**Patient Scheduler PHP project Assignment**

The URL to access my prototype can be accessed on <http://cssrvlab01.utep.edu/Classes/cs3360/arodriguez103/index.php>

A helpful website that furthered my understand of PHP can be found by Googling “ PHP Tutorial – w3schools “ (HIGHLY RECOMMEND THIS WEBESITE)

The diagram below depicts how my project jumped and used each php file to perform certain tasks depending on what was being requested or required at the time.

A close up of text on a white background

Description automatically generated

**Explanation of Diagram**

Project starts at index.php and creates and displays the patient input fields to add them into the schedule and no make schedule button is present because at least one patient must be on the schedule in my approach. However, before index.php does all that it will check if the SESSION variables have been created and set for the project to be able to use for storing patient info and using the info stored to create the schedule in the end. If the SESSION variables have yet to be created, index will jump to PatientInfo.php, which displays nothing for the user, to create the needed SESSION variables, which are arrays, for the project. My approach requires every php file to have in the code *session\_start();* , which will allow all the php files to have access and update to the arrays that have been created in PatientInfo.php.

Once all that is set up and index.php is given its first patient info and is sent, the patient info given is now all sent to ProcessData.php. ProcessData.php primary purpose is to process the patient info and store them into the SESSION array. The ProcessData.php also doesn’t display anything to the user and once the patient’s info is finished processing and stored, the php file will then redirect the project to AddAndCreate.php. AddAndCreate.php will display the same patient input fields but will also include a create schedule button. The AddAndCreate.php will have two primary functions and the first is to allow the user to add another patient info into the schedule by using the add patient info fields on the page displayed. Once the info is submitted, it will be sent to ProcessData.php to be stored in the arrays for after being processed and then the project will then be brought back to the AddAndCreate.php once again but with the patient info fields cleared and ready for another patient info to be inputted. The second function is once the user finishes adding patients into the arrays, which is also the schedule, the user can click the button Create Schedule which the project will be sent to the Schedule.php.

Schedule.php is the final display page of the project where it will access all the SESSION variables of the project and use them into create the Patient Schedule Table. The patient schedule table will display each scheduled patient in a row and column the info associated with that patient. The first column displays appointment time, second displays patient’s name, the third displays patient’s address, the fourth displays the travel time it will take to get to that patient’s address, the fifth displays their preference in when to be met, sixth will display if they are a new or existing patient, and the final seventh column will display the delete patient button. Within this page, you have two options to modify the patient schedule table and its either delete patient from the schedule or optimize the travel time by moving around the patience in the schedule in order to decrease the total amount of driving time.

The two extra features, delete and optimize, in the Schedule.php are triggered by buttons. The two features are only for processing data purpose and will not display anything for the user when they direct the project to the assigned php file. To delete a patient from the schedule, go to that patient’s row and click the delete patient button which is located at the end of their row. This will direct the project to Delete.php, delete that patient’s info from the SESSION arrays and then redirected to Schedule.php where the schedule will be recreated by displaying only the patient information found in the SESSION arrays. The other feature is the optimize feature which can be triggered by clicking the optimize button found directly below the schedule. Once clicked, the project will be direct to Optimize.php where all the scheduled patients will be re-indexed to decrease the total travel time. This is done by comparing each patient’s map coordinates with all the other patient’s map coordinates and modify /setting the patient’s index to where they are next to the patient where the travel distance is the least between them. After the re-indexing is completed, Optimize.php directs the project to Schedule.php where the patient schedule will update the table accordingly since it depends on the index of the array to place each element/data into the appropriate position in the patient schedule table.

**The Logic and Approach**

The Session Variables are variables that can be used in any php file as long as it has *session\_start();* at the beginning of the php code. I decided that 6 main Session array variables will be needed that will store each patient’s info accordingly and will be used to create the patient schedule in the end. The index is what determines what info is associated with which patient. So if PatientZ is in index 3, then all its data in the other arrays are at index 3. There is a separate array for Patient name, Patient address, Patient Preference, Patient New or existing status, Travel time to get to the patient from the previous patient, and appointment time.

**Create Patient Schedule- Schedule.php**

So how the Patient Schedule is created is by taking 6 Session arrays, mentioned in the Logic and Approach, that contain the info needed to create a patient’s schedule on a row. The index is what determines the patient. So index 0 in all arrays are for the first patient, index 1 in all the arrays are for the second patient, and so on and so forth.

**Schedule table creation code piece**

<h1>Schedule Draft for Tomorrow</h1>

<table align=*"left"* border=*"1"* cellpadding=*"3"* cellspacing=*"0"* style="width:*100%*">

<?php

$length = *count*(**$\_SESSION**['PName']);

**for**($i=0;$i<$length;$i++) //the number of rows created for the table is determined by

//the max length of Patient Name array

{

**echo** "<tr>";

**for** ($j=1;$j<=7;$j++) // this creates 7 columns for each row and each element being

//displayed uses $i to track which patient info to display

{

**switch** ($j) {

**case** "1": // 1st column which displays the patient’s full appointment

**echo** "<td>Appointment time:".**$\_SESSION**['PAppointment'][$i]."</td>";

**break**;

**case** "2": //2’nd column displays patient name

**echo** "<td>Patient Name: ".**$\_SESSION**['PName'][$i]."</td>";

**break**;

**case** "3": //3rd column displays Patient Address

**echo** "<td>Address: ".**$\_SESSION**['PAddress'][$i]."</td>";

**break**;

**case** "4": //4th column displays patient travel time

**echo** "<td>Travel Time: = ".**$\_SESSION**['PTtime'][$i]."</td>";

**break**;

**case** "5": //5’TH column displays Patient’s schedule perference

**echo** "<td>Perferred Time: = ".**$\_SESSION**['PSpecial'][$i]."</td>";

**break**;

**case** "6": // 6th column will display if patient is New or Exist

**echo** "<td>Client Status: ".**$\_SESSION**['PNew'][$i]."</td>";

**break**;

**default**: // last column will hold delete button

**echo** "<td>Delete Patient"?> <html><form action=*"Delete.php"*, method=*post*><input type=*"submit"* name=*"Delete"* value="<?php **echo** $i ?>" /></form></html>

<?php "</td>";

}

}

**echo** "</tr>";

}

?>

**The Delete Function- Delete.php**

The Delete function I used to delete a scheduled patient on the created schedule consists of using one predefined method in php which the array\_splice(). It takes three parameters which are the array, start index, and length of what to remove in the array. So the first parameter you just but the $ArrayName and the second parameter is the $index in the array in which you want the deletion to begin. In this case my $del holds the index of the patient that is to be deleted. If you do not specify the third parameter, the array\_splice() method will delete elements in the array starting from the $index given and all elements after that index. So I set that parameter to 1 so that the method will only remove/delete the starting index. So for example, if the starting index was 5 and you set the third parameter to 3, it will delete all elements from 5 to 7 which is 3 elements that were requested to be deleted/removed. But wait!!! There is more! Array\_Splice() also reindexes the elements for you meaning if you remove only one index, it will reindex the other elements accordingly so there is not an index in the array that is empty. So all elements after the starting index will be re-indexed by their position/index minus 1.

**Delete.php code piece**

<?php *session\_start*();?>

<?php

//delete by index

$del = **$\_POST**['Delete'];

//array\_splice($arrayName, $StartingIndex, $NumberOfElementsToBeDeleted)

*array\_splice*(**$\_SESSION**['PStime'], $del, 1);

*array\_splice*(**$\_SESSION**['PEtime'], $del, 1);

*array\_splice*(**$\_SESSION**['PAppointment'], $del, 1);

*array\_splice*(**$\_SESSION**['PTtime'], $del, 1);

*array\_splice*(**$\_SESSION**['PBlock'], $del, 1);

*array\_splice*(**$\_SESSION**['PName'], $del, 1);

*array\_splice*(**$\_SESSION**['PAddress'], $del, 1);

*array\_splice*(**$\_SESSION**['PSpecial'], $del, 1);

*array\_splice*(**$\_SESSION**['PNew'], $del, 1);

*header*("Location: Schedule.php");

**exit**;

?>