## **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.
* One possible feature is the storing of data like a MySQL database. You could use JavaScript and HTML create a table that could store all the data that you needed. However, the code could become complicated and expensive whereas with PHP and MySQL, storing and accessing data is extremely efficient and easy.

A second possible feature is the updating of front-end code depending on what buttons a user clicks such as changing the menu structure of an HTML document. Once again, JavaScript could be used to update the code on an HTML page, but it can be quickly resource dependent and slow down the website. A solution to this is the use of AJAX which works asynchronously which makes it very efficient at doing updates to code. Once the page receives a request generated by a button, AJAX can update the page accordingly.

A third possible feature is the implementation of a login page. You could again use HTML, JavaScript, and JSON to implement a log in page but it’s not very secure and could easily be defeated. Using PHP and MySQL databases, you can make sure that users are properly authenticated and log them into the website. It is also easier to store session data about the user with PHP. I could not even imagine trying to accomplish with JavaScript and JSON while is it done with only a few lines of code of PHP.

* **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.
* One action that can be taken to secure a web application is the use of hashing algorithms, with a hashing algorithm, the person trying to intercept the data would need to have actual access to the algorithm to encode the data to then decode it. For example, if a database were to be leaked that contained users’ passwords, if the passwords were hashed then the attackers wouldn’t have plaintext passwords and makes them almost useless to exploit. Another action that can be used to secure a web application is the use of salts. If an attacker was to have access to the hashing algorithm used to hash a password, if there was a salt attached then it would be next to impossible to decode. Salts are long strings that are concatenated to the ends of hashed data to prevent reverse hashing. This is because salts are unique to every user which eliminates the possibility of decoding a hash with a salt that matches other hashes with the same salts.

#### 

#### **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'])) {

$pstmt = 0; // variable needs to be declared outside of if statements

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']);

}  
 }

The code first checks if an input in a form was submitted. Then it checks if the input was not empty. After that, the code then prepares a MySQL database statement where the lname column is selected in the customers table, which is then bound to a parameter. If the input is empty, then the code outputs a message and then prepares a statement which selects the entire customers table. The statement is then executed, and a while loop prints out the fname column which has the matching lname column.

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

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

});  
 });

The code checks for when an element with the id=”trigger” is clicked on. Once the element is clicked, a JSON file with the name people.json is parsed and the data is accessed. Then, a loop that prepares to print out the name and profession of each person on the JSON file with the parameters of key and val. This makes variables for the key of the JSON object and its value. The data is then output on the users browser which is done through an alert which creates a pop up on the top of the users web browser each time the output is generated.

#### **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.