## **Quiz 2**

Same caveats as quiz 1: read everything really carefully, then read everything really carefully again, then read everything really 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 talk to each other and post the questions on StackExchange and the like.

#### **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 what it involves.
* **1. It could possibly prevent user device problem such as slow loading, and high CPU usage because client-side scripting requires user computers to compile the information from the client-end.**
* **2. It could run dynamic pages on the multiple browsers. For example, since php checks the type of variables at runtime, it could be used to run dynamic web system.**
* **3. Better security. Since the scripting takes on the server and the script itself shouldn’t e sent to the browser, it could help with increasing security level.**
* **1.2**. **Explain two actions** that can be taken to **secure** a web application. These may be related to user-authentication & authorization, server configuration, codebase, and/or network infrastructure.
* 1. Use good Password random hash functions, which is an one-way transformation that is only able to get from the plaintext end, not from the hash-end.
* 2. Have multi factor authentication. Since it requires multiple weighted factors, it reduces the risk of compromised/weak passwords.

#### 

#### **Part 2**

Explain each 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.

**2.1**.  
 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']);

}  
 }

1. This is the way of authenticating(PDO) the user to connect database. If the program successfully finds the elements from the database with specified scope, the system would fetch the queried data and print out first name and lastname in customized format.
2. First, it checks if there is a tag ‘lname’ and if there is, it should set and will take an action. Then it checks if lname is not an empty element. If so, the prepared statement(ptstmt) should be set to all of the prepared value connected to the database/table ‘customers’ where lname is equivalent to the specified value.

**2.2**.  
 $('#trigger').click(function(e) {

$.getJSON('people.json', function(data) {  
 $.each(data.people, function(key, val) {  
 alert(val.name + ", " + val.profession);

});  
 });

#### 

#### If there is a clickable object named trigger, user could click the object and expect the program to fetch data from ‘people.json’ and get alerts of every member’s name and profession on page.

#### If the portion selected as an id ‘trigger’ is clicked, it should run the current function. Then it gets json data and loop over data selected by ’people’. Then it assigns each key and value to the elements and alert(display) it’s current name and profession within specified format as the user placed ‘, ’ in the middle.

#### **Part 3**

**3.1**. In MySQL, create a database named websys\_quiz with the following 2 tables and link the 2 tables using a foreign key constraint:  
items

-id int(11) primary key

-name varchar(255)

-price decimal(6,2)

discounts

-id int primary key

-item\_id int(11)

-discount decimal(3,2)

Run the following code in the SQL tab to insert test data. As in Part 2, if this code is wrong, fix it before running it and document your fix(es) in your README.md file.  
INSERT INTO `items` (`id`, `name`, `price`) VALUES (1, 'MacBook Pro', '2499'),

(2, 'OpenBSD Tshirt, '20.0'),(3, 'Amazon echo', '99.99'),(4, 'Nvidia GTX 3080', '1999.99'),(5, 'AMD Ryzen 9 3900X’, '549.99');  
INSERT INTO `discounts` (`id`, `item\_id`, `discount`) VALUES (1, 1, 0.25), (2, 2, 0.5),(3, 3, 0.75),(4, 5, 0.1);

**3.2**. Create a PHP page with 3 buttons that connects to the previously created database and performs each of the following queries based on the button pressed:

* + 1) List **all** items ordered by price from lowest to highest, **before applying discount**.
  + 2) List **all** items ordered by price from lowest to highest, **after applying discount**.
  + 3) Find the average price of **only** the items that have a discount, **after applying discount**.

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. Don’t forget to include any MySQL fixes you needed at the start of Part 3.

#### 

#### **Submission**

* Create a **new branch** named **quiz2** in your personal repo
* Put all quiz materials in that branch
* **DO NOT MERGE INTO MAIN**
* **-5 points for each submission step not followed**

#### **Rubric**

* **Part 1** 20 Points
* **Part 2** 20 Points
* **Part 3:**
* **Database** 15 Points
* **PHP+queries** 25 Points
* **Creativity** 10 Points
* **readme** 10 Points

**Extra Credit (+5 points)**

Name the chat protocol developed at RPI in the 1990s.