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.

There are many possible features that are made easier by using a language such as PHP. One of these features is the ability to easily write to a database. PHP allows a fast and easy way to call to, retrieve data from, and even edit information and tables within a database. Programmers can easily create functions and variables that loop through a given database and only retrieve the required information. The second feature that is made easier by using PHP is when multiple functions relate to one another and can be created within a PHP class. A PHP class allows the programmer to create multiple functions using the same variable names and values, making code a lot cleaner, concise, and compact. An example of this would be our Lab 6 calculator lab, in which we created multiple functions within a class with the same variables, in order to carry out different operations on the same numbers. Instead of having to declare and re-declare the same variables and processes over and over again, we are able to organize code in a way that is easily accessible and effective. Lastly, A third component that is made easier by PHP is application security. Security is crucial for any application, and because PHP allows secure connections to databases and an easy way to both start and end user sessions to databases, it helps improve user and application security. For example, when a user logs out of a given site, PHP has the ability to end their user session, and revoke any and all privileges to the site by just using a few simple PHP lines of code, which greatly improves application security.

(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.

There are many ways to secure a web application, however two main methods can be to create a secure login and logout feature that is connected to a database, and to better password storage. For the first point, having a login / logout feature to an application is crucial to any application. It prevents individuals from visiting the site with the intentions of stealing a user’s personal information or compromising their personal security. In order for a user to access their information, they must enter a valid username and password, which is stored within a secure database. If entered correctly, this allows the user, and only the user, to view their account and any information that is held inside. However, if there was no login / logout feature to a site, then anyone could go into any given account or site information and view the data themselves. Building upon this, the second way to secure a web application is using improved password storage. For example, a user should never use a plain-text password, but instead should cryptographic hashes. Cryptographic hashes take a password and creates a new, improved hash version of the password by using some form of crypto: which is almost like "morse-code" in a sense that letters, numbers, and symbols are created in a pattern to equal the alphabet, but in a complicated way. This increases password security by making the password much harder and time-consuming to potentially decrypt and decreases the odds of someone logging into another person’s account through trial-and-error or hacking.

Part 2:

(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 { //with the wrong if statement (should be with the first, not the second)

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'])

} //One too many ‘{‘

Explanation 1:

This code first asks the user to enter their last name, most likely with the intention for logging into a personal website account or something of a similar nature. This code then compares this last name the user input to the list of all the last names stored, and if there is a match, returns the first and last name of the user. Additionally, a simple error check is put into place in which if the user does not enter a last name, it will return the message that no last name was given and output all of the data from the list of first and last names.

Explanation 2:

This code first checks to see if there is a value set to the lname variable using the $\_GET call. An if statement is then used to then check if the user has put anything into the field or not. If the lname variable is not empty, which is the second if statement within the code block, then a prepare statement is used to retrieve where the value the user put in is equal to the 'ln' variable is inside the database. After this, it then binds the lname and ln values to each other, due to the fact that both of these values are equal. The next if statement is for the case of if the user did not enter a value for lname (aka is an empty string). If this is the case then the code returns an error message, letting the user know, and then retrieves all of the first and last names from the database into a single variable. Lastly, now outside the scope of these two if statements, and execute() is called onto the variable which now holds all of the required info, and a printf statement is used to print out the first and last name (or names, depending on the situation).

(2.2)

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

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

    });  
 });

Explanation 1:

This code says once a button or some other box is clicked on by a user, then to get the data

from the list of all the data they possess. From there, the two values within this log, the persons name and profession, are returned back to the user on the screen.

Explanation 2:

This code states that once the “trigger” on the site is clicked (could be anything from a hyperlink to a button, to an image, etc.) then to activate function ‘e’. From there, a get request is used to retrieve data from a JSON file, more specifically the information found under the object of ‘people’. Next, the code says for each key-value pair under the object of ‘people’, to alert or output the infromation to the user in the form of “name , profession.”

Part 3:

(3.1)

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);