Quiz 2

Written answers:

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. Proper form validation
    a. Front-end form validation, has its uses in giving the user an immediate response in whether or not the form is valid or not, can be easily circumvented. In order to ensure that the contents of the form that have been inputted are proper, one would have to use PHP to make server-side validation, where no matter what the user tried to do in the front-end side of things the server will still have the proper data to check.
2. Proper use of authentication/authorization
    a. In order to have a login and hierarchy system that is consistent across all devices accessing the site, there needs to be a central database where the server can compare the credentials being presented to a table of actual users that have accounts/permissions. You can do this in one instance with front-end code, but if you wanted to have secure authentication/authorization, one would have to have some server-side component come into play in order to ensure that everything is valid.
3. Exchanging data through multiple different applications
    a. There are ways to do this purely through front-end, but it is tedious and not the best way to go about it. A way to have a centralized database where users from any machine can read in the same data as anyone else would be much easier to implement with back-end code. This would involve establishing a connection with a certain database and then querying it for the desired information.

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. Salting and hashing passwords
    a. Using PHP's built in encrypt_password() function, the salting is already applied for you when you provide a password and the password_verify() function takes care of

verifying the password regardless of the salt (as long as it was hashed using the encrpyt_password() function). This way, passwords are never stored in plaintext and a data breach would only net the hashed passwords, which would take lots of time to decrypt given that they were hashed by a good function.

2. Back end form validation and input sanitization
    a. Although front-end form validation works if the user does not attempt to change the front-end side of their code, once they do it becomes trivial to bypass front-end form validation. Back-end form validation, with input sanitization to protect against SQL injections, would be a lot more secure method to validate if the form was filled in properly or not.

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']);
          }
     }
```

This piece of code checks to see if we are trying to either look up a specific person by their last name or everyone that we know of. If there is someone that we know that has the last name that we are looking for, then we show all instances of that person with that last name on our webpage. If we don't specify who we are looking for, we then show everyone that our code remembers on the web page.

Line 1: The checks to see if there is a query string in the URL (specifically ?lname='something')
Line 2: Checks to see if the query is not an empty string or not
Line 3: Prepares the prepared statement where you are selecting all customers with a given last name
Line 3: The line above takes one parameter, here we define that parameter to be the string after lname=
Line 4: if the lname that we specified is an empty string ('') then we go here
Line 5: Writes to the output "lname not given, outputting entire file"

Line 6: Creates a prepared statement where this time we just select everything (*) from the customers table
Line 7:
Line 8: executing the prepared statement defined in either line 6 or line 3
Line 9: Creates a while loop where as long as there are rows to fetch in $pstmt the contents of the loop will run
Line 10: Prints 2 elements in the associative array $row, the first one is the one with the key 'fname' and the second is with the key 'lname'.

2.2.

```
     $('#trigger').click(function(e) {
         $.getJSON('people.json', function(data) {
                $.each(data.people, function(key, val) {
                     alert(val.name + ", " + val.profession);
         });
     });
});
```

This piece of code runs whenever someone clicks a button that has a special id called "trigger". When the button is pressed, it goes through a file called people.json and shows you each person's name and job.

Line 1: using the jQuery .click function for anything with the id "trigger"
Line 2: AJAX call with jQuery, if successful the data variable will be a JSON object with everything in the people.json file.
Line 3: For each loop with jQuery, looping through the the items at the key 'people' in the data object and creating a function there
Line 4: Inside the function we just create an alert that gives us the values of the val object (which is just data at the key of people) at the key of name and profession with a comma and space in between.
Line 5-7: closing brackets/parenthesis.

Extra Credit (+5 points)
Name the chat protocol developed at RPI in the 1990s.

The 80/20 Model