ECM1417-WEB DEV-NA510-COVID CONTACT TRACING

Link to My Deployment (Please turn on server before use)

http://ml-lab-4d78f073-aa49-4f0e-bce2-31e5254052c7.ukwest.cloudapp.azure.com:63699/index.php

How I Addressed each of the points in the marking scheme

HTML Pages

I Addressed the HTML aspect of the program by writing one html document for each page. I decided that to prevent myself from reusing code for the header bar and the side navigation panel as well as improving the upgradability, maintainability and debugging, that it would be best to create 2 base html files, “base.html” and “base1.html”. This meant, when the pages corresponding php file is opened by the user, the necessary base files are included along with the relevant html file. I used 2 base files as the login and registration page had the top header bar, and the rest of the pages had this bar along with the side navigation panel. The use of 2 base files meant that no code was reused in this area of my html.

The content of some of the pages was validated by the PHP that processed the form, and when data was invalid, the page was re-rendered with a message explaining the error. I also used JavaScript to validate inputs as well to prevent excessive back and forth communication between the browser and server.

CSS

I included almost all my CSS code in the “mycss.css” file. This could be accessed by any html file which could use any of the classes I’d written. I strived to make my classes modular, and this allowed me to prevent the re-use of code. Within the html, tags were able to take style information from multiple CSS classes and this modularity made it easier to manage my CSS file.

PHP

When accessing my pages, each html page required a corresponding PHP file to be opened by the browser. This was because of my use of base html files to prevent code repetition. I also used PHP files to manage and manipulate the data that was submitted by my html forms. In most cases, each form on my site had its own PHP script in a file which checked data to make sure it was valid, and to connect to the database and store it. The exception I made was for the login and registration system where I decided to create a function to do the main login. This was to prevent re-using the login code when the newly registered user was to be logged in automatically.

ECM1417-WEB DEV-NA510-COVID CONTACT TRACING

Secure sessions and cookies

Upon login a session is created within the PHP, this session is also accessed by some HTML files when accessing information such as the users name in the homepage. Cookies are also created in PHP and are also used when the settings are changed by the user. If the user hasn't specified any settings before, default values are assumed. Upon logout, the php script is responsible for deleting the cookie and session before redirecting the user to the login page.

JavaScript

I Used JavaScript as a way to verify the form entries validity as well as in the addvisit page. The main use was to select a location from the map. To do this I created an event listener on the map-div (Meaning any click not on the map are not detected), and made it trigger a function to fill in 2 hidden fields of the form with the X and Y positions. If no coordinates were submitted, the php would send the user back to the addvisit page and display a message asking them to select a coordinate. I also used JavaScript in the overview page. When the cross button was pressed for a visit, it would call a JavaScript function. This function would find the ID of the row we are deleting (from a hidden td) and would trigger a php script, passing in the ID. The php script would then delete the relevant row and redirect the user back to the overview page.

Security

I used SALT when encrypting the password. This Cyphertext was stored in the database and when a user is to be logged in, the password they enter is compared with the cyphered password using the PHP salt library.