Quiz 2

Same instructions as quiz 1: read everything carefully, then read everything carefully again, then read everything carefully a third time, then start answering questions.

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

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) One possible feature that a web application could require a back end for is a login or user system. There are many ways to do this but if you are creating a manual login system you will need a few things. First, you're going to need a database to store all of the users that are registered on the platform. This can be done on any database but for this example, we will use mySQL and phpmyadmin. We need to create a table in our database called users (anything you want), with the fields, ID, Username, Password, Salt all VarChar except ID which should autoincrement integers. Then in out php file we will connect to the database and have 2 options, Login and Register. If the user already has an account a command will be run, SELECT EXISTS(SELECT * FROM users WHERE UsernameandPassword); which will find if the user is registered and with the right credentials. If the user is not existing the register button will be clicked and with the given data an INSERT command will be relayed to the database and the row will be added.
2) The next feature that could be needed is reading data from a database. This can not be done without a back end such as php but with a back end, it is quite simple. First, we will create a database will as many tables as you desire. Then we will write the connection code needed to gain access to the database in our php file. This includes the database name, username, password, and server name. Next, we call the MySQLI command which will try and connect you, if it is successful you can now query whatever you want. To get the data you can go table by table with the command, SELECT * FROM TableName. With this, you can run the command for as many tables as needed and you will acquire as many arrays filled with the data as called. From

this, you can do whatever is needed to display this information with echo commands.

3) One last feature that a web app could need is API tools. Not everything that you want to include in your web app can be coded from scratch in an efficient manner. Because of this, we have APIs, which let you use other people's code to implement features on your application. There are thousands of different APIs that can do almost anything that you need. The majority of them are based in JS so you need to have a back-end JS structure to make them work. You start by finding the API that is needed for the job. Then there are a few ways to integrate it into your app, first is a simple API key that gets the data needed without any authentication necessary. Next is a simple auth system which required you to get a key from the developer and supply that in your API call. Lastly is an auth method called OAuth which requires a code that expires after a certain amount of time. For these, you must either manually refresh the token or write a function that makes the refresh call for you. After you figure out the authentication method of the API we can call it and get the data. Now, this is going to be different for every API but one general way that I have called them in the past is a XMLHttpRequest, and the command for that is:

   ```
   Let request = new XMLHttpRequest;

   Request.open(“GET”, “APIURL”);

   request.send();
   ```

   After the commands are called you will most likely get a json file in return. This can be parsed by using the JSON.parse(requestArray); command and all the data needed is there for use.

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) The first way you could go about securing your web app is by using prepare and execute statements with your sql queries. You first create the prepare statement by running: $conn->prepare("SELECT COUNT(id) FROM table WHERE column = :var");. This is just an example but the point is to not directly input any variables that the user has input any data into, into the query call. Then you want to write: $connectToDB->execute(['title' => $title, 'description' => $description, 'link' => $link]); with the variables that you want to input. This makes sure that the users can’t run any commands through their input on the back-end and secures your application from attacks such as an SQL injection.

2) The second way you can secure your web application is by adding salts to your passwords. The use of adding salts increases the security of the stored passwords of your users by adding a random string at the front or end of the password, then hashing it. If we do this then not only will the password hash be stronger but then there will never be 2 of the same hashes stored. To do this, we first have to look at the database with the users being stored. Then we will add a salt column and randomly generate salts for each user. After that, we will hash the salt and the password together with whatever language you are using, because it will be different in each, and store that new hash instead. Now, when your user tries to log in you will decrypt their password and check it by removing the salt. Then once it is checked the password is encrypted and stored in the database as before.

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 (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');
      }
      $pstmt->execute();
      while ($row = $pstmt->fetch()) {
            printf("%s %s",$row['fname'],$row['lname']);
      }
}
```

General: The code will first check to see that the name field was filled out. If it was then it will make sure that the data input is not empty. Then if there is data available it will create a call to the database to find the data where the lname value is the same as the one from the input. On the other hand, if the lname field is empty then the code will report back telling you the input was empty and then it will make a call to get all of the data instead of just the matching data. After it gets the data the code will go and print it all out in a format of Firstname Lastname.

Detailed: When the php script is run, the code will check for a change in the Last name input field and trigger the isset function for the lname id. Then the code checks whether the data that was input is empty or not. If it is not, it will prepare a statement to query, the query is getting all of the data from the customers table that has the same last name as the one given in the lname field. But for safety, it is without the variable. This is to make sure that there is no way that a user could input a query string and get access to the database without authorization. Then there is a bind call which makes sure that the value is the same type as what we have assigned to it, in this case, a PARAM_STR or string. If it is it will input the variable into the query. Now looking at if the string is empty, first there will be a message that says it is empty and will show the whole file. Then a query is set up to get all the rows from that table. After the query calls are set up, either for a lname value or an empty value, the query string is executed. After execution, the code loops through the returned array and prints out all of the records in a “FirstName LastName” format.

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.

If you have a JSON object for Web Systems, when the user logs into 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. If you don't have a JSON object for Web Systems, dynamically generate the left-hand side from the MBE JSON object; you still need to populate the preview window based on which item is clicked.

If you have a JSON object for Web Systems, 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. If you don't have a JSON object for Web Systems, add a button that, when clicked, destroys the left-hand side's content and dynamically regenerates it from the JSON object.

Make sure your archive button is able to archive both the MBE and (if applicable) 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 2 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.