CS 325 Midterm Project Report

Building and Designing a full website using:
HTML/CSS/JavaScript
JQuery
&
Testing PHP Knowledge through flat files

CS 325
Fall Semester
Academic year: 2019-2020
Worked By: Daniel KALEMI
Received By: Dr. Chris Christodoulou

1. DESCRIPTION

This program in Distributed Applications & Web Development was intended to practice on the learning material as part of the CS325 course. Specifically, the purpose of this project is to create a self-contained, hands-on project integrating all the knowledge gained so far on PHP.

2. TASKS

The requirements of the program were:

A fully functional Website with HTML pages, external CSS, and JavaScript files. What's important is that covering the PHP requirements we also apply and add to what we have learned about HTML, CSS JavaScript, and create an easy-to-read and navigated site.
The site will contain a home page (index.html) and 6 additional pages.

APALLOU CAFÉ WEBSITE
I have created a fully responsive 7-page website for a bar-restaurant and patisserie business in Thessaloniki.
The pages include:
HOME
BUY
PURCHASE DETAILS
DIRECTIONS
CUSTOMER DETAILS
PAYMENT
TEST FILES

DESIGN WALKTHROUGH
UI

By building this project, I was able to follow through the steps of creating a website where the customers can:

1. Check options online on available cakes HOMEPAGE
2. Select various products (CLICK THE ORDER BUTTON, OR THE BUY SECTION AT THE NAVIGATION BAR
3. Select the amount (size and number of orders)
4. Add special requirements to the order
5. Having the option of fast/standard delivery (fast delivery has an extra charge of $2).
6. OPTIONS 3,4,5 are on the Purchase FORM (FORM NR. 1)
7. After we press submit and pass the validation process with no errors, the information provided in the form is saved on a flat-file called ***"purchase_data.txt"***
8. The website directs us to ***FORM NR. 2: CUSTOMER FORM.***

9. Here we have the option to put our credentials and the address where we want the delivery to be sent.
10. All the fields require some time of validation (email, no special characters, etc.)
11. After we successfully pass the validation with no errors, the data entered in the fields of this form will be saved to a flat-file named ***"customer_data.txt"***
12. After gathering the order info and the customer details then the website will direct the customer to the 3rd and final form, ***FORM NR. 3: PAYMENT FORM.***
13. The form will require credit/debit card details like the **CARD NUMBER, EXPIRATION DATE, AND THE CVV NUMBER** (3 digits on the back)
14. After we successfully submit the form with no errors, the form data will be saved to a third flat-file ***"pay_data.txt"***
15. Finally, the website will direct the user to the ***TEST FILES*** webpage where we can see all the data we filled out in the forms.
16. The data is actually read and displayed directly from the flat files, which then passes to the website in a peasant formatting through some CSS queries.

17. ***NOTE:*** *ANY OF THE PAGES, EXCEPT THE CUSTOMERS & PAYMENT WEBPAGES CAN BE ACCESSED THROUGH ANY OF THE STAGES OF THE PROGRAM (ON THE NAVIGATION BAR). The reason why those 2 web pages are not made visible through all the stages is that they require their data to be saved on the flat files. But since in this project, we want a tidy and chronological input we need to access such pages only through 1 session per time to have more qualitative data displayed at the end.*

CODE DESIGN
HTML/CSS/JS/JQUERY/FLEXBOX

However, there are many more additional links in the form of buttons, clickable multimedia content, etc.

There is a wide usage of many of the techniques that we learned in class in addition to many more advanced techniques that I had the chance to search about while I was working on this project.

Some of the techniques are:
Tables
Forms (with JavaScript & PHP functionality)
Grid
Gallery Slideshow (with JavaScript functionality)
Flexbox
JavaScript functions
Color Gradient using keyframes
Animations and keyframes
Color changing using animation and keyframes
Sticky navigation bar
Importing APIs, iframe, and frameworks for fonts, animations, and figures

Material Design
Building Parallax Webpages (where the background image is static and the front material of the page is moving to create an animated visual
CSS Animation Design (typescript, color change, object animation, text manipulation, etc.

All the 7 pages, along with the CSS files have passed the W3C Validation Test. I was testing my code live all the time since in Visual Studio Code there is a plugin from W3C to install and validate while working.

The content is 1100px. (within the 1200px suggested)
The navigation is visible and accessible without scrolling.
Each page links to all other pages.
I have used a "sticky" class as a div around the navigation bar and I used it for creating a sticky menu bar (fixed navigation). However, I have commented it out since it didn't work on some of the pages due to the extra techniques that they had. For those pages where I used a parallax background, the menu bar was disappearing since the background was moving. However, on all the other websites the feature worked properly.
The instructor is encouraged to use the feature by copying this line in CSS:

.sticky{position: fixed;}

The text is large enough to read and contrasts strongly with the background color.
Each page has at least one heading and at least one paragraph of text (the home page can skip the heading.)
The heading elements and paragraph elements are styled in CSS.
Spelling and grammar are immaculate.
At least one page includes a list coded with HTML list tags.
At least one page contains a form.

CODE DESIGN – PHP

On the final output of the code structure and design idea, **efficiency, simplicity & creativity** have been the main drivers along with the main goal to cover all of the material learned so far.

CHRONOLOGICAL ORDER OF CONCEPT COVERAGE IS ILLUSTRATED IN THE PROJECT.

- **There is a wide coverage on the .php files on this project of concepts like:**
- **Escaping special characters (\r,\n,\t)**
- **Variables ($amount)**
- **Datatypes & Common Arithmetic Operators**

- **String Concatenations (when saving the data to flat files)**
- **Comparison Operators**
- **Logical and Assignment operators**
- **Input using forms**
- **Controlling the flow of the program (as shown in the design walkthrough)**
- **Formatting Strings**
- **Build functions (for the validation, for calculating the total amount to pay, for the multiple inputs to put them on an array a then pass them to a variable in order to be saved to the flat file.**

```php
function clean_text($string)
{
 $string = trim($string);
 $string = stripslashes($string);
 $string = htmlspecialchars($string);
 return $string;
}
```

- **Working with arrays (Especially with the "select" and "option" inputs as they have multiple variations and can be easily converted to arrays in the PHP part**

```php
//WE COULD ALSO USE A CSV FILE TO PUT CONTENTS OF THE FIELD ALTERNATIVELY
 /* $form_data = array(
   'sr_no'   => $no_rows,
   'name'   => $name,
   'email'   => $email,
   'subject' => $subject,
   'message' => $message
  );
  fputcsv($file_open, $form_data);*/
```

```php
$c = array ( "01" => 1, "02" => 2, "03" => 3,"04" => 4, "05" => 5, "06" => 6,"07" => 7, "08" => 8, "09" => 9,"1
$month = $_POST['month'];
foreach ($month as $m) {
  $month1 = $month1 + $c[$m];
}

$c1 = array ( "2020" => 2020, "2021" => 2021, "2022" => 2022, "2023" => 2023, "2024" => 2024, "2025" => 2025);
$year = $_POST['year'];
foreach ($year as $y) {
  $year1 = $year1 + $c1[$y];
}
```

-Use global variables

```php
<?php
//customer.php

$error = '';
$name = '';
$email = '';
$subject = '';
$message = '';
```

-Work with files

```php
if($error == '')
{
 $file_open = fopen("contact_data.txt", "a");
 $no_rows = count(file("contact_data.txt"));

 if($no_rows > 1)
 {
  $no_rows = ($no_rows-1)+1;
 }
 $count=($no_rows/6)+1;
 $form_data= "Customer ".(int)$count."\r\n  Name: ".$name."\r\n Email: '
 fwrite($file_open,$form_data);
 fclose($file_open);
```

```php
<?php
// read file into string
$str = file('contact_data.txt') or die('ERROR: Cannot find file');

$str1 = file('purchase_data.txt') or die('ERROR: Cannot find file');

$str2 = file('pay_data.txt') or die('ERROR: Cannot find file');

?>
```

-Use data validation for output accuracy:

```php
function clean_text($string)
{
 $string = trim($string);
 $string = stripslashes($string);
 $string = htmlspecialchars($string);
 return $string;
}

if(isset($_POST["submit"]))
{
 if(empty($_POST["name"]))
 {
  $error .= '<p><label class="text-danger">Please Enter your Name</label></p>';
 }
 else
 {
  $name = clean_text($_POST["name"]);
  if(!preg_match("/^[a-zA-Z ]*$/",$name))
  {
   $error .= '<p><label class="text-danger">Only letters and white space allowed</label></p>';
  }
 }
 if(empty($_POST["email"]))
 {
  $error .= '<p><label class="text-danger">Please Enter your Email</label></p>';
 }
 else
 {
  $email = clean_text($_POST["email"]);
  if(!filter_var($email, FILTER_VALIDATE_EMAIL))  //method for validating email field
  {
   $error .= '<p><label class="text-danger">Invalid email format</label></p>';
  }
 }
 if(empty($_POST["subject"]))
```

Integrate PHP within HTML

```php
  <div class="a">
  <h2><b><u>Payment Info</u></b></h2>
   <p></p>
   <?php foreach ($str2 as $line2) {
   echo $line2."<br>";
   }?>
  </div>
div>
```

Further Development of the Program

The program managed to come up with some useful solutions, however, there are many more things that could be further developed so that we can take more advantage of it.

Some suggestions for further expansion would be:

1. Usage of more scalable and powerful features of PHP like database and integration of MySQL.
2. Adding versatility by adding more web pages and more flat files (we can write to-array or read-array elements from the file)
3. Add the features of UPDATE & DELETE
4. Using more dynamic styling features like AJAX and JSON/JS

Overall the more versatile and bigger in size the program becomes, the bigger will become the need to replace a flat-file system with a solid database system where manipulating the data and scaling the information would be more **flexible and efficient**.
However, for now, with what problems are presented for the current version, the flat file system and the solutions offered to solve such problems are quite efficient and easy to use not to mention that the different techniques covered in the teaching material, make this program quite preferable for small-scale solutions.