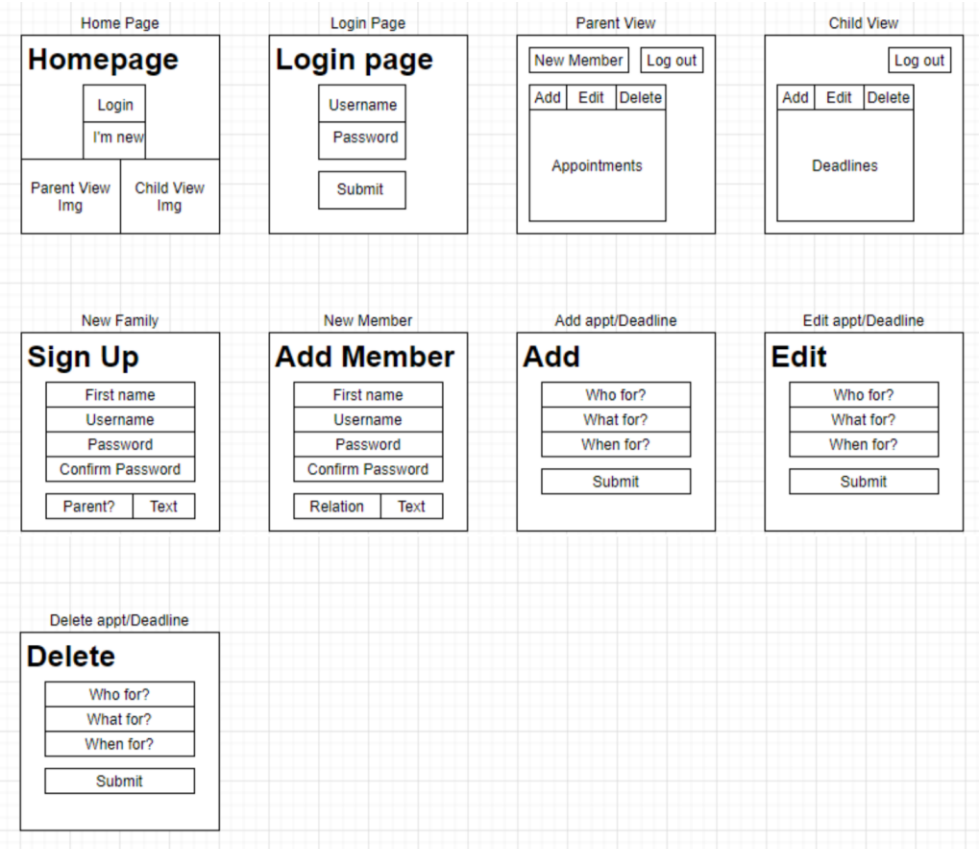
Application design document

Storyboard design

Initial ideas for my design

Figure 1: Screenshot below illustrates the initial design of the client-side webpage.

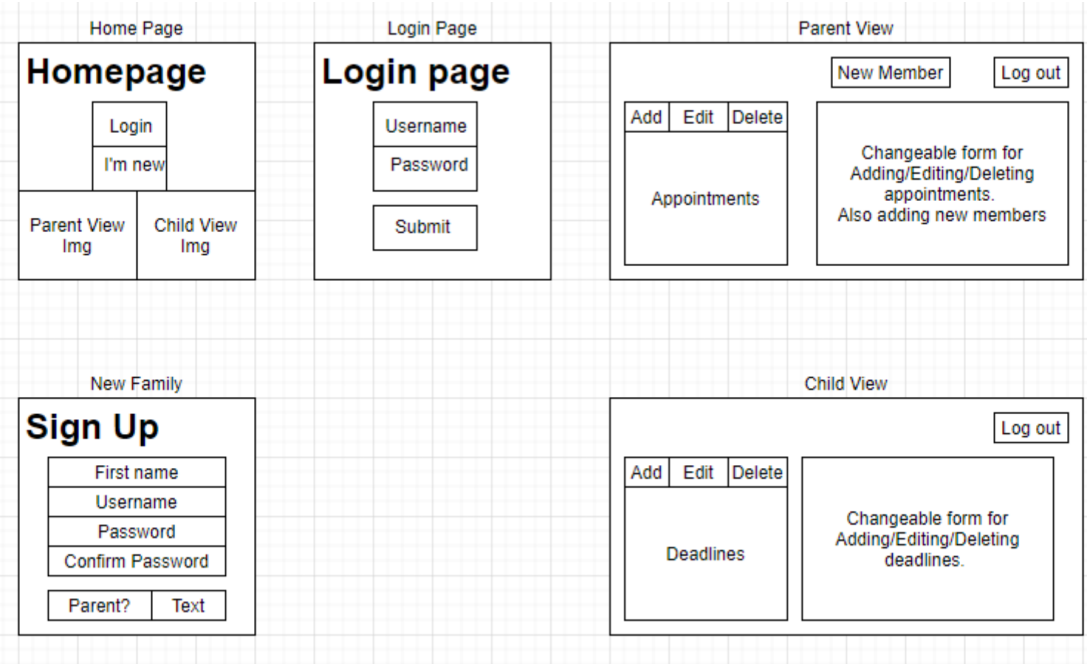
Figure 1.



Final storyboard design

Figure 2: Screenshot below illustrates the final storyboard design. Simplified to reduce file space.

Figure 2.

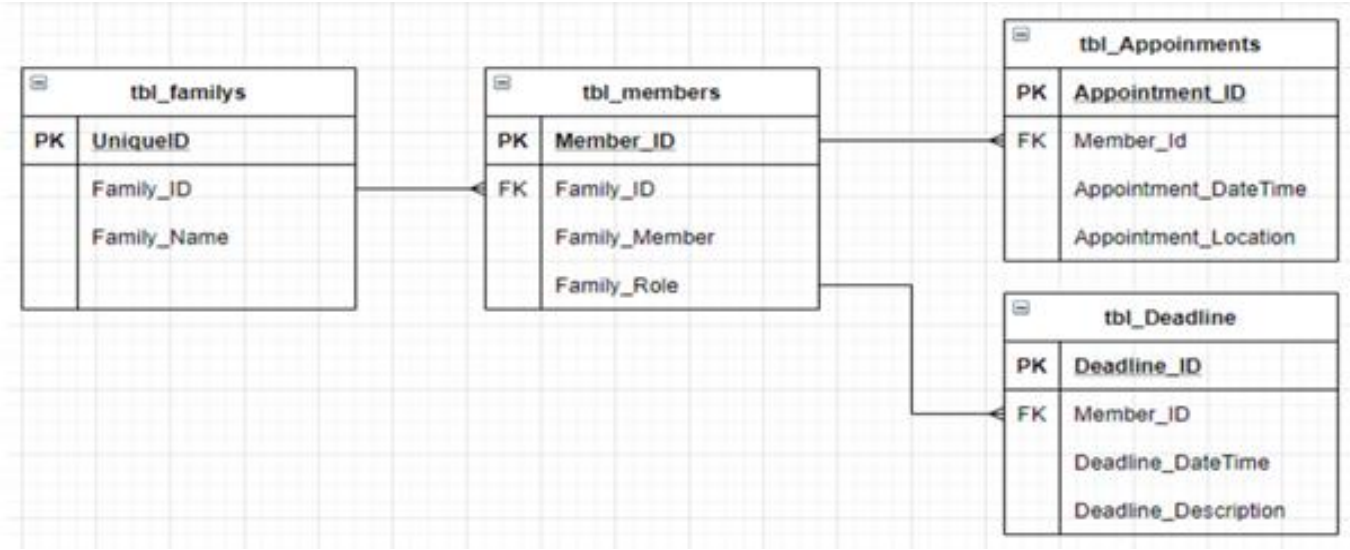


Entity Relationship Diagram

Initial database structure.

Figure 3: Screenshot below illustrates the capture of the relationships that were initially within the database.

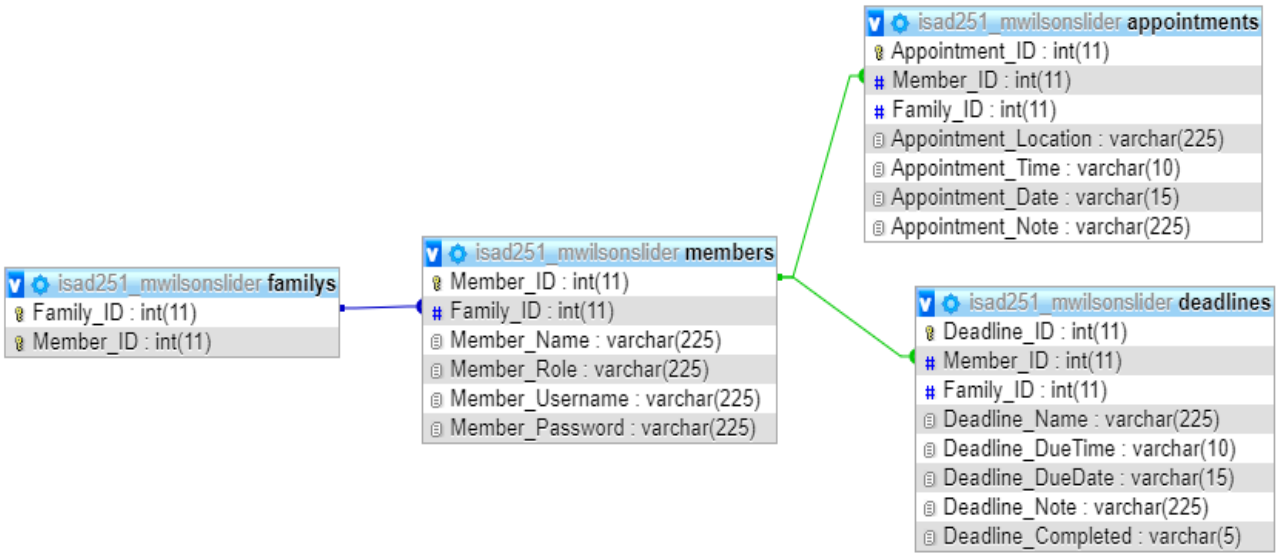
Figure 3



Final database structure

Figure 4: Screenshot below demonstrates the final database structure illustrating that figure 3 was implemented, although I had to add in additional information.

Figure 4.

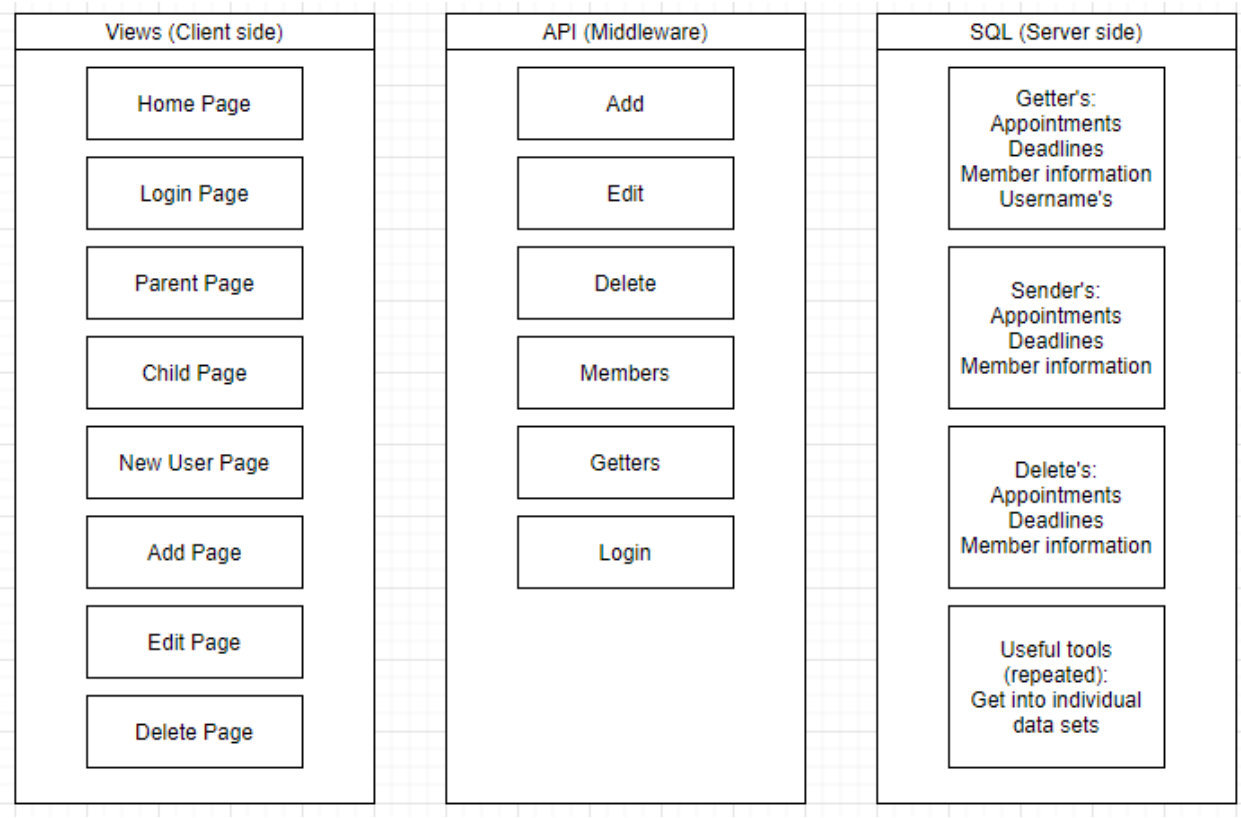


UML Diagrams

Initial UML Diagram.

Figure 5: Screenshot below demonstrates my Initial UML diagram captured in a screenshot: The UML diagram demonstrated different client-side, middleware and server-side coding.

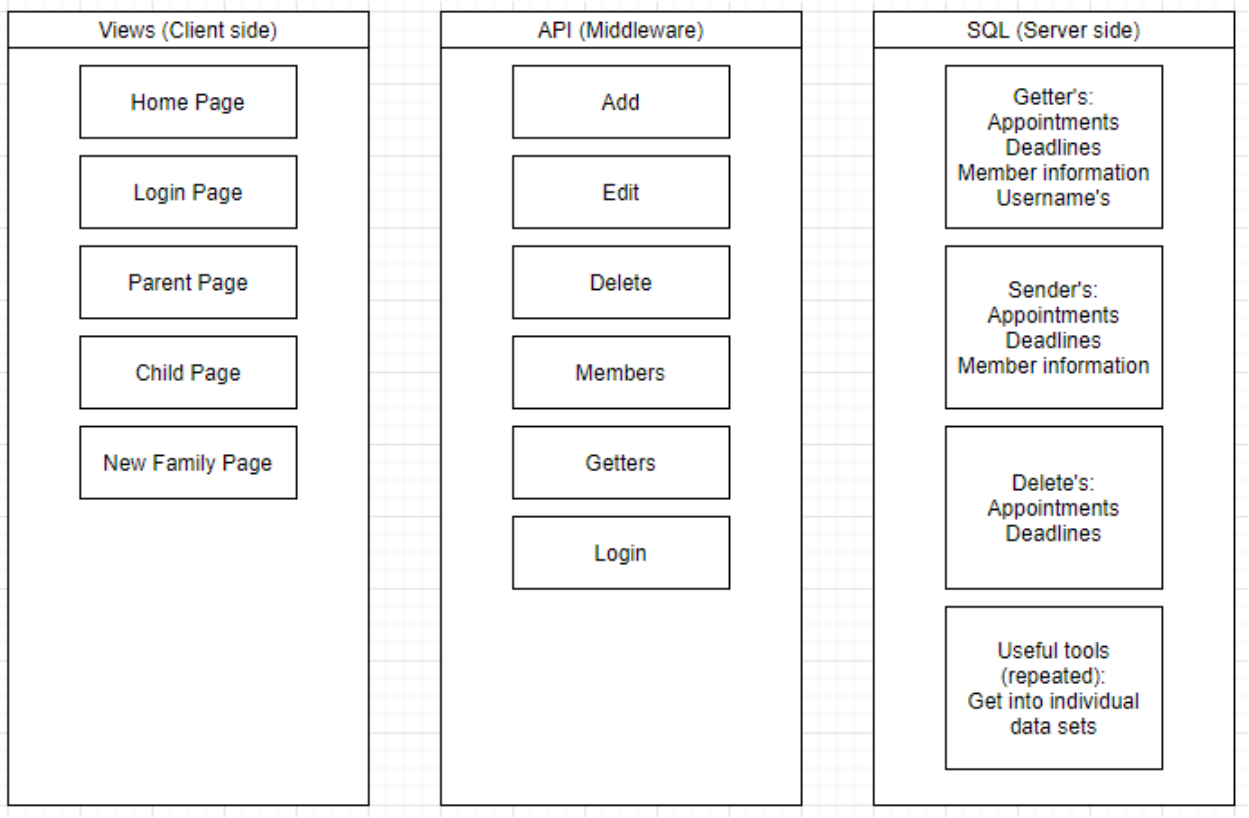
Figure 5



Final UML Diagram

Figure 6: Screenshot below illustrates changed client-side, middle-ware and server-side pages. I combined the pages together to use less coding and reduce storage, and simplify the use for the client.

Figure 6



Settings

The webpage will run with basic settings active; this application works best with Google Chrome. When using the webpage to add, edit or delete an appointment or a deadline the user will have to log out and relog in to view changes, this is due to AJAX not being used. To simplify the application, and make it more inclusive the interface will adjust to the size of the screen being used: ie if the screen is a tablet, pc or mobile phone the webpage can still be used. There are also pre-set usernames in the system to show example data (see README.md in appendix 1).

Images within the application

The following images in the figures below (7 & 8), have been taken from screenshots to demonstrate the log in views that can be used for parents and children. Although both have the same levels of security I wanted to make sure children could not add a parent user and use this to delete appointments.

Figure 7: Parent webpage

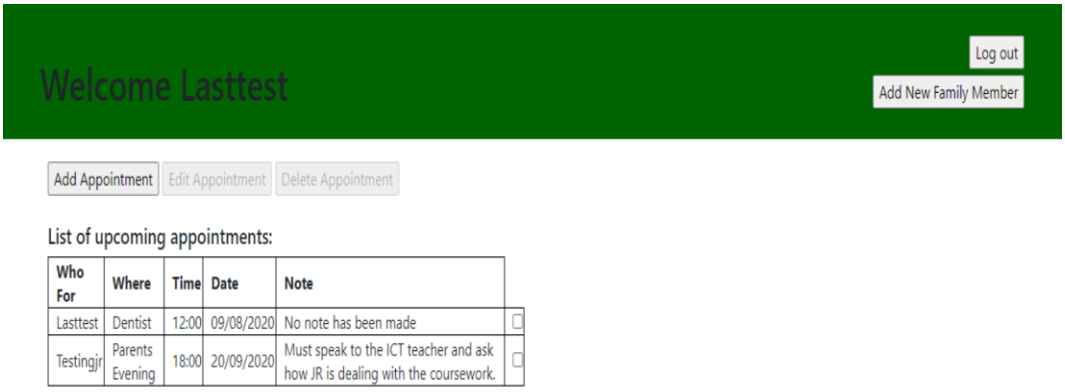
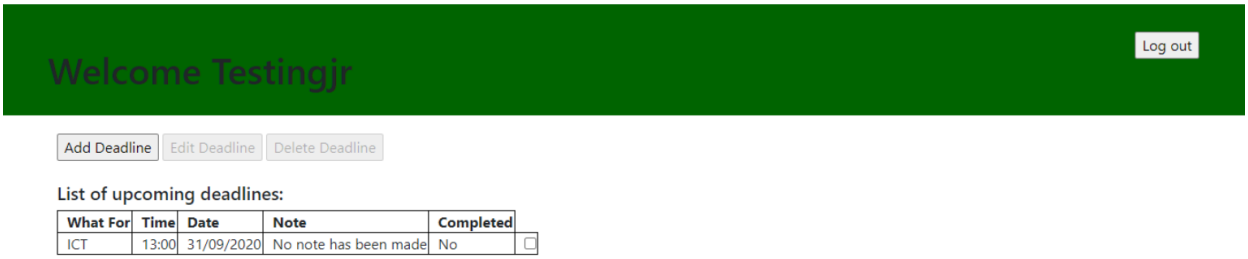


Figure 8: Child Webpage



Testing using WAI guidelines

To make sure that there were no errors within the application of my webpage, I used the weblink in appendix 2. The linked webpage highlighted missing tags, missing or grammatical errors and fatal errors. Figure 9-15 below demonstrate screenshots of errors found.

Results of errors

Figure 9 below: The Results of testing the home page

Figure 9.

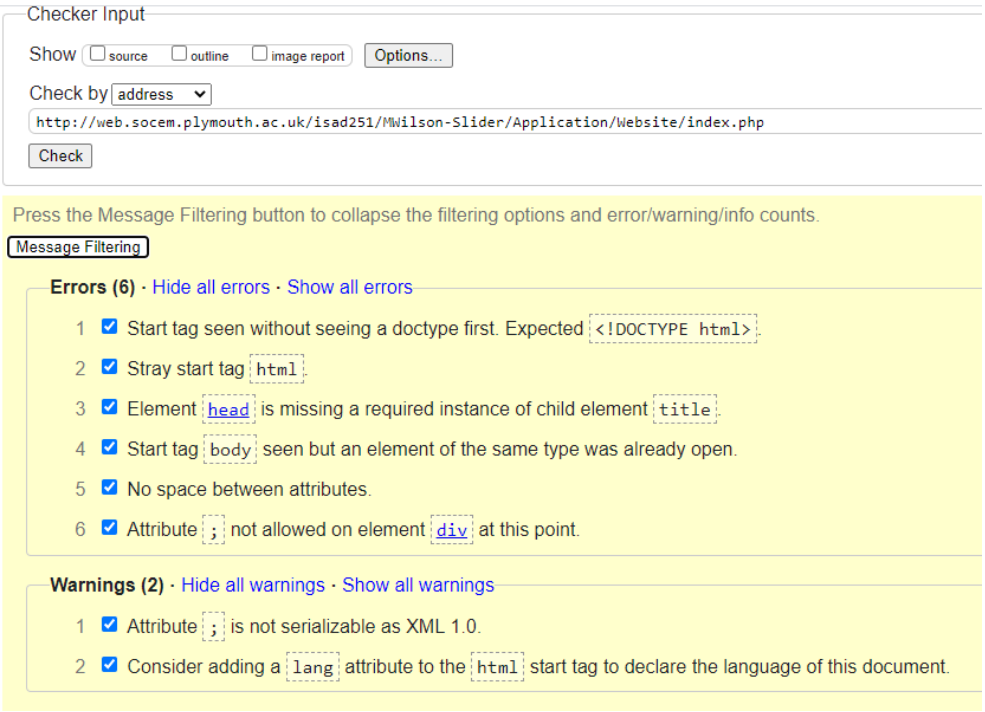


Figure 10 below: The results of testing the login page

Figure 10

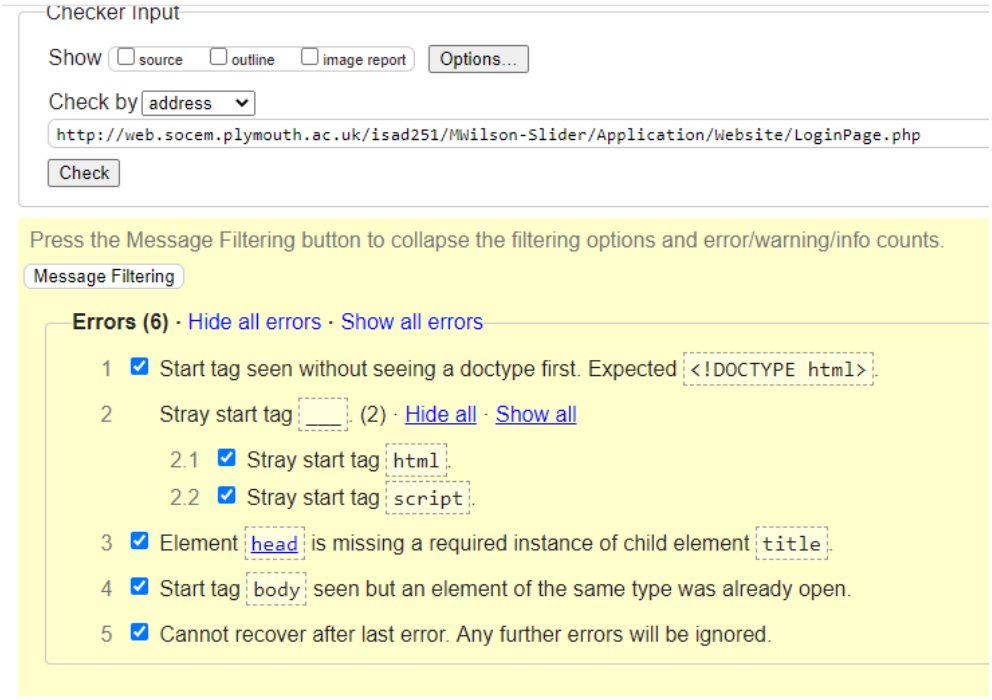


Figure 11 below: The results of testing the Parent page

Figure 11

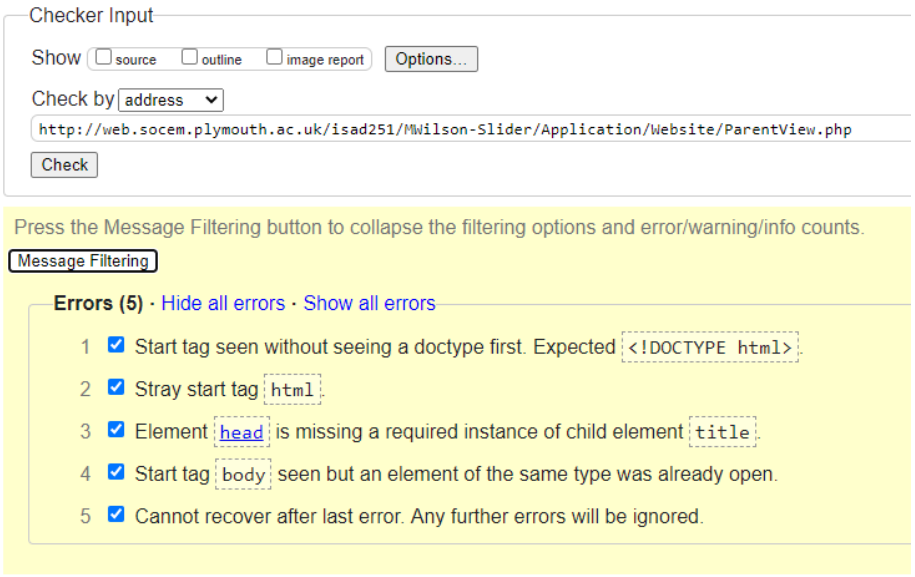


Figure 12 below: The results of the Child webpage

Figure 12

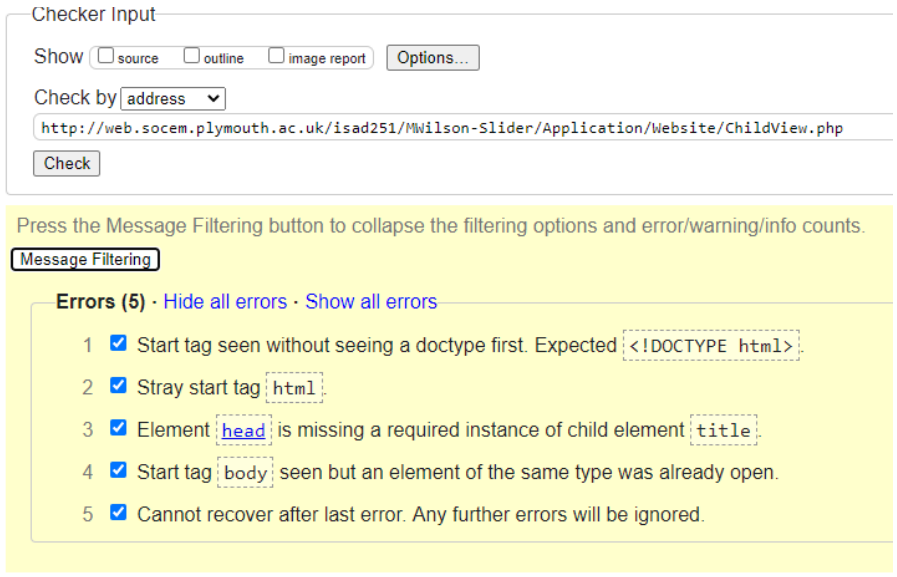


Figure 13 below: The results of webpage named Add new family

Figure 13

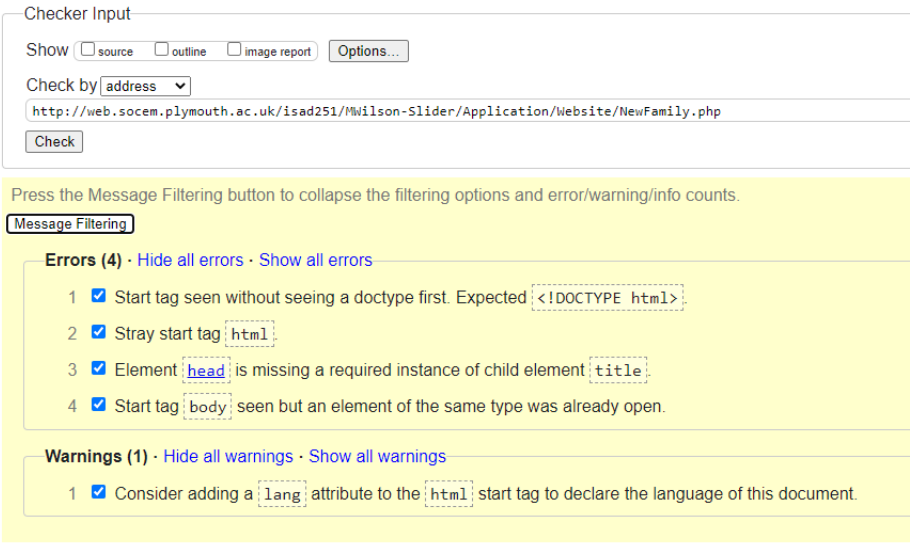


Figure 14 below: Results of Stylesheet

Figure 14

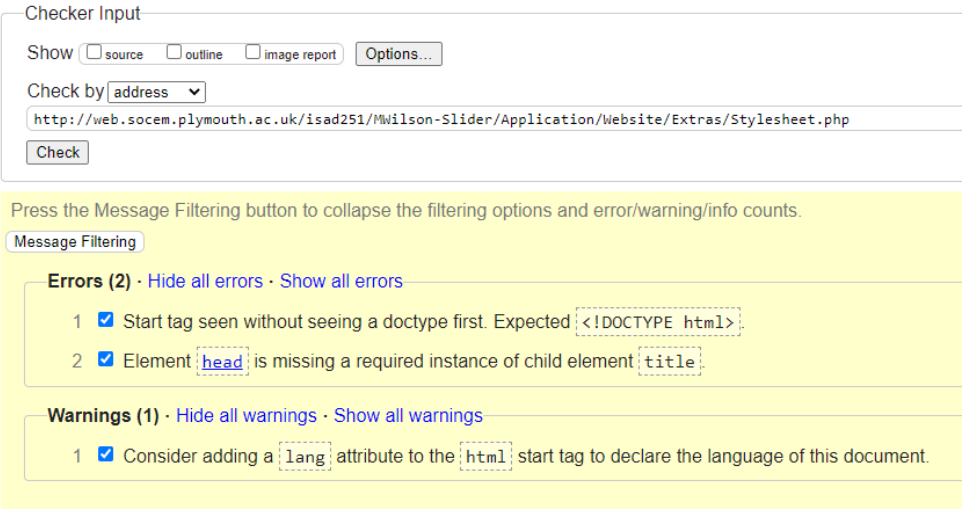
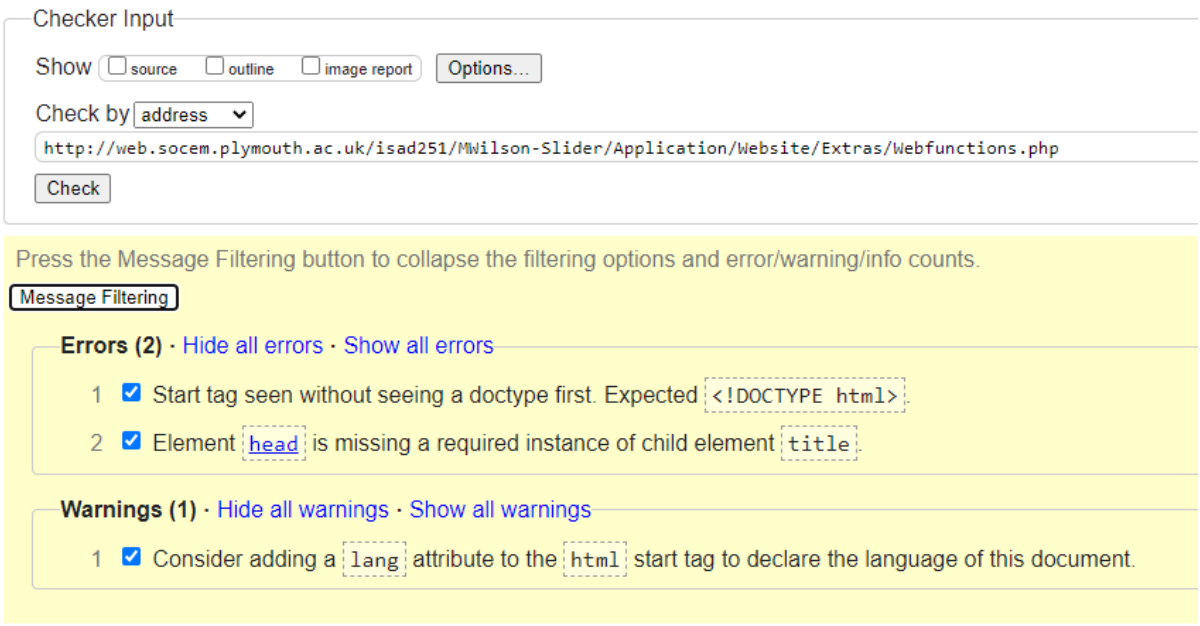


Figure 15 below: Results of Webfunctions

Figure 15



Changes I would make and why to develop the application

1. The home page of the application

I would of like to develop this further, by giving the application webpages a more aesthetically pleasing look, to make the website look more inviting. However, I would keep the colours simple, to ensure that it was disability friendly- for example it would be ok for colour-blind people. I would also try and create and image banner using transitions of the different features.

2. New User page

This page could have been improved by having parents and children make separate accounts instead of a parent having to create a child's login. At present, the webpage is already set up to create the parent by default and add the child, however this could be made to be more dynamic where a child may not have a parent, and needs to manage deadlines. Therefore, this would mean changing the initial page to allow a child user to be made without a parent – however the implementation would need to be considered due to adding additional people and safety.

This page also has another issue that I would of like to fix, when details are inputted to create an account, at the moment, if that account user name already exists, they are rejected on the first attempt. But when the user inputs a unique name not in the database it works? This at the moment is creating an alert box's appear multiple times often saying that the user exists then finally saying that the log in has been made.

3. Login page

This page works quite well, but I would of like to add a account revocery option, by using an email-based password rest.

4. User pages

The parent and child pages both share the same problems and solutions. At the moment when a user restarts the page, the website breaks as the session variables have been deleted, this can be fixed by using AJAX and I would have liked to produce a website incorporates this.

However, as I have not AJAX, when the user adds, edit's, or delete's an appointment/deadline the user currently has to log out and relog in. This has been used to overcome this issue with the page loading and the mentioned issues above where the session variables are deleted, using AJAX will resolve this.

Demo/Peer Review Feedback

Feedback was given to me via a scheduled appointment on the 4th of August 2020: 12:15pm with Dr. Shirley Atkinson in an online review demo appointment. And the following issues were highlighted. All comments below are spoken by Dr Atkinson at the time of the appointment.

Part 1. Feedback given

1. “Sort out file structure of the API”.
2. “Rename “Homepage.php” to “index.php”, as the default of a server looks for “index.php” unless it has been programmed otherwise”.
3. “The login page uses GET instead of POST, this is a massive security risk as the login details are put into the URL, and the page currently refresh's so this is visible”.
4. “You’d solve the issue of the appointments not showing by using AJAX, but this is outside the scope of the system so don’t fuss with that”.
5. “Instead of two separate Varchars for the time and date use DateTime, but that’s something that again I wouldn’t fuss about, as long you can argue why you have done it this way”.
6. “There are no specific requirements for how the page looks, as long as it works as expected, but the select boxes, they would be huge if a user has been using the website for over a year, so maybe checkboxes might be the best way to deal with this”.
7. “The UML is not a proper UML but is more of a data flow diagram, I'd suggest that this get looked at, there is an example UML diagram on the DLE for you to look at”.
8. “And obviously the child page needs to be finished but remember I'm looking for quality more than quantity”.

Part 2. Changes made using feedback

1. The API has been put into separate folders, except the 3 that needed to be outside of their designated folders, due to the session.ID being different inside the folder.
2. “Homepage.php” was renamed to “index.php”.
3. GET was changed to POST and the login page no longer refreshes, so immediate login is achieved.

4. I investigated AJAX briefly to get my head around it, but this was not implemented.
5. Due to time restraints this was not changed, I did change both of the Varchars into DateTime, and managed to split it and display the data on the website, however adding and editing the data was not quite right, I spent 2 days on this without much success.
6. The drop box was removed, and the checkboxes were added on the right side of the appointment/deadline
7. UML diagram has been updated to be a working UML instead of a data flow diagram.
8. I finished the parent page fully, and implemented all the above before moving on to the child page ensuring that any repeated code was moved onto the "webfunctions.php" file as they can both be shared, the function to send form data however could not be moved into this file.

Part 3. Suggested changes I would make if I had more time

I would implement AJAX so that when the page gets refreshed or data gets edited then the website automatically shows this without breaking anything. I would like figure out the DateTime submission as that was the only part that I could not figure out, and in doing this implement the code for it.

When I added the checkboxes and made it so the buttons could edit and delete this was only enabled when a checkbox has been checked. However, I encountered an issue, if one checkbox is checked then the buttons are enabled this would work, but if another checkbox is check then the buttons become disabled. This can be redone to fix this issue, but I would have liked to solve this issue or added a troubleshooting page/pop up. Finally, I would have liked to figure out how to pass the id of a session to the files in the API or investigate this further to develop a way to allow the same session to be picked up from different files.

URLS used.

API: https://github.com/DigSwine/ISAD251-Resit/tree/master/Application/API

: \\CENT-5-534.uopnet.plymouth.ac.uk\ISAD251\MWilson-Slider\Application\API

Hosted web pages

Webpage URL:http://web.socem.plymouth.ac.uk/isad251/MWilson-Slider/Application/Website/index.php

Server: \\CENT-5-534.uopnet.plymouth.ac.uk\ISAD251\MWilson-Slider

GitHub repository. https://github.com/DigSwine/ISAD251-Resit

Application: https://github.com/DigSwine/ISAD251-Resit/tree/master/Application

Documentation: https://github.com/DigSwine/ISAD251-Resit/tree/master/Documentation

SQL: https://github.com/DigSwine/ISAD251-Resit/tree/master/SQL

YouTube Video: https://youtu.be/m8K-2s13D3Y

Figure references.

Appendix

Appendix 1. DigSwine(2020). README.md [online] available at: https://github.com/DigSwine/ISAD251-Resit/blob/master/README.md (accessed on 20th August 2020).

Appendix 2. W3C® (MIT, ERCIM, KEIO, BEIHANG)(opensource). The w3C Markup Validation Service [Online]. Available at: https://validator.w3.org/ .