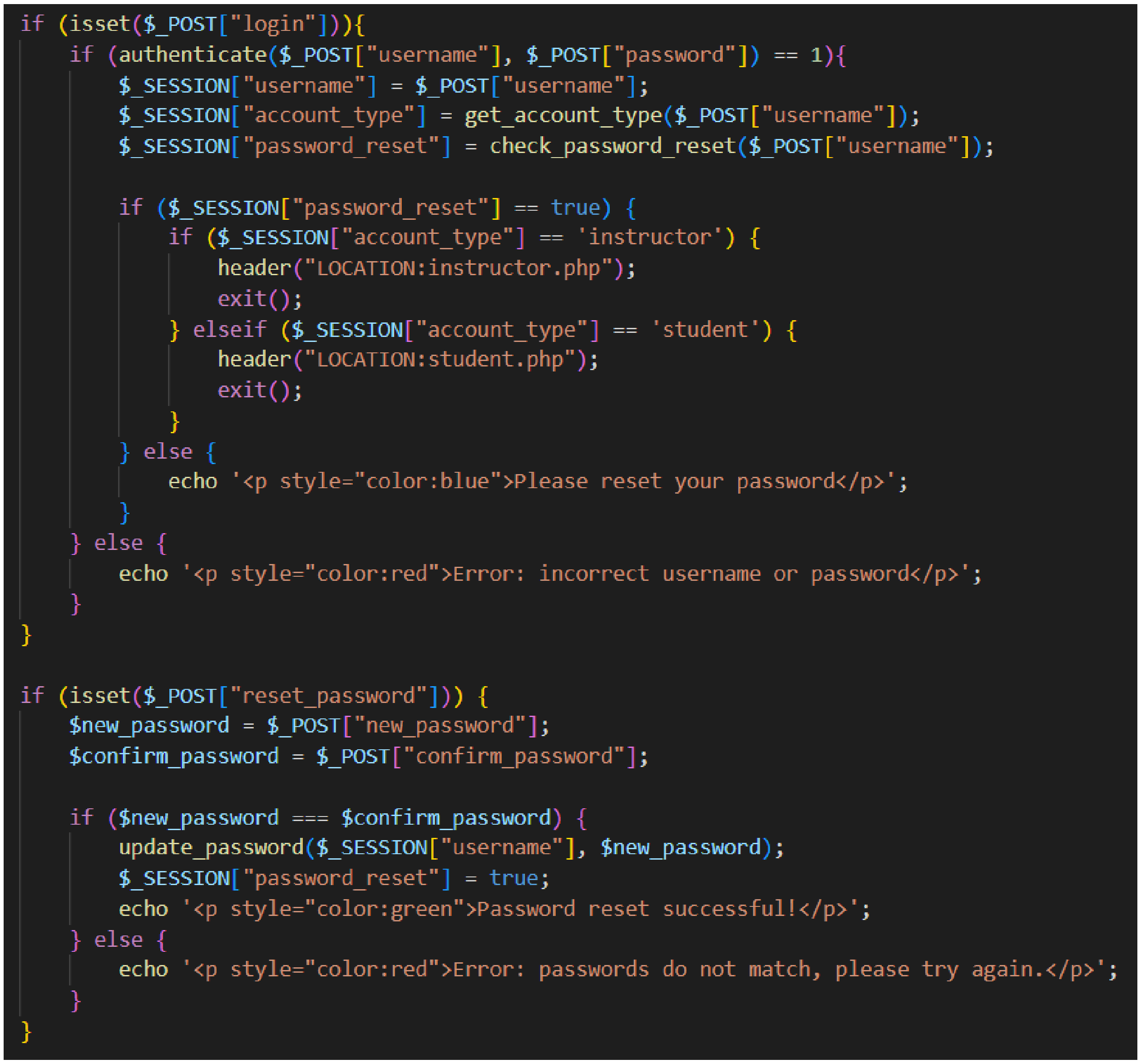
Course Questions

- **Q1: How would you rate your current understanding of basic computer architecture and operating systems?**
  Answer: Beginner

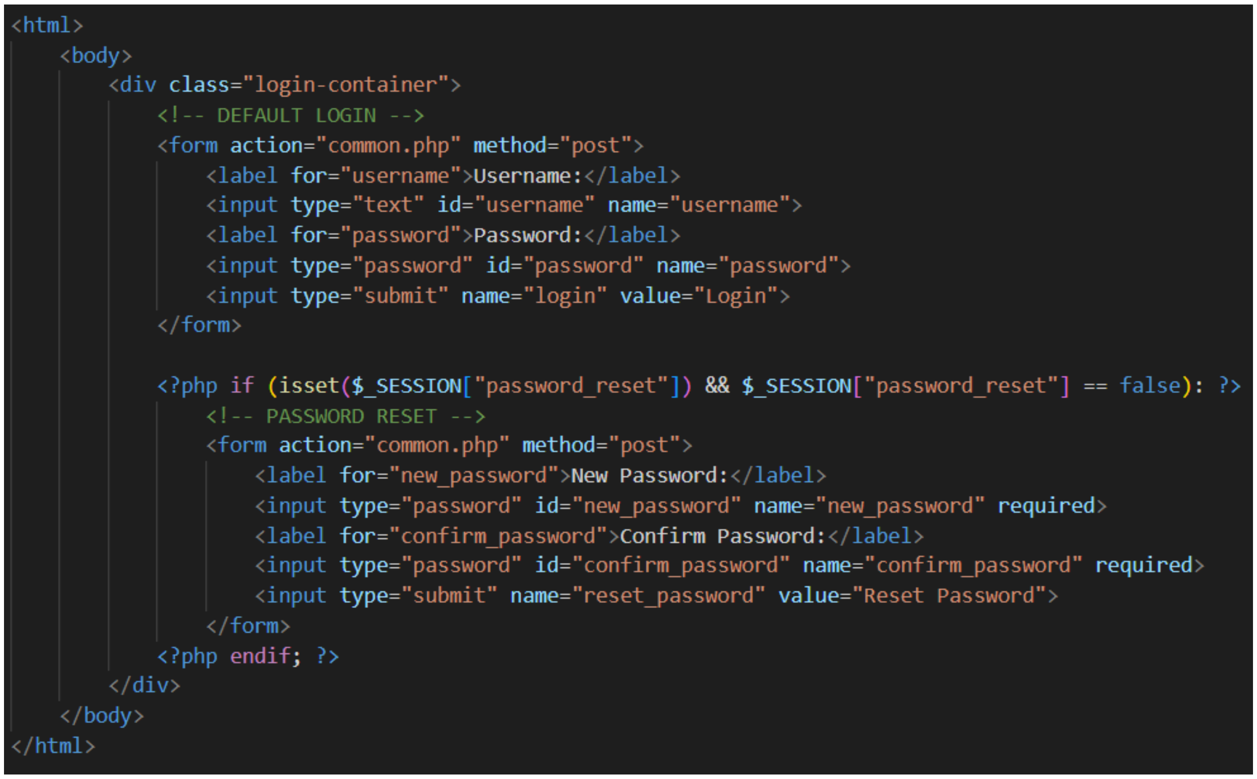
Student Functions:

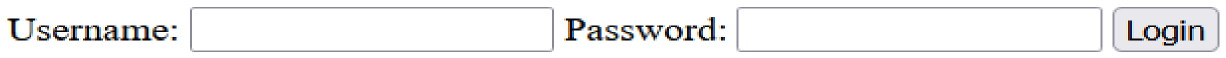
Student Page)

login & logout (same as instructor):

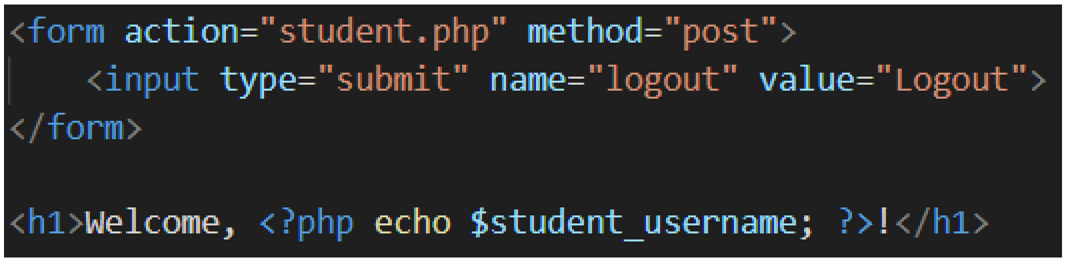


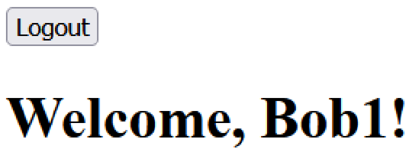
```php
if (isset($_POST["login"])){
    if (authenticate($_POST["username"], $_POST["password"]) == 1){
        $_SESSION["username"] = $_POST["username"];
        $_SESSION["account_type"] = get_account_type($_POST["username"]);
        $_SESSION["password_reset"] = check_password_reset($_POST["username"]);

        if ($_SESSION["password_reset"] == true) {
            if ($_SESSION["account_type"] == 'instructor') {
                header("LOCATION:instructor.php");
                exit();
            } elseif ($_SESSION["account_type"] == 'student') {
                header("LOCATION:student.php");
                exit();
            }
        } else {
            echo '<p style="color:blue">Please reset your password</p>';
        }
    } else {
        echo '<p style="color:red">Error: incorrect username or password</p>';
    }
}

if (isset($_POST["reset_password"])) {
    $new_password = $_POST["new_password"];
    $confirm_password = $_POST["confirm_password"];

    if ($new_password === $confirm_password) {
        update_password($_SESSION["username"], $new_password);
        $_SESSION["password_reset"] = true;
        echo '<p style="color:green">Password reset successful!</p>';
    } else {
        echo '<p style="color:red">Error: passwords do not match, please try again.</p>';
    }
}
```


```
<html>
    <body>
        <div class="login-container">
            <!-- DEFAULT LOGIN -->
            <form action="common.php" method="post">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username">
                <label for="password">Password:</label>
                <input type="password" id="password" name="password">
                <input type="submit" name="login" value="Login">
            </form>

            <?php if (isset($_SESSION["password_reset"]) && $_SESSION["password_reset"] == false): ?>
                <!-- PASSWORD RESET -->
                <form action="common.php" method="post">
                    <label for="new_password">New Password:</label>
                    <input type="password" id="new_password" name="new_password" required>
                    <label for="confirm_password">Confirm Password:</label>
                    <input type="password" id="confirm_password" name="confirm_password" required>
                    <input type="submit" name="reset_password" value="Reset Password">
                </form>
            <?php endif; ?>
        </div>
    </body>
</html>
```

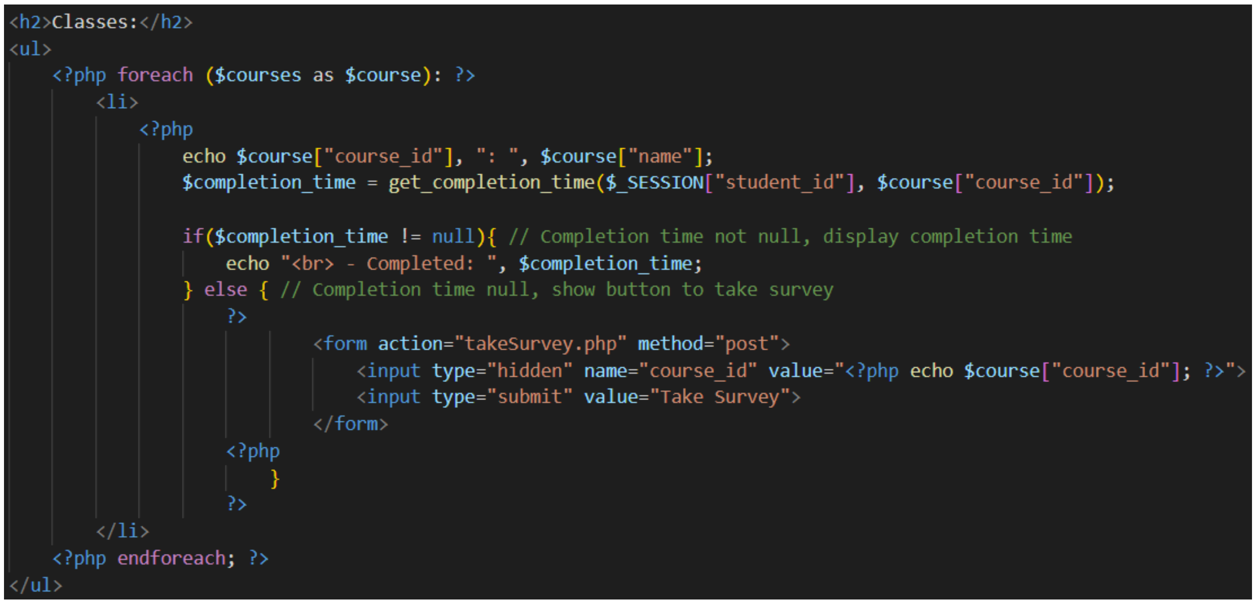


welcome and logout button:

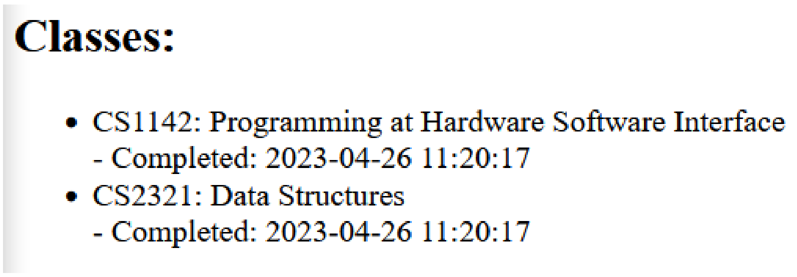

```
<form action="student.php" method="post">
    <input type="submit" name="logout" value="Logout">
</form>

<h1>Welcome, <?php echo $student_username; ?>!</h1>
```



display enrolled classes & survey completion time:



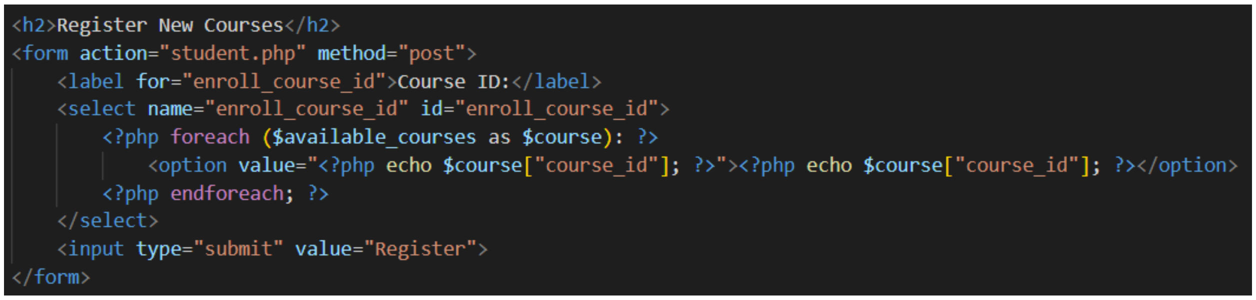
```
<h2>Classes:</h2>
<ul>
    <?php foreach ($courses as $course): ?>
        <li>
            <?php
                echo $course["course_id"], ": ", $course["name"];
                $completion_time = get_completion_time($_SESSION["student_id"], $course["course_id"]);

                if($completion_time != null){ // Completion time not null, display completion time
                    echo "<br> - Completed: ", $completion_time;
                } else { // Completion time null, show button to take survey
                    ?>
                        <form action="takeSurvey.php" method="post">
                            <input type="hidden" name="course_id" value="<?php echo $course["course_id"]; ?>">
                            <input type="submit" value="Take Survey">
                        </form>
                    <?php
                    }
                    ?>
        </li>
    <?php endforeach; ?>
</ul>
```



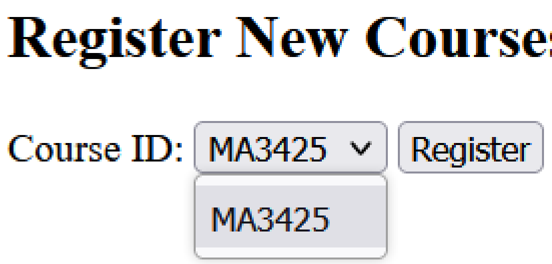
Classes:

- CS1142: Programming at Hardware Software Interface
  - Completed: 2023-04-26 11:20:17
- CS2321: Data Structures
  - Completed: 2023-04-26 11:20:17

list courses not registered and register for courses:



```
<h2>Register New Courses</h2>
<form action="student.php" method="post">
    <label for="enroll_course_id">Course ID:</label>
    <select name="enroll_course_id" id="enroll_course_id">
        <?php foreach ($available_courses as $course): ?>
            <option value="<?php echo $course["course_id"]; ?>"><?php echo $course["course_id"]; ?></option>
        <?php endforeach; ?>
    </select>
    <input type="submit" value="Register">
</form>
```

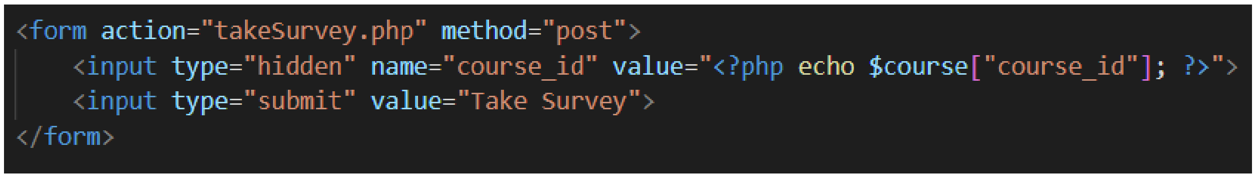


Register New Courses

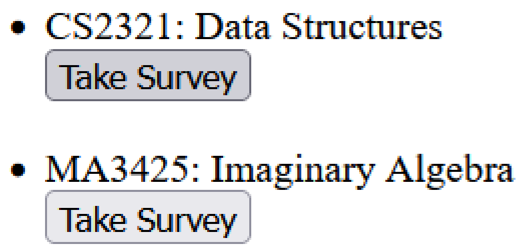
Course ID: MA3425 Register

MA3425

take survey button:

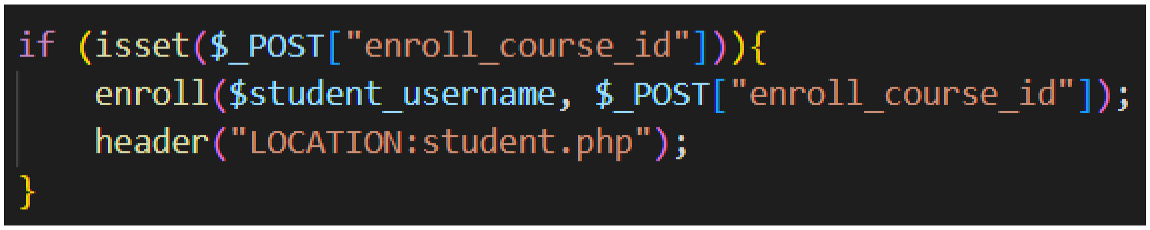


```
<form action="takeSurvey.php" method="post">
    <input type="hidden" name="course_id" value="<?php echo $course["course_id"]; ?>">
    <input type="submit" value="Take Survey">
</form>
```



- CS2321: Data Structures
  Take Survey
- MA3425: Imaginary Algebra
  Take Survey

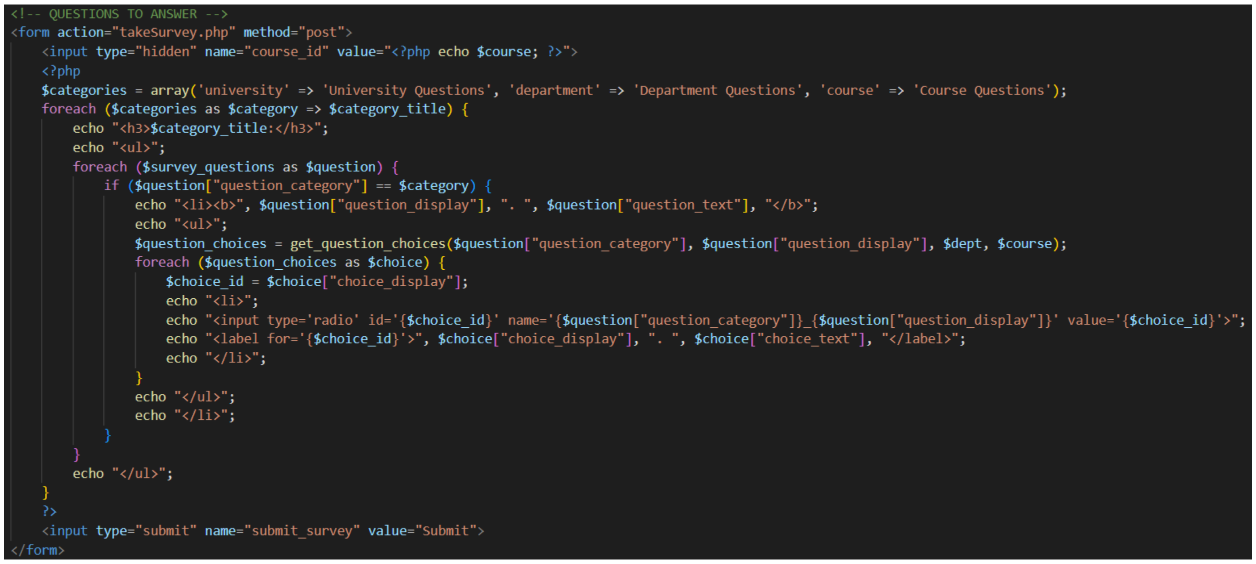
register for class:



```
if (isset($_POST["enroll_course_id"])){
    enroll($student_username, $_POST["enroll_course_id"]);
    header("LOCATION:student.php");
}
```

Survey Page)

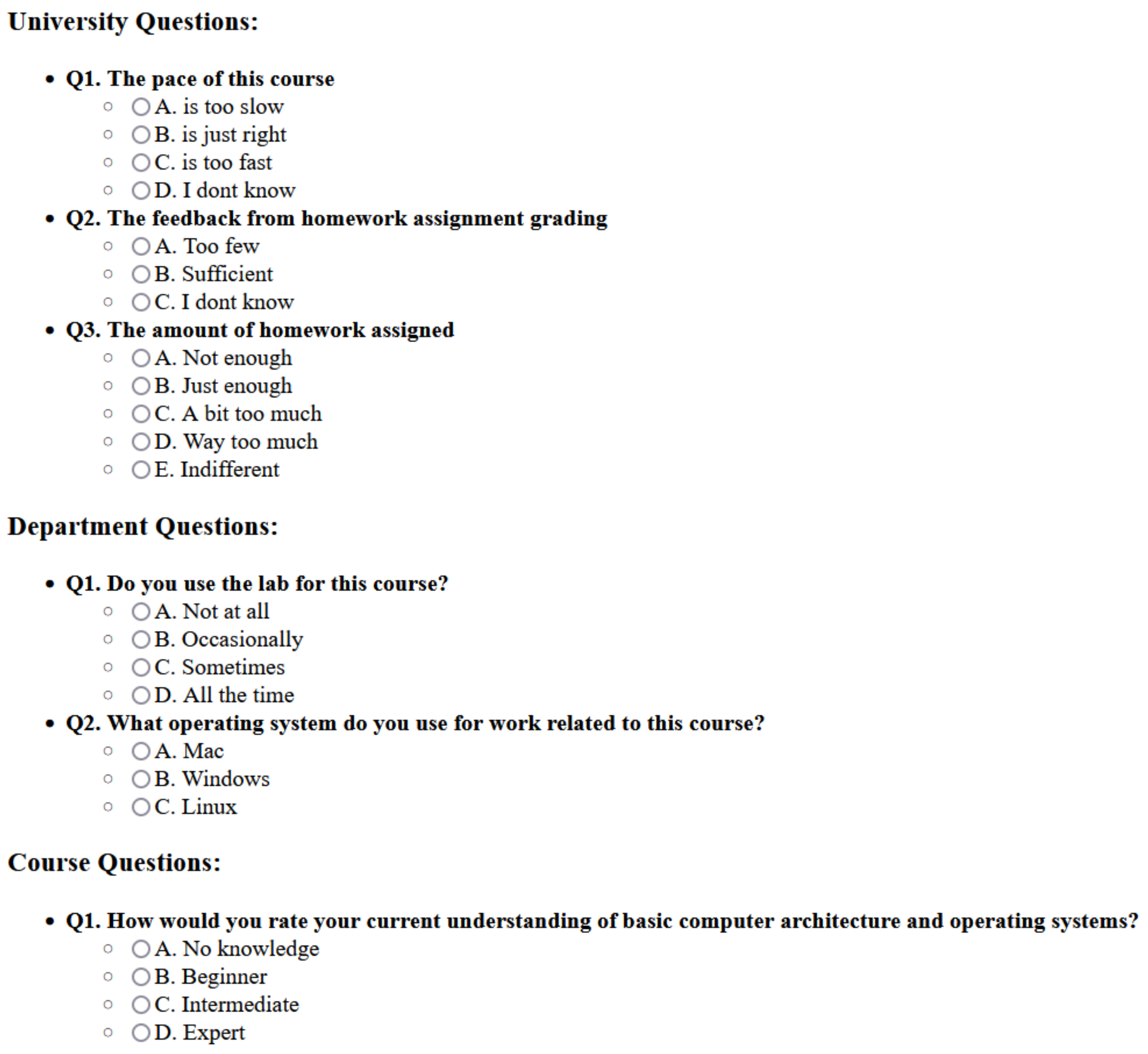
display survey questions to answer:



```
<!-- QUESTIONS TO ANSWER -->
<form action="takeSurvey.php" method="post">
    <input type="hidden" name="course_id" value="<?php echo $course; ?>">
    <?php
    $categories = array('university' => 'University Questions', 'department' => 'Department Questions', 'course' => 'Course Questions');
    foreach ($categories as $category => $category_title) {
        echo "<h3>$category_title:</h3>";
        echo "<ul>";
        foreach ($survey_questions as $question) {
            if ($question["question_category"] == $category) {
                echo "<li><b>", $question["question_display"], ". ", $question["question_text"], "</b>";
                echo "<ul>";
                $question_choices = get_question_choices($question["question_category"], $question["question_display"], $dept, $course);
                foreach ($question_choices as $choice) {
                    $choice_id = $choice["choice_display"];
                    echo "<li>";
                    echo "<input type='radio' id='{$choice_id}' name='{$question["question_category"]}_{$question["question_display"]}' value='{$choice_id}'>";
                    echo "<label for='{$choice_id}'>", $choice["choice_display"], ". ", $choice["choice_text"], "</label>";
                    echo "</li>";
                }
                echo "</ul>";
                echo "</li>";
            }
        }
        echo "</ul>";
    }
    ?>
    <input type="submit" name="submit_survey" value="Submit">
</form>
```

example of two students from different courses in same department:

Bobby in CS1142:



University Questions:

- **Q1. The pace of this course**
    - ○ A. is too slow
    - ○ B. is just right
    - ○ C. is too fast
    - ○ D. I dont know
- **Q2. The feedback from homework assignment grading**
    - ○ A. Too few
    - ○ B. Sufficient
    - ○ C. I dont know
- **Q3. The amount of homework assigned**
    - ○ A. Not enough
    - ○ B. Just enough
    - ○ C. A bit too much
    - ○ D. Way too much
    - ○ E. Indifferent

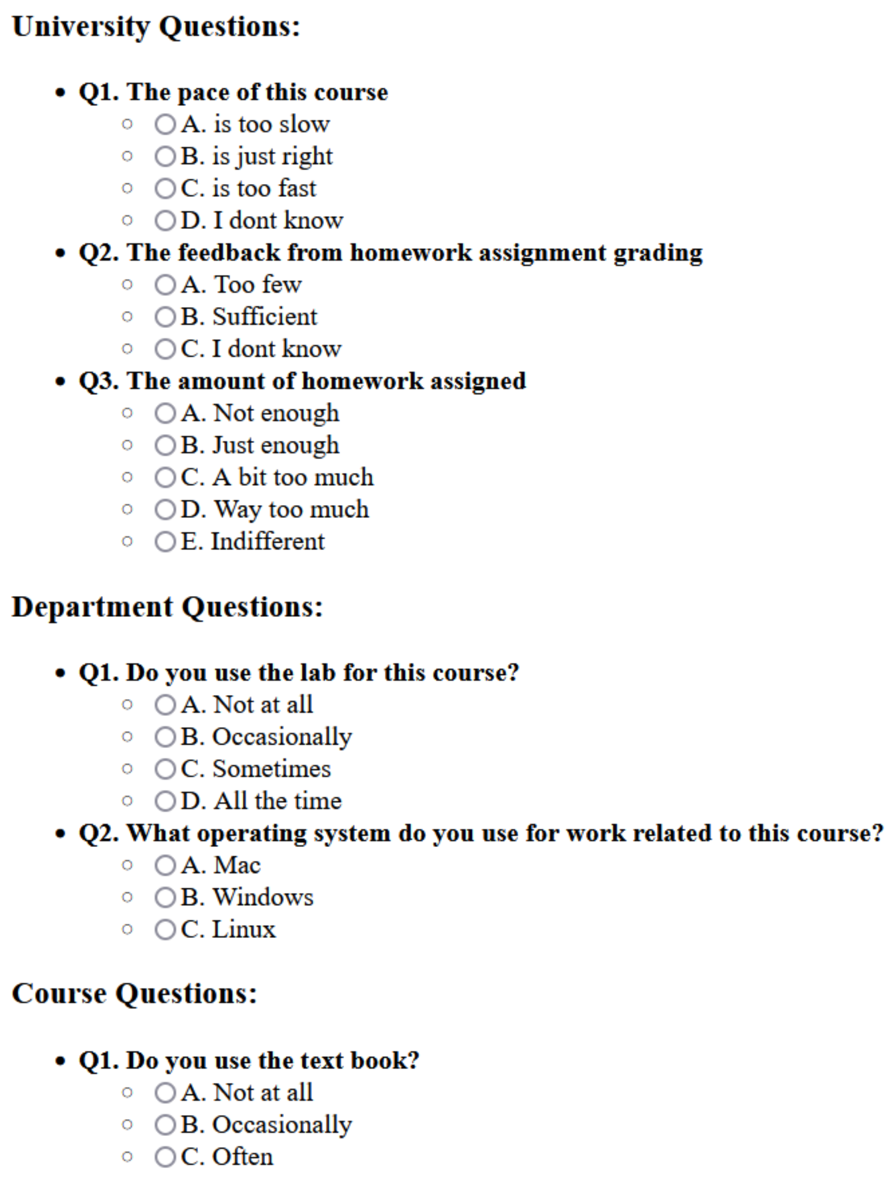
Department Questions:

- **Q1. Do you use the lab for this course?**
    - ○ A. Not at all
    - ○ B. Occasionally
    - ○ C. Sometimes
    - ○ D. All the time
- **Q2. What operating system do you use for work related to this course?**
    - ○ A. Mac
    - ○ B. Windows
    - ○ C. Linux

Course Questions:

- **Q1. How would you rate your current understanding of basic computer architecture and operating systems?**
    - ○ A. No knowledge
    - ○ B. Beginner
    - ○ C. Intermediate
    - ○ D. Expert

Billy in CS2321:



University Questions:

- **Q1. The pace of this course**
  - ◯A. is too slow
  - ◯B. is just right
  - ◯C. is too fast
  - ◯D. I dont know
- **Q2. The feedback from homework assignment grading**
  - ◯A. Too few
  - ◯B. Sufficient
  - ◯C. I dont know
- **Q3. The amount of homework assigned**
  - ◯A. Not enough
  - ◯B. Just enough
  - ◯C. A bit too much
  - ◯D. Way too much
  - ◯E. Indifferent

Department Questions:

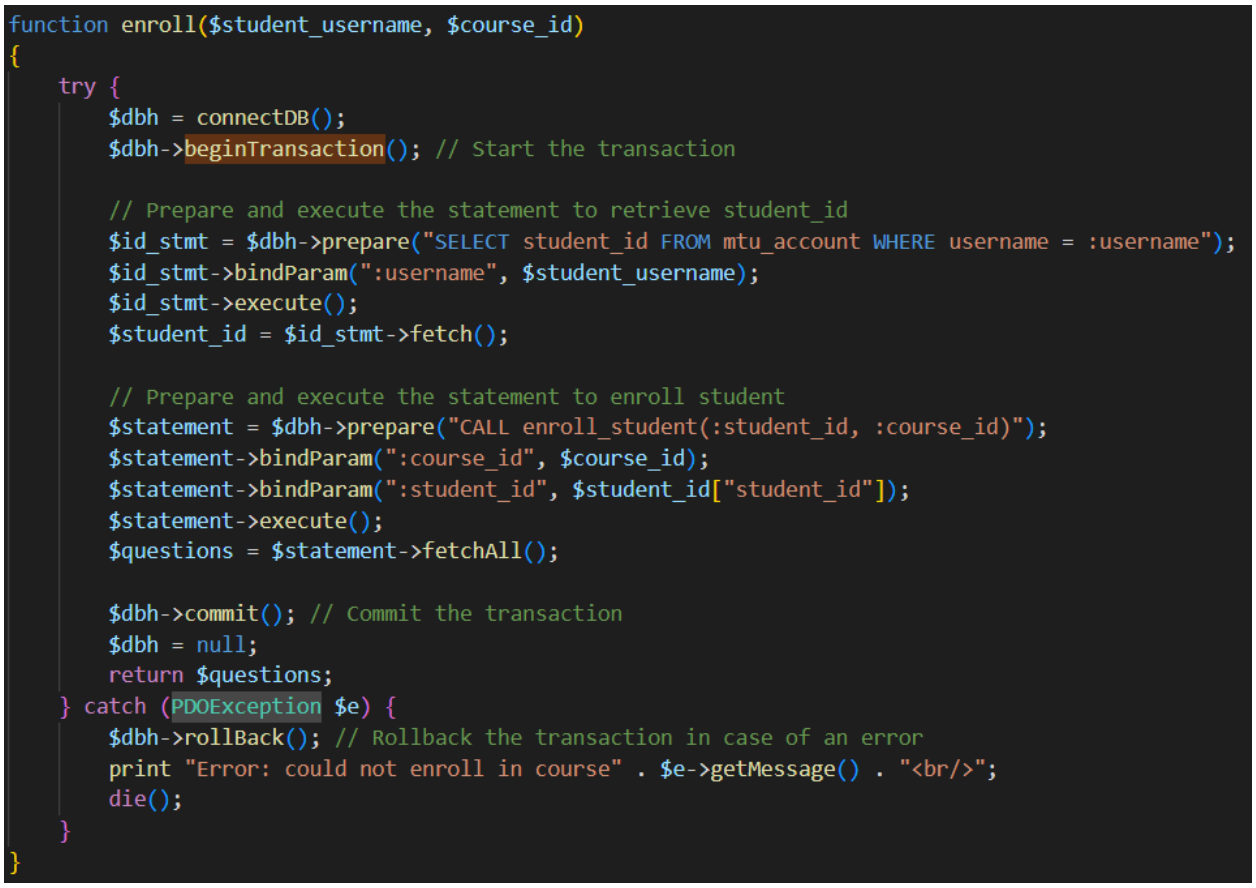
- **Q1. Do you use the lab for this course?**
  - ◯A. Not at all
  - ◯B. Occasionally
  - ◯C. Sometimes
  - ◯D. All the time
- **Q2. What operating system do you use for work related to this course?**
  - ◯A. Mac
  - ◯B. Windows
  - ◯C. Linux

Course Questions:

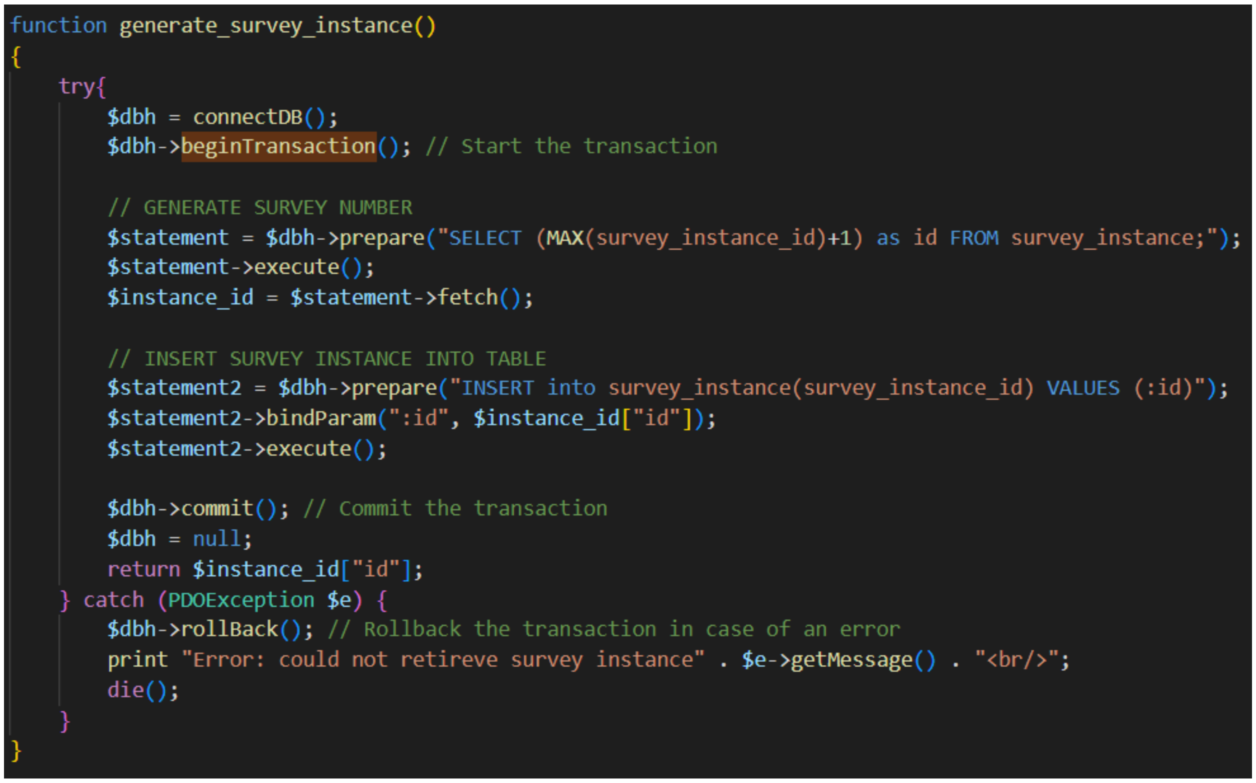
- **Q1. Do you use the text book?**
  - ◯A. Not at all
  - ◯B. Occasionally
  - ◯C. Often

Use of Transaction:

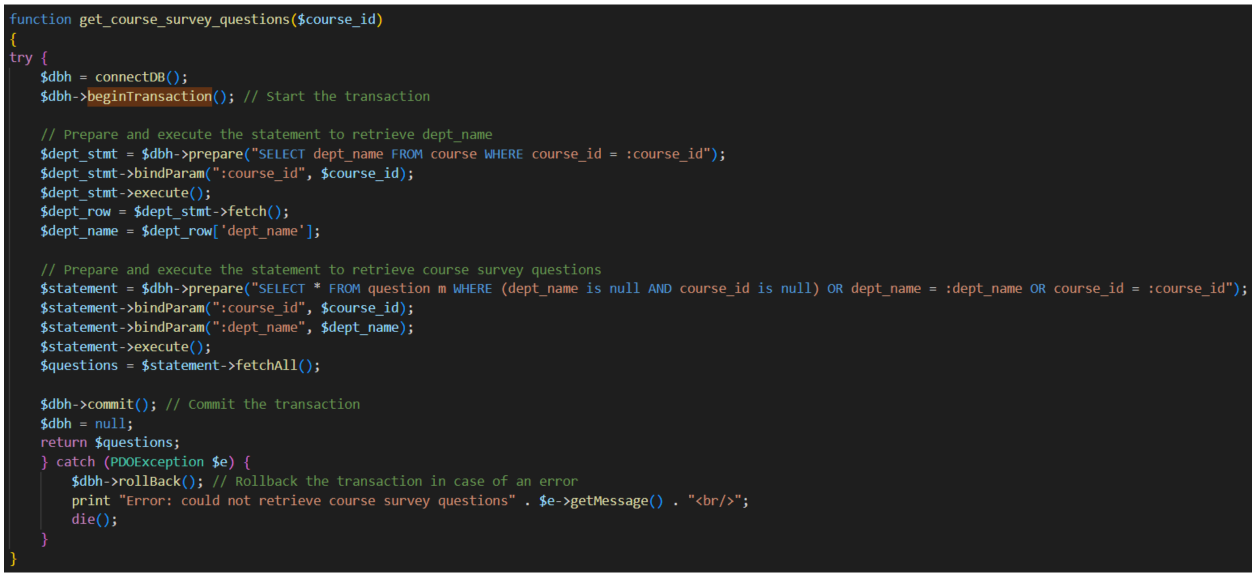
I used transaction for my database functions whenever I had to execute two or more statements inside a single function call. This is usually when I needed to query another variable for the later call, without the need of inputting too many parameters into the original function. I've provided some examples below. Doing this provides the properties of ACID, and prevents dirty reads.



```php
function enroll($student_username, $course_id)
{
    try {
        $dbh = connectDB();
        $dbh->beginTransaction(); // Start the transaction

        // Prepare and execute the statement to retrieve student_id
        $id_stmt = $dbh->prepare("SELECT student_id FROM mtu_account WHERE username = :username");
        $id_stmt->bindParam(":username", $student_username);
        $id_stmt->execute();
        $student_id = $id_stmt->fetch();

        // Prepare and execute the statement to enroll student
        $statement = $dbh->prepare("CALL enroll_student(:student_id, :course_id)");
        $statement->bindParam(":course_id", $course_id);
        $statement->bindParam(":student_id", $student_id["student_id"]);
        $statement->execute();
        $questions = $statement->fetchAll();

        $dbh->commit(); // Commit the transaction
        $dbh = null;
        return $questions;
    } catch (PDOException $e) {
        $dbh->rollBack(); // Rollback the transaction in case of an error
        print "Error: could not enroll in course" . $e->getMessage() . "<br/>";
        die();
    }
}
```



```php
function generate_survey_instance()
{
    try{
        $dbh = connectDB();
        $dbh->beginTransaction(); // Start the transaction

        // GENERATE SURVEY NUMBER
        $statement = $dbh->prepare("SELECT (MAX(survey_instance_id)+1) as id FROM survey_instance;");
        $statement->execute();
        $instance_id = $statement->fetch();

        // INSERT SURVEY INSTANCE INTO TABLE
        $statement2 = $dbh->prepare("INSERT into survey_instance(survey_instance_id) VALUES (:id)");
        $statement2->bindParam(":id", $instance_id["id"]);
        $statement2->execute();

        $dbh->commit(); // Commit the transaction
        $dbh = null;
        return $instance_id["id"];
    } catch (PDOException $e) {
        $dbh->rollBack(); // Rollback the transaction in case of an error
        print "Error: could not retireve survey instance" . $e->getMessage() . "<br/>";
        die();
    }
}
```



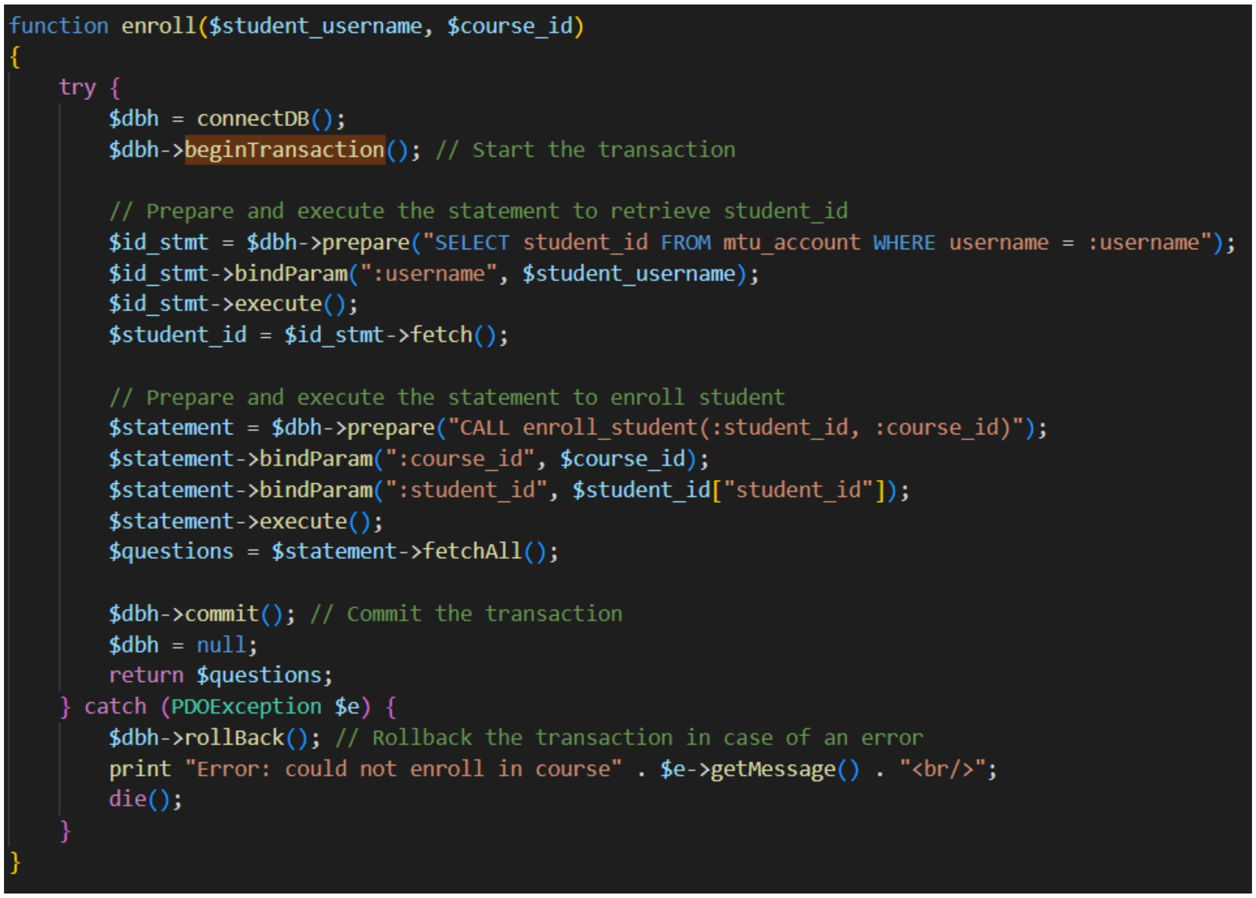
```php
function get_course_survey_questions($course_id)
{
try {
    $dbh = connectDB();
    $dbh->beginTransaction(); // Start the transaction

    // Prepare and execute the statement to retrieve dept_name
    $dept_stmt = $dbh->prepare("SELECT dept_name FROM course WHERE course_id = :course_id");
    $dept_stmt->bindParam(":course_id", $course_id);
    $dept_stmt->execute();
    $dept_row = $dept_stmt->fetch();
    $dept_name = $dept_row['dept_name'];

    // Prepare and execute the statement to retrieve course survey questions
    $statement = $dbh->prepare("SELECT * FROM question m WHERE (dept_name is null AND course_id is null) OR dept_name = :dept_name OR course_id = :course_id");
    $statement->bindParam(":course_id", $course_id);
    $statement->bindParam(":dept_name", $dept_name);
    $statement->execute();
    $questions = $statement->fetchAll();

    $dbh->commit(); // Commit the transaction
    $dbh = null;
    return $questions;
    } catch (PDOException $e) {
        $dbh->rollBack(); // Rollback the transaction in case of an error
        print "Error: could not retrieve course survey questions" . $e->getMessage() . "<br/>";
        die();
    }
}
```

Preventing SQL Injection

The first thing I’ve done to prevent SQL injection is through the use of prepared statements. In all the database functions, I have utilized prepared statements with placeholders to separate the SQL query structure from the actual data. By using the prepare() function and binding parameters with bindParam(), I ensure that user-supplied data is treated as separate from the query string, thus preventing SQL injection. Prepared statements are effective because they don't allow user input to interfere with the SQL query structure.

I’ve also used try and catch blocks to handle any exceptions thrown by the PDO operations. This helps ensure that if an error occurs during the database operation, a controlled error message is displayed to the user, rather than revealing sensitive information about the application or database structure.

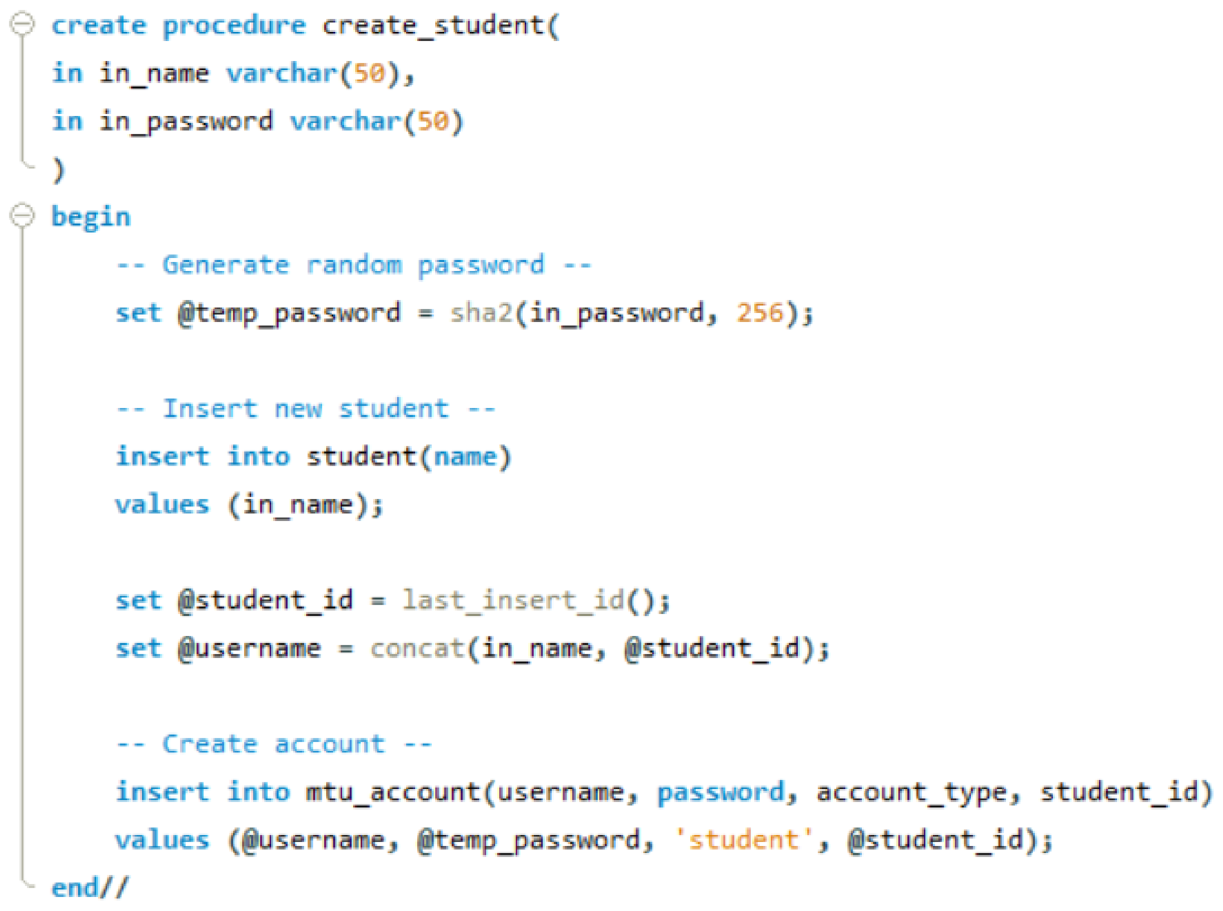
Example of both:



```php
function enroll($student_username, $course_id)
{
    try {
        $dbh = connectDB();
        $dbh->beginTransaction(); // Start the transaction

        // Prepare and execute the statement to retrieve student_id
        $id_stmt = $dbh->prepare("SELECT student_id FROM mtu_account WHERE username = :username");
        $id_stmt->bindParam(":username", $student_username);
        $id_stmt->execute();
        $student_id = $id_stmt->fetch();

        // Prepare and execute the statement to enroll student
        $statement = $dbh->prepare("CALL enroll_student(:student_id, :course_id)");
        $statement->bindParam(":course_id", $course_id);
        $statement->bindParam(":student_id", $student_id["student_id"]);
        $statement->execute();
        $questions = $statement->fetchAll();

        $dbh->commit(); // Commit the transaction
        $dbh = null;
        return $questions;
    } catch (PDOException $e) {
        $dbh->rollBack(); // Rollback the transaction in case of an error
        print "Error: could not enroll in course" . $e->getMessage() . "<br/>";
        die();
    }
}
```

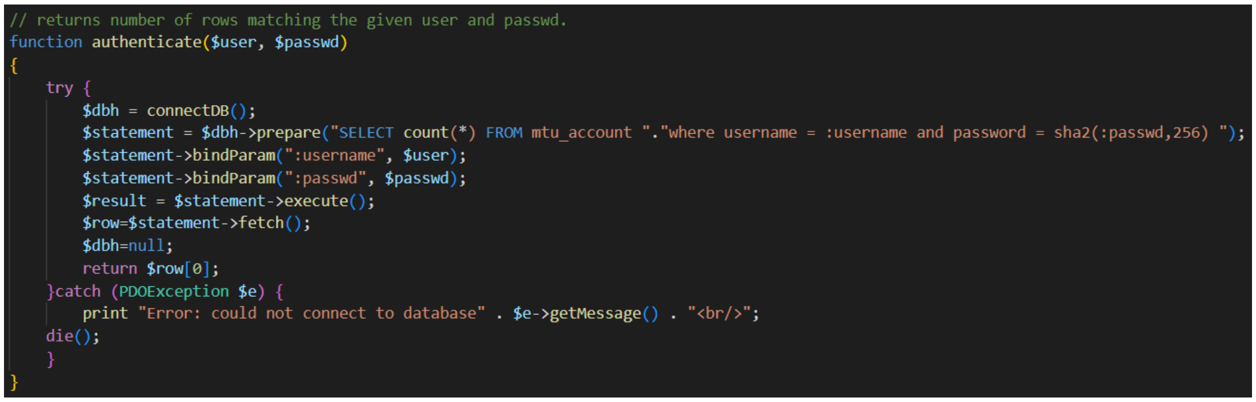
Encryption and Force Reset

In order to protect the user's password, I've used a process of encryption and decryption. When a user is created, the password they input to the database is encrypted and stored using a hashing algorithm, sha256.



```
create procedure create_student(
in in_name varchar(50),
in in_password varchar(50)
)
begin
    -- Generate random password --
    set @temp_password = sha2(in_password, 256);

    -- Insert new student --
    insert into student(name)
    values (in_name);

    set @student_id = last_insert_id();
    set @username = concat(in_name, @student_id);

    -- Create account --
    insert into mtu_account(username, password, account_type, student_id)
    values (@username, @temp_password, 'student', @student_id);
end//
```

The hashed password is stored in the database. When the user is retrieved, the password is decrypted and returned.



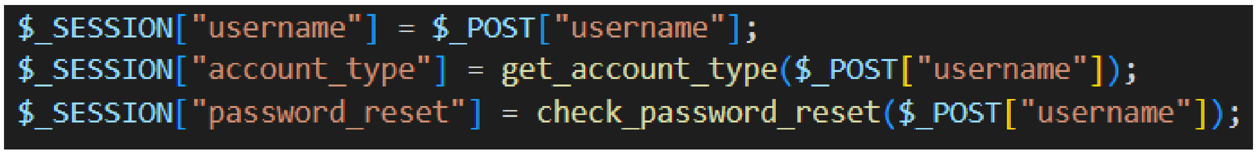
```
// returns number of rows matching the given user and passwd.
function authenticate($user, $passwd)
{
    try {
        $dbh = connectDB();
        $statement = $dbh->prepare("SELECT count(*) FROM mtu_account "."where username = :username and password = sha2(:passwd,256) ");
        $statement->bindParam(":username", $user);
        $statement->bindParam(":passwd", $passwd);
        $result = $statement->execute();
        $row=$statement->fetch();
        $dbh=null;
        return $row[0];
    }catch (PDOException $e) {
        print "Error: could not connect to database" . $e->getMessage() . "<br/>";
    die();
    }
}
```

I also force the user to reset their password, so someone who gains access to the temporary password, isn't able to log in to the user's account. (See login code p3-4)

Use of Session

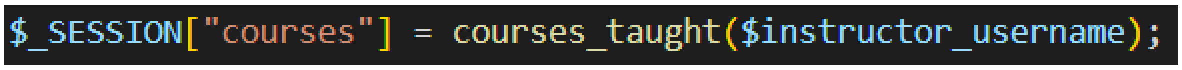
I use Session in order to store information about a user that is used in multiple pages. This allows me to carry information from page to page, without constantly re-declaring important variables. I've provided some examples below:

common.php:



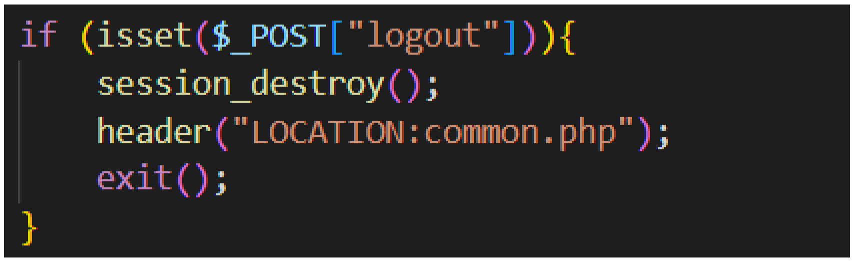
```
$_SESSION["username"] = $_POST["username"];
$_SESSION["account_type"] = get_account_type($_POST["username"]);
$_SESSION["password_reset"] = check_password_reset($_POST["username"]);
```

instructor.php:



```
$_SESSION["courses"] = courses_taught($instructor_username);
```

Once the user logs out, it's important that other users are not able to access their session information. Because of this, once a user logs out, their session information is destroyed, and redirecting the user to the login page.



```
if (isset($_POST["logout"])){
    session_destroy();
    header("LOCATION:common.php");
    exit();
}
```

Database Functions: (reference)

```php
<?php
    // connects to database
    function connectDB()
    {
        $config = parse_ini_file("/local/my_web_files/tpneal/classdb/database.ini");
        $dbh = new PDO($config['dsn'], $config['username'], $config['password']);
        $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $dbh;
    }

    // returns number of rows matching the given user and passwd.
    function authenticate($user, $passwd)
    {
        try {
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT count(*) FROM mtu_account "."where username =
:username and password = sha2(:passwd,256) ");
            $statement->bindParam(":username", $user);
            $statement->bindParam(":passwd", $passwd);
            $result = $statement->execute();
            $row=$statement->fetch();
            $dbh=null;
            return $row[0];
        }catch (PDOException $e) {
            print "Error: could not connect to database" . $e->getMessage() . "<br/>";
        die();
        }
    }

    // returns account_type associated with user
    function get_account_type($user)
    {
        try {
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT account_type FROM mtu_account WHERE username =
:username");
            $statement->bindParam(":username", $user);
            $result = $statement->execute();
            $row = $statement->fetch();
            $dbh = null;
            return $row['account_type'];
        } catch (PDOException $e) {
            print "Error: could not retrieve account_type" . $e->getMessage() . "<br/>";
            die();
        }
    }

    // returns account_type associated with user
    function get_student_id($username)
    {
        try {
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT student_id FROM mtu_account WHERE username =
:username");
            $statement->bindParam(":username", $username);
            $result = $statement->execute();
            $id = $statement->fetch();
            $dbh = null;
            return $id['student_id'];
        } catch (PDOException $e) {
            print "Error: could not retrieve student id" . $e->getMessage() . "<br/>";
            die();
        }
    }

    // returns if password has been reset
    function check_password_reset($user)
    {
        try {
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT password_reset FROM mtu_account WHERE username =
:username");
            $statement->bindParam(":username", $user);
            $result = $statement->execute();
            $row = $statement->fetch();
            $dbh = null;
            return $row['password_reset'];
        } catch (PDOException $e) {
            print "Error: could not retrieve password_reset" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function update_password($user, $new_password)
    {
        try {
            $dbh = connectDB();
            $statement = $dbh->prepare("UPDATE mtu_account SET password = sha2(:new_password, 256),
password_reset = 1 WHERE username = :username");
            $statement->bindParam(":username", $user);
            $statement->bindParam(":new_password", $new_password);
            $result = $statement->execute();
            $dbh = null;
```

```php
            return $result;
        } catch (PDOException $e) {
            print "Error: could not update password" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function courses_taught($instructor_username)
    {
        try{
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT course_id FROM teaches t JOIN mtu_account a ON
t.instructor_id = a.instructor_id WHERE a.username = :username");
            $statement->bindParam(":username", $instructor_username);
            $statement->execute();
            $courses = $statement->fetchAll();
            $dbh = null;
            return $courses;
        } catch (PDOException $e) {
            print "Error: could not retireve courses taught" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function enroll($student_username, $course_id)
    {
        try {
            $dbh = connectDB();
            $dbh->beginTransaction(); // Start the transaction

            // Prepare and execute the statement to retrieve student_id
            $id_stmt = $dbh->prepare("SELECT student_id FROM mtu_account WHERE username =
:username");
            $id_stmt->bindParam(":username", $student_username);
            $id_stmt->execute();
            $student_id = $id_stmt->fetch();

            // Prepare and execute the statement to enroll student
            $statement = $dbh->prepare("CALL enroll_student(:student_id, :course_id)");
            $statement->bindParam(":course_id", $course_id);
            $statement->bindParam(":student_id", $student_id["student_id"]);
            $statement->execute();
            $questions = $statement->fetchAll();

            $dbh->commit(); // Commit the transaction
            $dbh = null;
            return $questions;
        } catch (PDOException $e) {
            $dbh->rollBack(); // Rollback the transaction in case of an error
            print "Error: could not enroll in course" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function courses_enrolled($instructor_username)
    {
        try{
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT * FROM takes t JOIN mtu_account a ON t.student_id =
a.student_id JOIN course c ON c.course_id = t.course_id WHERE a.username = :username");
            $statement->bindParam(":username", $instructor_username);
            $statement->execute();
            $courses = $statement->fetchAll();
            $dbh = null;
            return $courses;
        } catch (PDOException $e) {
            print "Error: could not retireve courses enrolled" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function get_completion_time($student_id, $course_id)
    {
        try{
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT survey_completion_time FROM takes t JOIN mtu_account
a ON t.student_id = a.student_id WHERE t.student_id = :student_id AND t.course_id = :course_id");
            $statement->bindParam(":student_id", $student_id);
            $statement->bindParam(":course_id", $course_id);
            $statement->execute();
            $completion_time = $statement->fetch();
            $dbh = null;
            return $completion_time['survey_completion_time'];
        } catch (PDOException $e) {
            print "Error: could not retireve survey completion time" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function set_completion_time($student_id, $course_id)
    {
        try{
            $dbh = connectDB();
            $statement = $dbh->prepare("UPDATE takes SET survey_completion_time =
current_timestamp() WHERE student_id = :student_id AND course_id = :course_id;");
            $statement->bindParam(":student_id", $student_id);
            $statement->bindParam(":course_id", $course_id);
            $statement->execute();
            $dbh = null;
            return;
        } catch (PDOException $e) {
            print "Error: could not set survey completion time" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function get_department($course_id)
    {
        try{
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT dept_name FROM course WHERE course_id = :course_id");
            $statement->bindParam(":course_id", $course_id);
            $result = $statement->execute();
            $dept_name = $statement->fetch();
            $dbh = null;
            return $dept_name['dept_name'];
        } catch (PDOException $e) {
            print "Error: could not retrieve department name" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function get_available_courses($student_id)
    {
        try{
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT distinct course_id FROM takes WHERE course_id not in
(SELECT course_id FROM takes WHERE student_id = :student_id)");
            $statement->bindParam(":student_id", $student_id);
            $result = $statement->execute();
            $available_courses = $statement->fetchAll();
            $dbh = null;
            return $available_courses;
        } catch (PDOException $e) {
            print "Error: could not retrieve available courses" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function insert_response($survey_instance_id, $question_category, $question_display,
$choice_display, $dept_name, $course_id)
    {
        try{
            $dbh = connectDB();
            $key_check_off = $dbh->prepare("SET foreign_key_checks = 0");
            $key_check_off->execute();
            $statement = $dbh->prepare("INSERT INTO response (
                survey_instance_id,
                question_category,
                question_display,
                choice_display,
                dept_name,
                course_id
            ) VALUES (
                :survey_instance_id,
                :question_category,
                :question_display,
                :choice_display,
                :dept_name,
                :course_id
            );");
            $statement->bindParam(":survey_instance_id", $survey_instance_id);
            $statement->bindParam(":question_category", $question_category);
            $statement->bindParam(":question_display", $question_display);
            $statement->bindParam(":choice_display", $choice_display);
            $statement->bindParam(":course_id", $course_id);
            $statement->bindParam(":dept_name", $dept_name);
            $statement->execute();
            $key_check_on = $dbh->prepare("SET foreign_key_checks = 1");
            $key_check_on->execute();

            $dbh = null;
            return;
        } catch (PDOException $e) {
            print "Error: could not insert response" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function generate_survey_instance()
    {
        try{
            $dbh = connectDB();
            $dbh->beginTransaction(); // Start the transaction

            // GENERATE SURVEY NUMBER
            $statement = $dbh->prepare("SELECT (MAX(survey_instance_id)+1) as id FROM
survey_instance;");
            $statement->execute();
            $instance_id = $statement->fetch();

            // INSERT SURVEY INSTANCE INTO TABLE
```

```php
            $statement2 = $dbh->prepare("INSERT into survey_instance(survey_instance_id) VALUES
(:id)");
            $statement2->bindParam(":id", $instance_id["id"]);
            $statement2->execute();

            $dbh->commit(); // Commit the transaction
            $dbh = null;
            return $instance_id["id"];
        } catch (PDOException $e) {
            $dbh->rollBack(); // Rollback the transaction in case of an error
            print "Error: could not retireve survey instance" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function get_course_survey_questions($course_id)
    {
    try {
        $dbh = connectDB();
        $dbh->beginTransaction(); // Start the transaction

        // Prepare and execute the statement to retrieve dept_name
        $dept_stmt = $dbh->prepare("SELECT dept_name FROM course WHERE course_id = :course_id");
        $dept_stmt->bindParam(":course_id", $course_id);
        $dept_stmt->execute();
        $dept_row = $dept_stmt->fetch();
        $dept_name = $dept_row['dept_name'];

        // Prepare and execute the statement to retrieve course survey questions
        $statement = $dbh->prepare("SELECT * FROM question m WHERE (dept_name is null AND course_id
is null) OR dept_name = :dept_name OR course_id = :course_id");
        $statement->bindParam(":course_id", $course_id);
        $statement->bindParam(":dept_name", $dept_name);
        $statement->execute();
        $questions = $statement->fetchAll();

        $dbh->commit(); // Commit the transaction
        $dbh = null;
        return $questions;
        } catch (PDOException $e) {
            $dbh->rollBack(); // Rollback the transaction in case of an error
            print "Error: could not retrieve course survey questions" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function get_question_choices($question_category, $question_display, $dept_name, $course_id)
    {
        try {
            $dbh = connectDB();
            $statement = $dbh->prepare
            ("SELECT *
            FROM multiple_choice
            WHERE question_category = :question_category
                AND question_display = :question_display
                AND ((dept_name is null AND course_id is null) OR dept_name = :dept_name OR
course_id = :course_id)");
            $statement->bindParam(":question_category", $question_category);
            $statement->bindParam(":question_display", $question_display);
            $statement->bindParam(":course_id", $course_id);
            $statement->bindParam(":dept_name", $dept_name);
            $statement->execute();
            $choices = $statement->fetchAll();
            $dbh = null;
            return $choices;
        } catch (PDOException $e) {
            print "Error: could not retireve question choices" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function get_question_by_display($question_category, $question_display, $dept_name, $course_id)
{
        {
            try {
                $dbh = connectDB();
                $statement = $dbh->prepare
                ("SELECT *
                FROM question
                WHERE question_category = :question_category
                    AND question_display = :question_display
                    AND ((dept_name is null AND course_id is null) OR dept_name = :dept_name OR
course_id = :course_id)");
                $statement->bindParam(":question_category", $question_category);
                $statement->bindParam(":question_display", $question_display);
                $statement->bindParam(":course_id", $course_id);
                $statement->bindParam(":dept_name", $dept_name);
                $statement->execute();
                $question = $statement->fetch();
                $dbh = null;
                return $question;
            } catch (PDOException $e) {
                print "Error: could not retireve question" . $e->getMessage() . "<br/>";
                die();
            }
        }
    }

    function get_question_response_rates($question_category, $question_display, $dept_name,
$course_id)
    {
        try{
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT
                m.choice_text AS 'Response Option',
                COUNT(r.choice_display) AS 'Frequency',
                ROUND(COUNT(r.choice_display) / COALESCE((SELECT COUNT(*) FROM takes WHERE course_id
= :course_id), 1) * 100, 2) AS 'Percent'
            FROM
                multiple_choice m
            LEFT JOIN
                response r
                ON r.question_category = m.question_category
                AND r.question_display = m.question_display
                AND r.choice_display = m.choice_display
                AND r.dept_name = :dept_name
                AND r.course_id = :course_id
                AND r.survey_instance_id IS NOT NULL
            WHERE
                m.question_category = :question_category
                AND m.question_display = :question_display
                AND ((m.dept_name is null AND m.course_id is null) OR m.dept_name = :dept_name OR
m.course_id = :course_id)
            GROUP BY m.choice_text;");
            $statement->bindParam(":question_category", $question_category);
            $statement->bindParam(":question_display", $question_display);
            $statement->bindParam(":course_id", $course_id);
            $statement->bindParam(":dept_name", $dept_name);
            $statement->execute();
            $response_rates = $statement->fetchAll();
            $dbh = null;
            return $response_rates;
        } catch (PDOException $e) {
            print "Error: could not retireve question response rates" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function get_course_survey_ids($dept_name, $course_id)
    {
        try{
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT distinct survey_instance_id FROM response WHERE
dept_name = :dept_name AND course_id = :course_id;");
```

```
            $statement->bindParam(":course_id", $course_id);
            $statement->bindParam(":dept_name", $dept_name);
            $statement->execute();
            $survey_ids = $statement->fetchAll();
            $dbh = null;
            return $survey_ids;
        } catch (PDOException $e) {
            print "Error: could not retireve course survey ids" . $e->getMessage() . "<br/>";
            die();
        }
    }


    function get_completion_rate($course_id){
        try{
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT (SELECT count(*) FROM takes t JOIN mtu_account a ON
t.student_id = a.student_id WHERE course_id = :course_id AND survey_completion_time is not null) as
numerator,
            (SELECT count(*) FROM takes t JOIN mtu_account a ON t.student_id = a.student_id WHERE
course_id = :course_id) as denomenator");
            $statement->bindParam(":course_id", $course_id);
            $statement->execute();
            $completion_rate = $statement->fetch();
            $dbh = null;
            return $completion_rate;
        } catch (PDOException $e) {
            print "Error: could not retireve course completion rate" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function get_completed_survey_responses($survey_instance_id, $dept_name, $course_id)
    {
        try{
            $dbh = connectDB();
            $statement = $dbh->prepare("SELECT
            r.response_id, r.survey_instance_id, r.question_category,
            r.question_display, q.question_text, r.choice_display,
            m.choice_text, r.response_text, r.dept_name, r.course_id
            FROM question q
            LEFT JOIN
                multiple_choice m
                ON q.question_category = m.question_category
                AND q.question_display = m.question_display
                AND (q.course_id = m.course_id OR q.dept_name = m.dept_name OR (q.course_id is null
AND q.dept_name is null))
            LEFT JOIN
```

```
                response r
                ON r.question_category = m.question_category
                AND r.question_display = m.question_display
                AND r.choice_display = m.choice_display
                AND r.dept_name = :dept_name
                AND r.course_id = :course_id
                AND r.survey_instance_id IS NOT NULL
                AND (m.dept_name = :dept_name OR m.course_id = :course_id OR (m.dept_name is null
AND m.course_id is null))
            WHERE
                r.survey_instance_id = :survey_instance_id
            ORDER BY survey_instance_id, m.question_category desc");
            $statement->bindParam(":course_id", $course_id);
            $statement->bindParam(":dept_name", $dept_name);
            $statement->bindParam(":survey_instance_id", $survey_instance_id);
            $statement->execute();
            $responses = $statement->fetchAll();
            $dbh = null;
            return $responses;
        } catch (PDOException $e) {
            print "Error: could not retireve survey responses" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function add_question($category, $question_display, $text, $type, $dept_name, $course_id)
    {
        try {
            $dbh = connectDB();
            $statement = $dbh->prepare("CALL add_question(:category, :question_display, :text,
:type, :dept_name, :course_id)");
            $statement->bindParam(":category", $category);
            $statement->bindParam(":question_display", $question_display);
            $statement->bindParam(":text", $text);
            $statement->bindParam(":type", $type);
            $statement->bindParam(":dept_name", $dept_name);
            $statement->bindParam(":course_id", $course_id);
            $statement->execute();
            $dbh = null;
        } catch (PDOException $e) {
            print "Error: could not add question" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function add_question_choice($category, $question_display, $choice_display, $choice_text,
$dept_name, $course_id)
    {
        try {
            $dbh = connectDB();
            $statement = $dbh->prepare("CALL add_question_choice(:category, :question_display, 
:choice_display, :choice_text, :dept_name, :course_id)");
            $statement->bindParam(":category", $category);
            $statement->bindParam(":question_display", $question_display);
            $statement->bindParam(":choice_display", $choice_display);
            $statement->bindParam(":choice_text", $choice_text);
            $statement->bindParam(":dept_name", $dept_name);
            $statement->bindParam(":course_id", $course_id);
            $statement->execute();
            $dbh = null;
        } catch (PDOException $e) {
            print "Error: could not add question choice" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function delete_question($category, $question_display, $dept_name, $course_id)
    {
        try {
            $dbh = connectDB();
            $statement = $dbh->prepare("DELETE FROM question WHERE question_category = :category AND 
question_display = :question_display AND ((dept_name is null AND course_id is null) OR dept_name = 
:dept_name OR course_id = :course_id)");
            $statement->bindParam(":category", $category);
            $statement->bindParam(":question_display", $question_display);
            $statement->bindParam(":dept_name", $dept_name);
            $statement->bindParam(":course_id", $course_id);
            $statement->execute();
            $dbh = null;
        } catch (PDOException $e) {
            print "Error: could not delete question" . $e->getMessage() . "<br/>";
            die();
        }
    }

    function delete_question_choice($category, $question_display, $choice_display, $dept_name, 
$course_id)
    {
        try {
            $dbh = connectDB();
            $statement = $dbh->prepare("DELETE FROM multiple_choice WHERE question_category = 
:category AND question_display = :question_display AND choice_display = :choice_display AND 
((dept_name is null AND course_id is null) OR dept_name = :dept_name OR course_id = :course_id)");
            $statement->bindParam(":category", $category);
            $statement->bindParam(":question_display", $question_display);
            $statement->bindParam(":choice_display", $choice_display);
```

```php
            $statement->bindParam(":dept_name", $dept_name);
            $statement->bindParam(":course_id", $course_id);
            $statement->execute();
            $dbh = null;
        } catch (PDOException $e) {
            print "Error: could not delete question choice" . $e->getMessage() . "<br/>";
            die();
        }
    }
?>
```