ml-lab-4d78f073-aa49-4f0e-bce2 31e5254052c7.ukwest.cloudapp.azure.com:55595/index.php

HTML pages for static content

I used HTML for all pages listed to display static content. On pages where user input is required (such as the login page and settings page) I used a HTML form, which allows for the data entered to be sent to a PHP script using the POST method.

CSS

I followed all requirements provided in the style guide. The CSS for the navigation bar is in a separate file, as it is reused over multiple different forms. All of the rest of the CSS is in the header of each HTML file.

PHP

PHP is used in every part of the system. PHP is used to connect to the contact_tracing database and make modifications to tables using SQL. This allows for the process of registration, login, checking infections, creating overview, adding a new visit, reporting an infection and changing the settings to occur. Any input data is sent to a corresponding PHP file which then processes that data.

Secure sessions and cookies

A session is created when the user logs in successfully and session variables are set which store the user ID of the person and the name of the person. A session variable is also set indicating that the user is logged in. This session is then destroyed and closed when the user clicks the log out option by using session_destroy. Cookies are set when the user enters their settings for distance and time window on the settings page.

JavaScript

I used JavaScript to remove a location from the overview on the client's side after a successful server response. I Also used JavaScript to select a location from the map on the add visit page by adding a function in the onclick attribute of the image representing a map of Exeter. When the user clicks on the map the function is executed, the hidden form inputs are filled using JS DOM methods and a marker is placed where the user clicked.

Ajax

I used Ajax to remove rows from the table in the overview page synchronously between the server and the current page. Removing a row waits for a reply from the sever which confirms that the row has been deleted from the database before being removed client side using JavaScript.

Web Service

When an infection is reported, all locations visited by that user in the database are reported to the web service using its report method. This is done using cURL. I only managed to partially implement the functionality for checking for infections, using cURL to get an array of all visited locations by an infected user in a given timeframe. I did not however manage to integrate this with the homepage to display locations retrieved from the web service on the map.

Security is Implemented

In accordance with the mark scheme, I implemented all security requirements listed. I hashed and salted all passwords of users by using the password_hash function in PHP and the algorithm PASSWORD_DEFAULT. When logging in password_verify is used to compare the salted and hashed password to the plaintext password entered by the user to see if they match. I ensured my page is secure against cross site scripting by using htmlentities to escape any script tags. I made my page secure against SQL Injections by using prepared statements whenever an SQL query is to be performed using user input data.