Code Review of The Software Project:
Online-Exam-System : Study Outline

Project By:
Student ID: 180238
Student ID: 170238

Reviewed By:

Student ID: 210203

Introduction

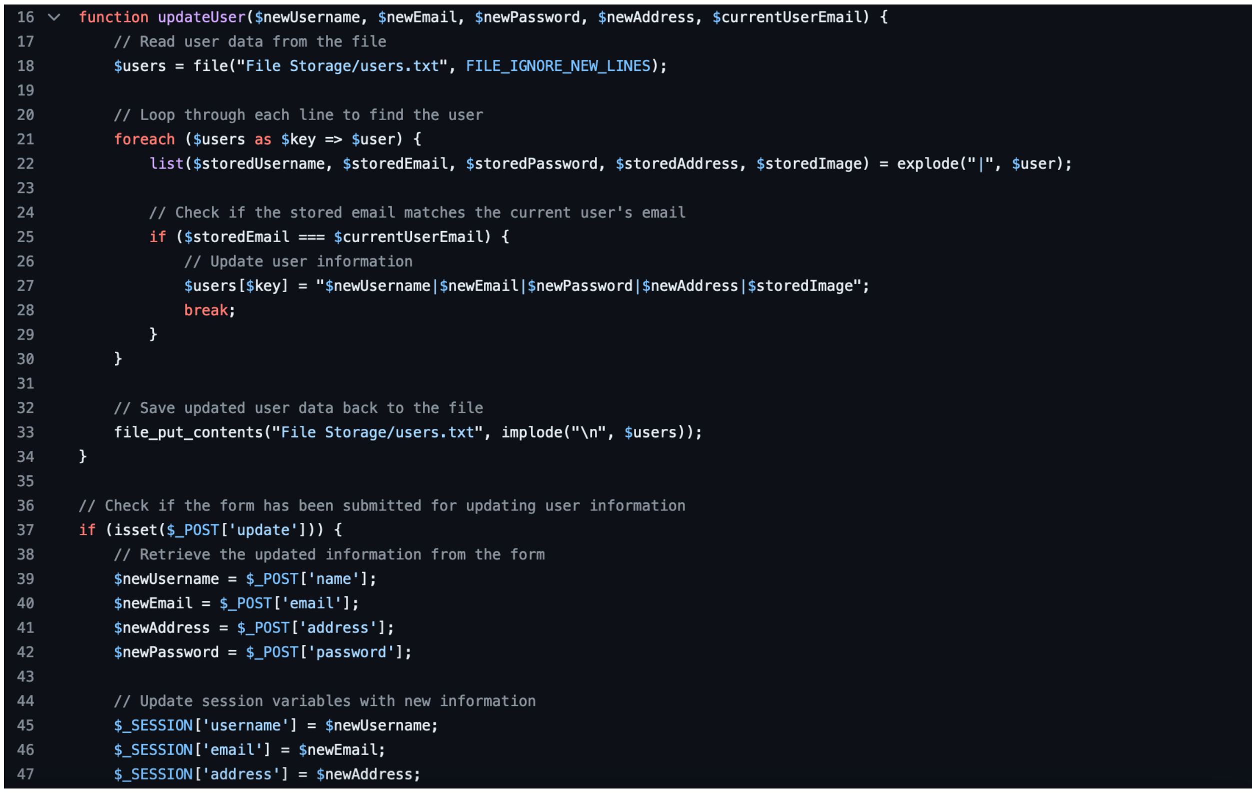
This code review evaluates the productivity application: **Study Outline**. The review identifies areas for improvement, adherence to best practices, and suggestions for enhancing maintainability, security, and overall code quality. In this code review, bad smells of the code, architecture evaluation, modularity check, condition statements of the code & other related sections are evaluated.

Code Smells

1. **Large or complex functions:**
    - Most of the .php files are containg more than. 80 lines of code. To remove this inconsistency break the files into multiple reuseable files such the inconsistency can be eradicated.
      Affected files are,
        - /Admin/ScheduleExam.php
        - /Admin/addclass.php
        - /Admin/addquestion.php
        - /Admin/addsubject.php
        - /Admin/editclass.php
        - /Admin/editexam.php
        - /Admin/editquestion.php
        - /Admin/editsubject.php
        - /Admin/home.php
        - /Admin/index.php
        - /Admin/logout.php
        - /Admin/result.php
        - /Admin/users.php
        - ScheduleExam.php
        - addclass.php
        - addquestion.php
        - addsubject.php
        - editclass.php
        - editexam.php
        - editquestion.php
        - editsubject.php
        - home.php

        - index.php
        - result.php
        - users.php

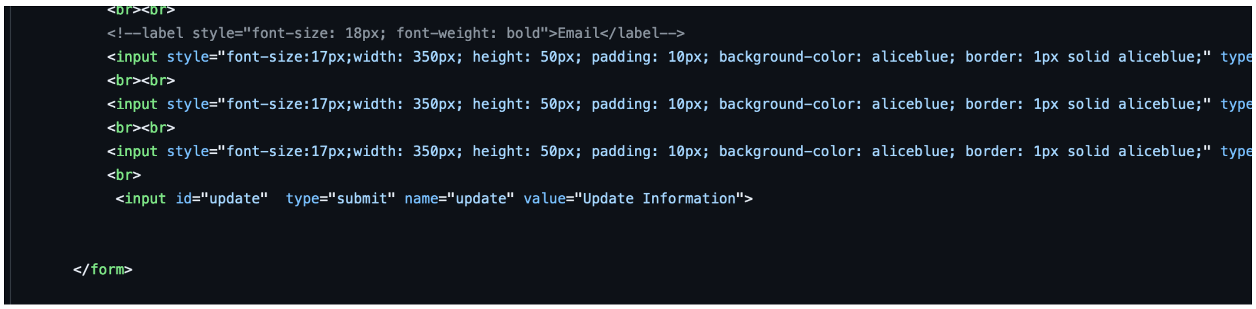
- In profile.php at updateUser() function, it is seen that the function contains more than 10 lines of code.



```php
function updateUser($newUsername, $newEmail, $newPassword, $newAddress, $currentUserEmail) {
    // Read user data from the file
    $users = file("File Storage/users.txt", FILE_IGNORE_NEW_LINES);

    // Loop through each line to find the user
    foreach ($users as $key => $user) {
        list($storedUsername, $storedEmail, $storedPassword, $storedAddress, $storedImage) = explode("|", $user);

        // Check if the stored email matches the current user's email
        if ($storedEmail === $currentUserEmail) {
            // Update user information
            $users[$key] = "$newUsername|$newEmail|$newPassword|$newAddress|$storedImage";
            break;
        }
    }

    // Save updated user data back to the file
    file_put_contents("File Storage/users.txt", implode("\n", $users));
}

// Check if the form has been submitted for updating user information
if (isset($_POST['update'])) {
    // Retrieve the updated information from the form
    $newUsername = $_POST['name'];
    $newEmail = $_POST['email'];
    $newAddress = $_POST['address'];
    $newPassword = $_POST['password'];

    // Update session variables with new information
    $_SESSION['username'] = $newUsername;
    $_SESSION['email'] = $newEmail;
    $_SESSION['address'] = $newAddress;
```

Here, for checking the form being updated, another new function can be created.

- Avoid using direct use of style into html attribute, Rather the files the it should be included into a css file. Such the code will become compact and lines of code will get reduced.



```html
            <br><br>
            <!--label style="font-size: 18px; font-weight: bold">Email</label-->
            <input style="font-size:17px;width: 350px; height: 50px; padding: 10px; background-color: aliceblue; border: 1px solid aliceblue;" type
            <br><br>
            <input style="font-size:17px;width: 350px; height: 50px; padding: 10px; background-color: aliceblue; border: 1px solid aliceblue;" type
            <br><br>
            <input style="font-size:17px;width: 350px; height: 50px; padding: 10px; background-color: aliceblue; border: 1px solid aliceblue;" type
            <br>
             <input id="update"  type="submit" name="update" value="Update Information">


        </form>
```

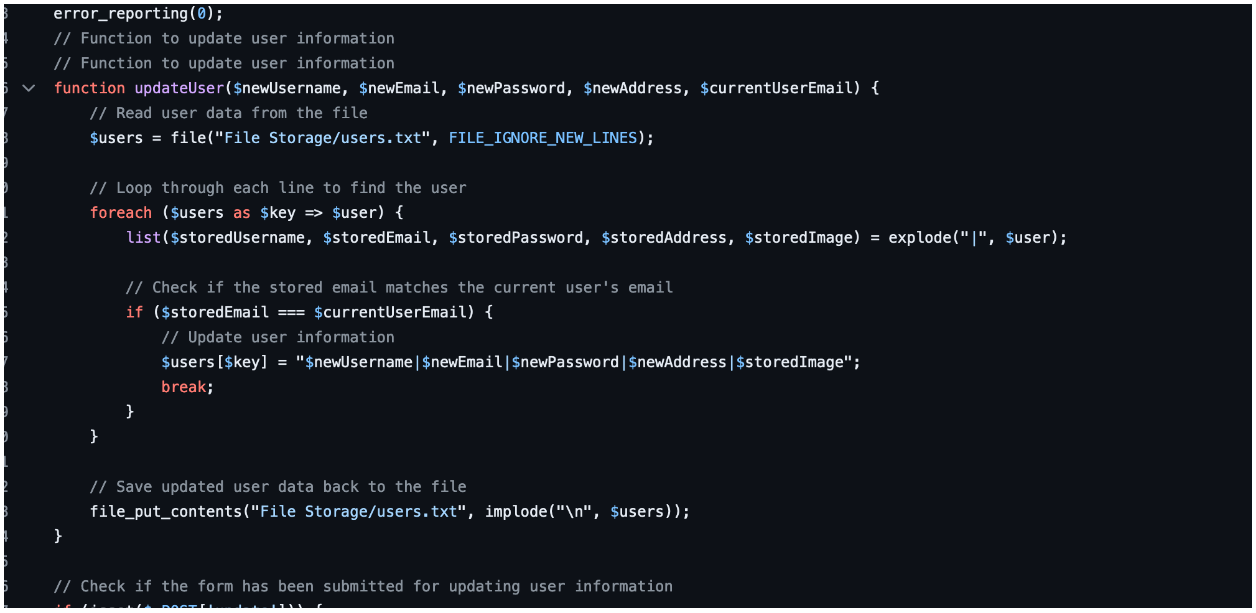
2. Long parameter lists:

Long parameter has been found in profile.php at updateUser(), can be passed as a bundle of those paratemers such an object of User that will contain the data of ($newUsername, $newEmail, $newPassword, $newAddress, $currentUserEmai).

3. Excessive comments/Bad Comments:

In some modules there are excessive comments and some doesn't have any comment at all.

Compose comments convergently for a block of code, not for line of code.



```php
error_reporting(0);
// Function to update user information
// Function to update user information
function updateUser($newUsername, $newEmail, $newPassword, $newAddress, $currentUserEmail) {
    // Read user data from the file
    $users = file("File Storage/users.txt", FILE_IGNORE_NEW_LINES);

    // Loop through each line to find the user
    foreach ($users as $key => $user) {
        list($storedUsername, $storedEmail, $storedPassword, $storedAddress, $storedImage) = explode("|", $user);

        // Check if the stored email matches the current user's email
        if ($storedEmail === $currentUserEmail) {
            // Update user information
            $users[$key] = "$newUsername|$newEmail|$newPassword|$newAddress|$storedImage";
            break;
        }
    }

    // Save updated user data back to the file
    file_put_contents("File Storage/users.txt", implode("\n", $users));
}

// Check if the form has been submitted for updating user information
```

Avoid these types of line commenting in the following files. Rather compose a block of comment for the function after being defined. Include what the the total function or the block of code does, not how it works.

/Admin/ScheduleExam.php
/Admin/addclass.php
/Admin/addquestion.php

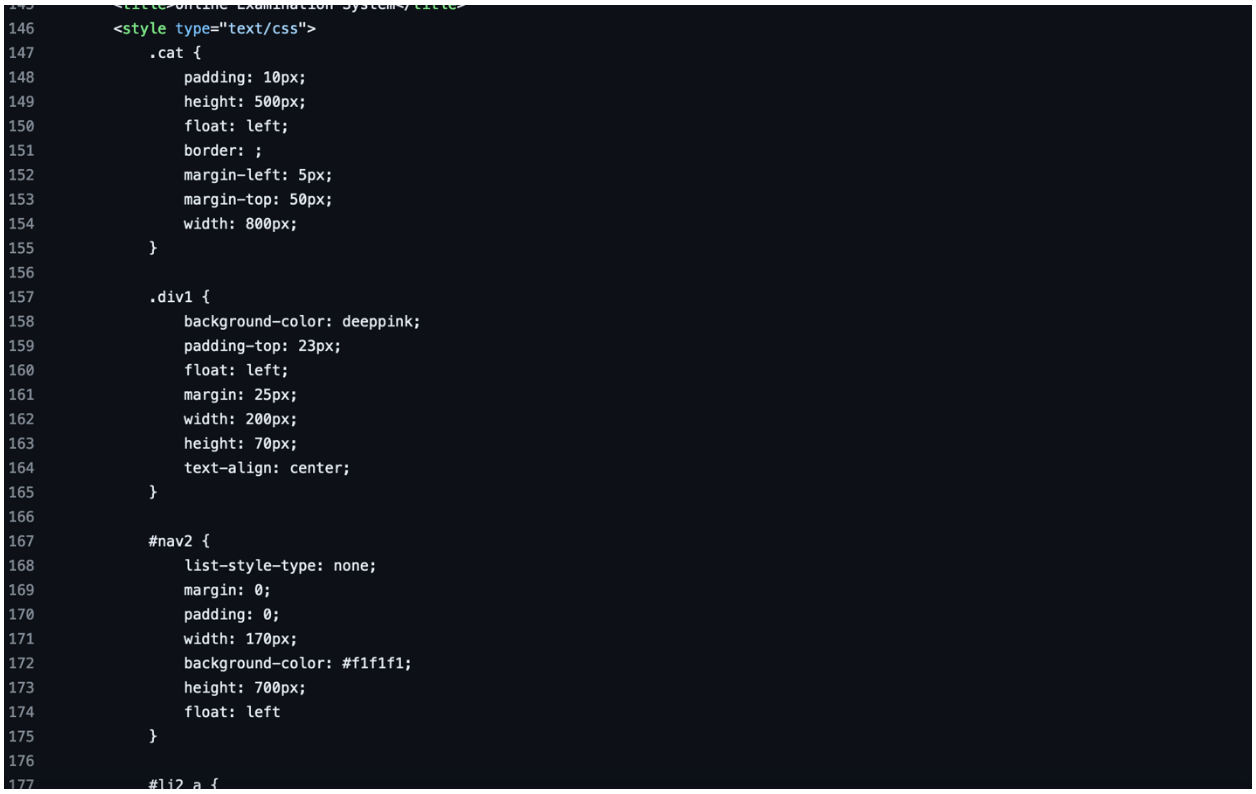
/Admin/addsubject.php
/Admin/editclass.php
/Admin/editquestion.php
/Admin/editsubject.php
/Admin/login_process.php
/Admin/users.php
login_process.php
profile.php
profileresult.php
signup_process.php
mostly all across the files.

4. **Duplicate code:**
   Most of the php files are containing duplicate code.
   Such are,

   - /Admin/ScheduleExam.php
   - /Admin/addclass.php
   - /Admin/addquestion.php
   - /Admin/addsubject.php
   - /Admin/editclass.php
   - /Admin/editexam.php
   - /Admin/editquestion.php
   - /Admin/editsubject.php
   - /Admin/home.php
   - /Admin/index.php
   - /Admin/logout.php
   - /Admin/result.php
   - /Admin/users.php
   - ScheduleExam.php
   - addclass.php
   - addquestion.php
   - addsubject.php
   - editclass.php
   - editexam.php
   - editquestion.php
   - editsubject.php
   - home.php
   - index.php
   - login_process.php

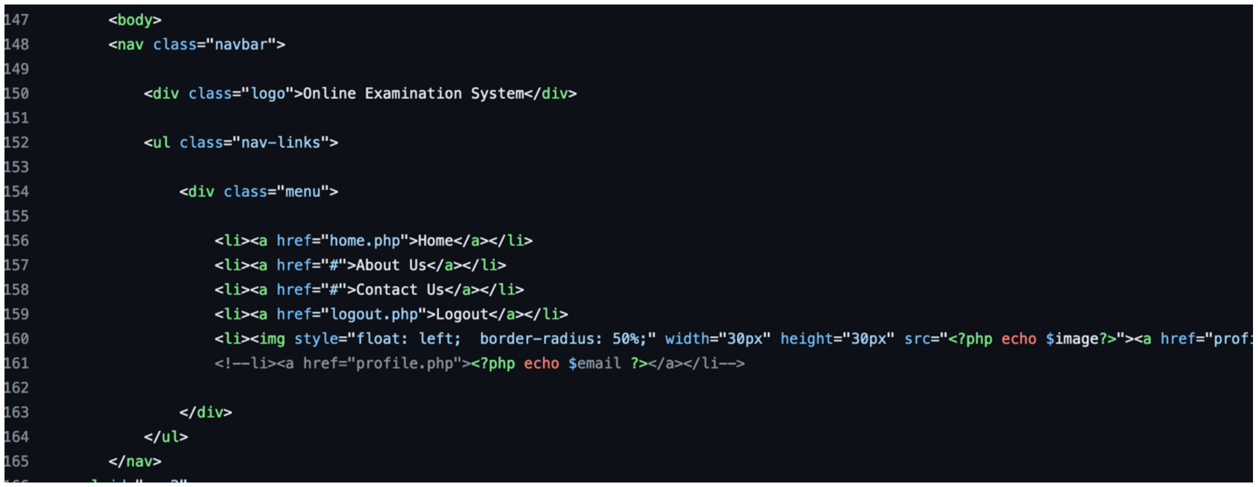
- logout.php
- result.php
- users.php

There exist a block of CSS code that repeats all across the .php files. A new css file should be created to remove the redundancy.



```
    <title>Online Examination System</title>
    <style type="text/css">
        .cat {
            padding: 10px;
            height: 500px;
            float: left;
            border: ;
            margin-left: 5px;
            margin-top: 50px;
            width: 800px;
        }

        .div1 {
            background-color: deeppink;
            padding-top: 23px;
            float: left;
            margin: 25px;
            width: 200px;
            height: 70px;
            text-align: center;
        }

        #nav2 {
            list-style-type: none;
            margin: 0;
            padding: 0;
            width: 170px;
            background-color: #f1f1f1;
            height: 700px;
            float: left
        }

        #li2 a {
```

At most of the view files, code of the navigation bar is repeated several times. Create new file as navigation_bar.php and include the file all acorss the files where the navigation bar is required.



```html
<body>
<nav class="navbar">

    <div class="logo">Online Examination System</div>

    <ul class="nav-links">

        <div class="menu">

            <li><a href="home.php">Home</a></li>
            <li><a href="#">About Us</a></li>
            <li><a href="#">Contact Us</a></li>
            <li><a href="logout.php">Logout</a></li>
            <li><img style="float: left;  border-radius: 50%;" width="30px" height="30px" src="<?php echo $image?>"><a href="prof
            <!--li><a href="profile.php"><?php echo $email ?></a></li-->

        </div>
    </ul>
</nav>
```

5. **Inconsistent naming conventions:**
   In most cases, naming conventions are standardized.

6. **Incomplete error handling:**
   Errors are mostly handled in each of the modules. There are mainly scope of running into error while doing file operations which seems to be handled carefully.

7. **Too many if/else statements:**
   There are no significance of redundant usage of if/else statements.
8. **Poor use of inheritance:**
   There is no poor use of inheritance in this project which is particularly causing any problem.

9. **Unnecessary dependencies:**
   No signs of unnecessary library usage.

10. **Magic numbers or hard-coded values:**
   In most of the cases, there is no sign of hard coding or magic numbers. Instead, contents are introduced and used in the project.

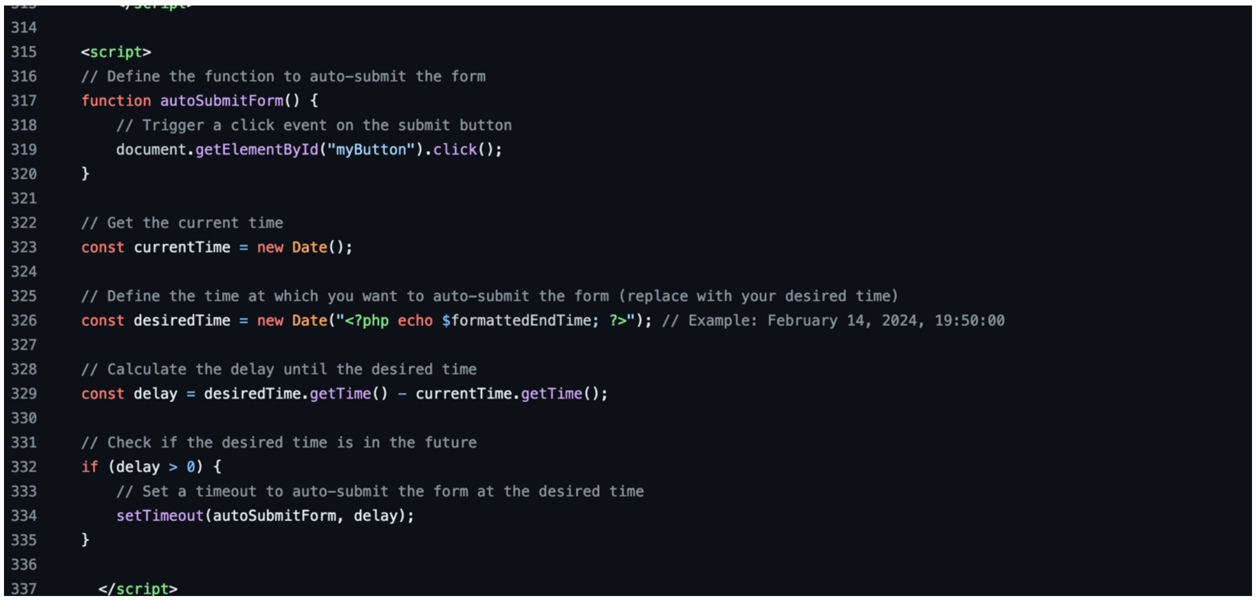
Magic numbers are present in question.php at function countdownTimer(), replace the literal values by meaningfull constant variables.

Architectural Evaluation

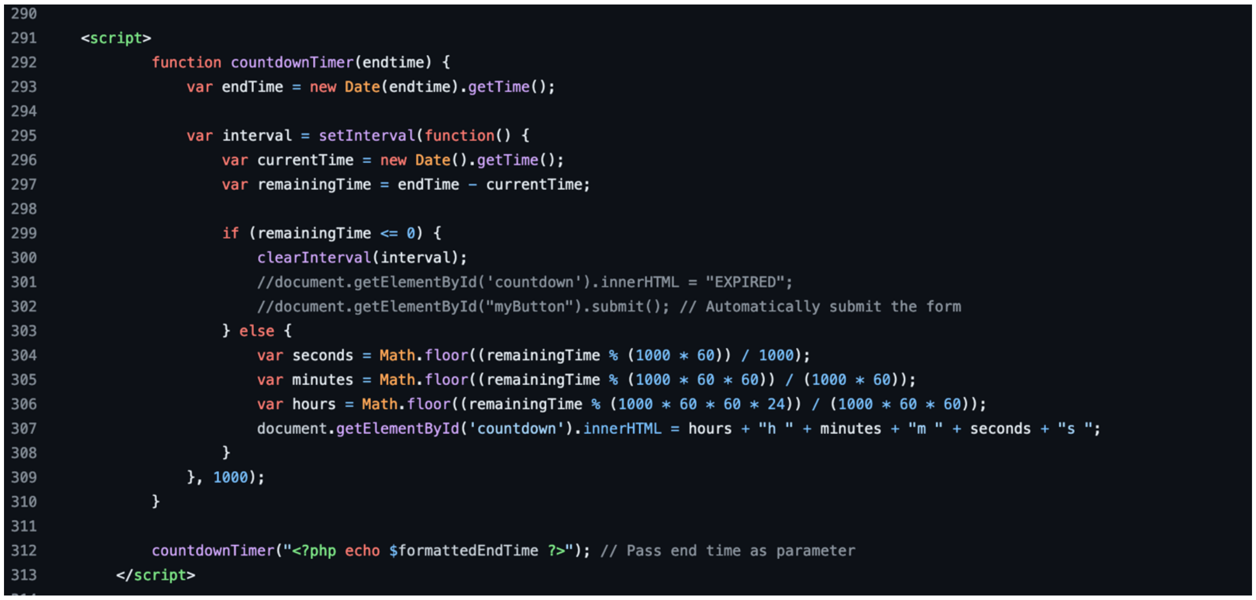
Although there exists modularity in the project but doesn’t satisfy the MVC pattern. Usually, web applications follow Model-View-Controller Architecture, In the code base no signs of the architecture are found.
The modular components should be rearranged by following proper file-folder structure.
Data Models should be created properly like User.php, Question.php, Subject.php
Views components like signup.php, contact.php, login.php, home.php, index.php, should be kept at specific views folder.
And proper controller should be created for the accordance model.

Modularity Check

The Project is not properly modular, need a little bit brush up in arranging the files in accordance with their roles, as Model, View, and Controller.
In the file question.php the code under <script> tag should be modularized by a javascript file.

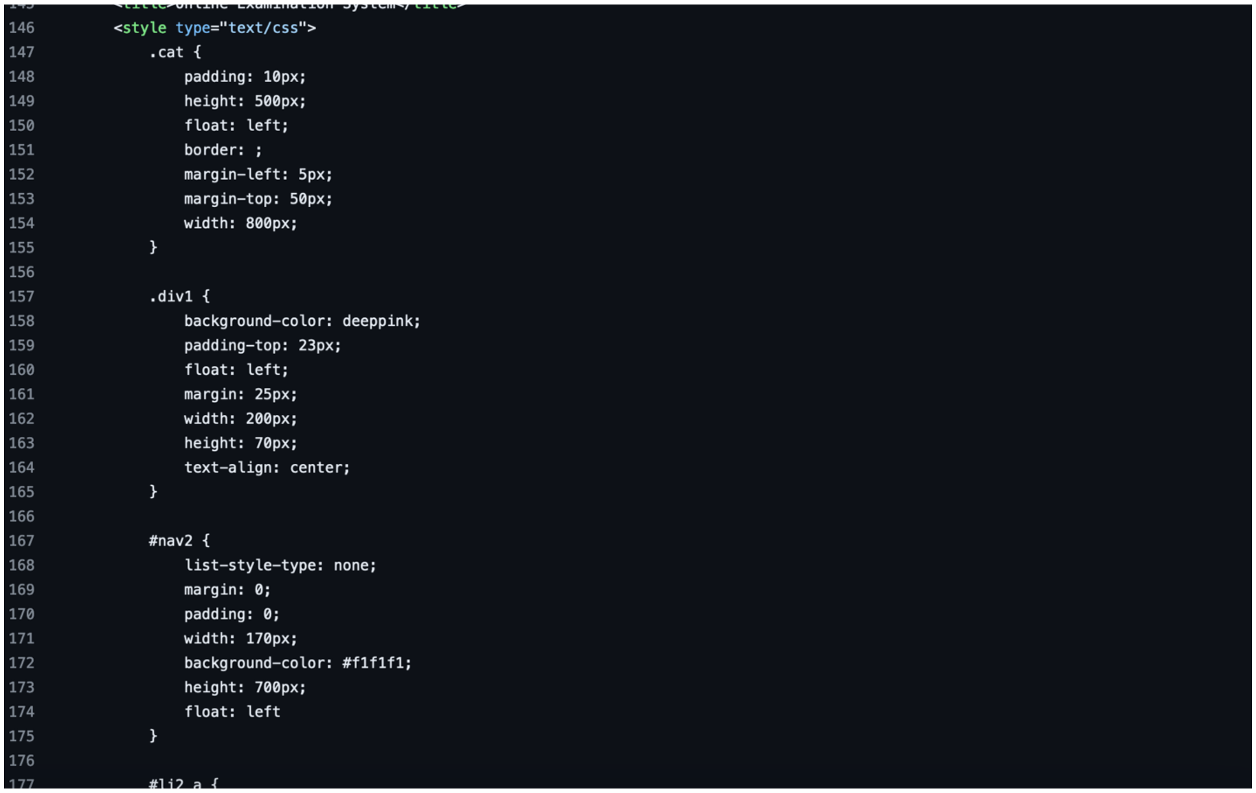


```
<script>
// Define the function to auto-submit the form
function autoSubmitForm() {
    // Trigger a click event on the submit button
    document.getElementById("myButton").click();
}

// Get the current time
const currentTime = new Date();

// Define the time at which you want to auto-submit the form (replace with your desired time)
const desiredTime = new Date("<?php echo $formattedEndTime; ?>"); // Example: February 14, 2024, 19:50:00

// Calculate the delay until the desired time
const delay = desiredTime.getTime() - currentTime.getTime();

// Check if the desired time is in the future
if (delay > 0) {
    // Set a timeout to auto-submit the form at the desired time
    setTimeout(autoSubmitForm, delay);
}

  </script>
```



```
<script>
        function countdownTimer(endtime) {
            var endTime = new Date(endtime).getTime();

            var interval = setInterval(function() {
                var currentTime = new Date().getTime();
                var remainingTime = endTime - currentTime;

                if (remainingTime <= 0) {
                    clearInterval(interval);
                    //document.getElementById('countdown').innerHTML = "EXPIRED";
                    //document.getElementById("myButton").submit(); // Automatically submit the form
                } else {
                    var seconds = Math.floor((remainingTime % (1000 * 60)) / 1000);
                    var minutes = Math.floor((remainingTime % (1000 * 60 * 60)) / (1000 * 60));
                    var hours = Math.floor((remainingTime % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
                    document.getElementById('countdown').innerHTML = hours + "h " + minutes + "m " + seconds + "s ";
                }
            }, 1000);
        }

        countdownTimer("<?php echo $formattedEndTime ?>"); // Pass end time as parameter
    </script>
```

Some a new css file need to be created to avoid the redundancy of styles.



```
        <title>Online Examination System</title>
        <style type="text/css">
            .cat {
                padding: 10px;
                height: 500px;
                float: left;
                border: ;
                margin-left: 5px;
                margin-top: 50px;
                width: 800px;
            }

            .div1 {
                background-color: deeppink;
                padding-top: 23px;
                float: left;
                margin: 25px;
                width: 200px;
                height: 70px;
                text-align: center;
            }

            #nav2 {
                list-style-type: none;
                margin: 0;
                padding: 0;
                width: 170px;
                background-color: #f1f1f1;
                height: 700px;
                float: left
            }

            #li2 a {
```

Suggestions,

- Create header.php
- Create navigation_bar.php
- Create style.css
- Break Long functions into smaller ones.

If/else Condition to Switch statement

There doesn’t exist any heavy usage of if/else condition. So, the introduction of switch case statement is not required.