Quiz 2

Same instructions as quiz 1: read everything twice before answering questions.

When you are finished, you are free to leave. Quiz is open notes, open Internet. Only things you can't do are Generative AI of any kind, run Live Share, talk to each other (or any other students), and post the questions on StackExchange and the like.

You must sign the sign-in sheet at the front of the room. No name on sign-in sheet = 0 on quiz.

Good luck!

Part 1

1.1. Explain three possible features of a web application that require (or, at least, made easier by) a server-side component written in a language such as PHP. Don't just mention the feature, explain in detail what it involves.

1. Sanitization is much simpler with PHP due to its in-built implementation in contrast to a user-defined implementation. PHP has the filter_input() function that based on the filter parameter will do basic sanitization (character escaping, removing characters, etc all based on filter parameters.

2. Connection to databases like SQL using PDOs where you enter your login information to said database, the database name, and a given table that you want to extract data from

3. User authentication and authorization also have built-in methods to perform authentication and authorization. Since PHP can connect the front-end application to the database using the aforementioned PDO and then check inputted credentials against those within the connected database and tables using queries.

1.2. Explain <u>two actions</u> that can be taken to secure a web application. These may be related to user-authentication & authorization, server configuration, codebase, and/or network infrastructure. Don't just mention the feature, explain in detail what it involves.

1. When dealing with user data, and as a result user logins, it is critical that passwords are hashed using, in our case PHP, a hashing function/implementation. Additionally, utilizing salts can also be huge in securing a user's information by adding additional characters to mislead and add to a cracker's attempt to get user info.

2. As I've stated before, sanitization is an important security measure for web applications due to websites' implicit vulnerabilities to attacks like injections and Cross-site scripting. Sanitization prevents most of the previously mentioned attacks by escaping, removing, ignoring, stripping, or adding characters, resulting in nullified attempts to get vulnerable data.

Part 2

Explain this code segment in two different ways: first, explain the overall picture without using any technical jargon, as if you were explaining the code to someone who doesn't understand any programming, and; second, explain in as exacting detail as possible, line by line, what the code is doing. If there are any mistakes or errors in the code, fix them inline using a different color. If you come up to me to tell me there are mistakes, -5 points.

```
if (isset($_GET['lname'])) {
     if ($_GET['lname'] != '') {
          $pstmt = $conn->prepare('SELECT * from customers WHERE
lname = :ln');
          $pstmt->bindParam('ln', $_GET['lname'], PDO::PARAM_STR);
     } else {
          echo "lname not given, outputting entire file";
          $pstmt = $conn->prepare('SELECT * from customers');
     }

          $pstmt->bindParam('ln', $_GET['lname'], PDO::PARAM_STR);
     //Added the bind statement after either was created so either
     statement can properly be executed
     $pstmt->execute();
     while ($row = $pstmt->fetch()) {
          printf("%s %s",$row['fname'],$row['lname']);
     }}
```

Simple Explanation:
The text/code seen allows us to view the first names and last names that were stored by the website, if and only if, a connection between our code and the website is made and there are last names stored in our website. Meaning we were storing users' input of their last name on the website. Our code then just shows us first and last names in the website(in sequential, linked order).

Technical Explanation:

The code above is first checking that a connection between our database (presumably MySQL) and PHP has been established–possibly by PDO. If said connection is underway, we check that there is a last name column. All valid entries are then selected using a stored query. However, if there is no last name column, we just prepare to dump all the other information on a customer (I believe just a first name in this case). Finally, based on the stored selection query, we go row by row in the database and output their first and last name (if applicable). We also used prepare, execute method for safer data extraction from our database. Note: I forgot to mention, but if an entry is empty the output skips them.

Part 3
Here is a Google Drive folder containing all the lectures for my Modern Binary Exploitation course:
https://drive.google.com/drive/folders/1rjc__npAFJn2oyz-1QpNpyH2qqHankwQ?usp=sharing

Extend your (or create a) mini-LMS application to include MBE. You should create a JSON object that represents the lectures. Each lecture should be represented with a Title, Description, and Link. Since I know (most of) you have not (yet) taken MBE, you may use what you see on the opening slide for each lecture as its Description.

When the user ~~logs in~~ to the mini-LMS, they should be able to select which course to display. The left-hand side should be dynamically generated from the appropriate JSON object. Clicking on an item on the left-hand side should populate the preview window containing the Title, Description, and Link for that particular item.

Add a button that, when clicked, switches from one course to the other. Clicking this button should destroy and dynamically regenerate the left-hand side with the other JSON object.

Make sure your archive button is able to archive both the MBE and Web Systems courses.

Creativity counts for this! Don't just stop once this works. Showcase all your talents in HTML, CSS, Javascript, PHP, and MySQL.

README.md Don't forget a readme! Briefly explain your solution and any issues you faced. Tell us what you'd like to have considered for creativity. Don't forget to put your citations!
No citations = 0 on quiz (that's plagiarism!)

Submission

- Put everything into a quiz2 folder on your personal GitHub repo
- Part 3 must be hosted on your VM at https://[FQDN]/[your-repo]/quiz2

Rubric

Part 1	15 Points
Part 2	15 Points
Part 3:	
JSON object	10 Points
HTML/CSS/JS/PHP/MySQL	40 Points
Creativity	10 Points
README.md	10 Points
Total	**100 Points**

Extra Credit (+5 points)
What are the lyrics to the Alma Mater of RPI. No typos. All or nothing.

Here's to old RPI, her fame may never die.
Here's to old Rensselaer, she stands today without a peer.
Here's to those olden days,
Here's to those golden days,
Here's to the friends we made at dear old RPI.

Extra Credit 2 (+1 point to your final grade at the end of the semester)
Get **everyone** in the class to sing the Alma Mater at the same time on November 28. All or nothing. If even one person doesn't sing, no extra credit.