**Cheat Sheet**

1. Shuffle array: shuffle()
2. Include php file: include\_once “file\_path”
3. Check empty variable: empty()
4. Determine if a variable is declared and is different than null: isset()
5. Get string length: mb\_strlen()
6. Convert special characters to HTML entities: htmlspecialchars()
7. Returns the JSON representation of a value: json\_encode()
8. write data to a file: file\_put\_contents()
9. Checks wehter a file or directory exists: file\_exists()
10. Open file or URL: fopen()
11. Close file or URL: fclose()
12. Decodes a JSON string: json\_decode()
13. defines a named constant: define()
14. Send a raw HTTP header: header()
15. Format a local time/date: date()
16. Set a user-defined error handler function: set\_error\_handler()
17. Sets a user-defined exception handler function: set\_exception\_handler()
18. Output one or more strings: echo
19. HTTP POST variables: $\_POST
20. HTTP GET variables: $\_GET
21. Finds whether a variable is a number or a numeric string: is\_numeric()
22. Finds whether the type of a variable is float: is\_float()

**Important Header**

send all the HTTP headers required to prevent page caching, so when users will reload the page, they will always get the latest version of your files.

header('Expires: Sat, 03 Dec 1994 16:00:00 GMT');

header('CacheControl: no-cache');

header('Pragma: no-cache');

header('Content-type: text/html; charset=UTF-8');

**Error Handling**

If an error/exception occurs in your code, save the details into a specific folder and log file (give them names). The log file should contain all these details: the description of the error, the error code (if available), the date and time when the problem happened (format should be year/month/day hour:minute:second), the name of the PHP file and the line number of the error. You should also create a boolean constant to show the details in the browser (for debugging) or to hide the details to the users and just display a generic error (for production).

function manageError($errorNumber, $errorString, $errorFile, $errorLine) {

global $currentDateTime;

$detailedError = $currentDateTime . " - An error " . $errorNumber . "{" . $errorString . "} occurred in the file " . $errorFile . " at line " . $errorLine;

if(DEBUGGING\_MODE == true) {

#for developers

echo $detailedError;

}

#for end-user

echo "<br>An error occurred";

#save in the file the detail error

$data = $detailedError;

$JSONdata = json\_encode($detailedError);

file\_put\_contents(FILE\_ERROR, "$JSONdata\r\n", FILE\_APPEND);

exit(); # kill PHP

}

function manageException($exception) {

global $currentDateTime;

$detailedError = $currentDateTime . " - An exception " . $exception->getCode() . "{" . $exception->getMessage() . "} occurred in the file " . $exception->getFile() . " at line " . $exception->getLine();

if(DEBUGGING\_MODE == true) {

#for developers

echo $detailedError;

}

#for end-user

echo "<br>An exception occurred";

#save in the file the detail error

$JSONdata = json\_encode($detailedError);

file\_put\_contents(FILE\_ERROR, "$JSONdata\r\n", FILE\_APPEND);

exit(); # kill PHP

}

set\_error\_handler("manageError");

set\_exception\_handler("manageException");

**Read and display the file data**

if(file\_exists(DATA\_FILE)){

#open the file

$fileHandle = fopen(DATA\_FILE,"r");#use r for Reading

#read from a file

while(! feof($fileHandle)){

#in the project: read the file, decode the JSON string,

#use the array to fill the HTML table

$fileLine = fgets($fileHandle); #read a line in the file and put it in variable

if(($fileLine)!=""){

$array = json\_decode($fileLine);

echo "<tr>";

foreach ($array as $key => $value) {

if($key == 0) {

echo "<th scope='row'>".$value."</th>";

} else if ($key == 5 || $key == 7 || $key == 8 ||$key == 9) {

?>

<td class="text-end

<?php

if($key == 7 && isset($\_GET["command"])) {

if ($\_GET["command"] == "color") {

if($value < 100.00) {

echo " red-text";

} else if ($value < 999.99) {

echo " orange-text";

} else {

echo " green-text";

}

}

}

?>"><?php echo $value."$</td>";

} else if($key == 6) {

echo "<td class=text-end>".$value."</td>";

} else {

echo "<td>".$value."</td>";

}

}

echo "</tr>";

}

}

#close the file

fclose($fileHandle);

}

**Add css completion using PHP**

<img

class="logo

<?php

if($curPageName === FILE\_ORDERS && isset($\_GET["command"])) {

if ($\_GET["command"] == "print") {

echo "print-screen";

}

}

?>

"

src="<?php echo WEBSITE\_LOGO ?>" alt="website-logo"

/>

**Form Field Validation on submit**

#check if the user clicked the submit button

if(isset($\_POST["submitButton"])) {

#variable Validation and save the POSTed data into a variable

if(empty($\_POST["productCode"])) {

$errorOccurred = true;

$errorProductCode = "Please enter the product code";

} else if(mb\_strlen($\_POST["productCode"]) > 12) {

$errorOccurred = true;

$errorProductCode = "Product code not be more than 12 characters";

}

else if(mb\_substr($\_POST["productCode"], 0,1) != 'p' && mb\_substr($\_POST["productCode"], 0,1) != 'P') {

$errorOccurred = true;

$errorProductCode = "Product code should start from P";

}

else {

$productCode = htmlspecialchars($\_POST["productCode"]);

}

if(empty($\_POST["firstName"])) {

$errorOccurred = true;

$errorFirstName = "Please enter the first name";

} else if (mb\_strlen($\_POST["firstName"]) > 20) {

$errorOccurred = true;

$errorFirstName = "First name can not be more than 20 characters";

} else {

$firstName = htmlspecialchars($\_POST["firstName"]);

}

if(empty($\_POST["lastName"])) {

$errorOccurred = true;

$errorLastName = "Please enter the last name";

} else if (mb\_strlen($\_POST["lastName"]) > 20) {

$errorOccurred = true;

$errorLastName = "Last name can not be more than 20 characters";

} else {

$lastName = htmlspecialchars($\_POST["lastName"]);

}

if(empty($\_POST["city"])) {

$errorOccurred = true;

$errorCity = "Please enter city name";

} else if (mb\_strlen($\_POST["city"]) > 8) {

$errorOccurred = true;

$errorCity = "City name can not be more than 8 characters";

} else {

$city = htmlspecialchars($\_POST["city"]);

}

if(mb\_strlen($\_POST["comment"]) > 0){

if (mb\_strlen($\_POST["comment"]) > 200) {

$errorOccurred = true;

$errorComment = "Comment can not be more than 200 characters";

} else {

$comment = htmlspecialchars($\_POST["comment"]);

}

}

if(empty($\_POST["price"])) {

$errorOccurred = true;

$errorPrice = "Please enter the price";

} else if (is\_numeric($\_POST["price"]) || is\_float($\_POST["price"])) {

$floatPrice = (float) $\_POST["price"];

if($floatPrice > 10000) {

$errorOccurred = true;

$errorPrice = "Price can not be higher than 10,000.00$";

} else {

$price = htmlspecialchars($\_POST["price"]);

}

}

else {

$errorOccurred = true;

$errorPrice = "Price can not be higher than 10,000.00$";

}

if(empty($\_POST["quantity"])) {

$errorOccurred = true;

$errorQuantity = "Please enter the quantity";

} else if ($\_POST["quantity"] < 1 && $\_POST["quantity"] > 99) {

$errorOccurred = true;

$errorQuantity = "Quantity should be between 1 to 99";

} else {

$quantity = htmlspecialchars($\_POST["quantity"]);

}

if($errorOccurred == false) {

#calculate the sub total, taxes amount and grand total

$subTotal = (float)$price \* (float)$quantity;

$taxesAmount = ($subTotal/100) \* LOCAL\_TAXES;

$grandTotal = $subTotal + $taxesAmount;

$roundedTotal = round($grandTotal, 2);

#save all the values in file

$data = array($productCode, $firstName, $lastName, $city, $comment, $price, $quantity, $subTotal, $taxesAmount, $roundedTotal);

$JSONdata = json\_encode($data);

file\_put\_contents(DATA\_FILE, "$JSONdata\r\n", FILE\_APPEND);

# clear all the variables

$productCode = "";

$firstName = "";

$lastName = "";

$city = "";

$comment = "";

$price = 0;

$quantity = 0;

#show success message

// echo "<meta http-equiv='refresh'content='0'>";

$success = true;

}

}

Notes After Midterm:

1. PHP Session:
   1. To start the session: session\_start()
   2. Get or assign values to session variable: $\_SESSION[“variable\_name”]
   3. To remove all the global session variables: session\_unset() and session\_destroy()
2. Universally unique identifier: UUID()
   1. 36 char alphanumeric string
   2. Impossible to generate same UUID
3. SQL Stored Procedures: A stored procedure is a prepared SQL code that you can save, so the code can be reused repeatedly.
4. $\_ENV: Environment Variables
5. $\_SERVER: array containing information such as headers, paths and script locations.
6. Try-Catch: In the try block, you do some tasks e.g., reading a file. If an exception occurs, the execution jumps to the catch block. In the catch block, you specify the exception name and the code to handle a specific exception.
7. PDO: PHP Data Object
   1. Advantage:
      1. Database Support
      2. Database Connecting
      3. Error Handling
         1. Silent: it is a default error mode
         2. Warning: It is useful for debugging
         3. Exception: This mode allows graceful error handling while hiding data that a person might use to exploit your system.
      4. Insert Update
         1. Prepare >> [Bind] >> Execute
   2. Benefits of using PDO
      1. Usability
      2. Reusability
      3. Security
   3. PDO Classes
      1. PDO
      2. PDOStatement
      3. PDOException
   4. Database supported by PDO: Almost all the sql databases
8. PHP OOP – Classes and Objects: A class is a template for objects, and an object is an instance of class.
9. AJAX:
   1. Asynchronous JavaScript And XML
   2. Not a programming language\
10. Store Password:
    1. password\_hash(): creates a password hash
    2. password\_verify(): verifies that a password matches a hash
11. SQL BLOB Type: Blob is the data type in SQL that helps us store the object in the binary format. Like: images, pdf, doc etc. Blob datatypes:
    1. TINYBLOB – length + 1 bytes, where length<2^8
    2. BLOB – length +2 bytes, where length<2^16
    3. MEDIUMBLOB – length + 3 bytes, where length<2^24
    4. LONGBLOB – length + 4 bytes, where length<2^32
12. HTTP vs HTTPS
    1. HTTP stands for Hypertext Transfer Protocol. HTTP stands for Hypertext Transfer Protocol.