## Evaluation

### Fitness for Purpose

##### Requirement Specification Evaluation

From the results and evidence produced during the testing phase, I am able to conclude that my program meets all end-user and functional requirements…

|  |  |
| --- | --- |
| All end-user and functional requirements listed in Analysis | Completed |
| Use session variables to store the current account in use | Yes |
| Format the user interface based on the CSS in an external stylesheet / make each page similar in layout and design for visual clarity | Yes |
| Use a clear and legible font family, colour and size for readability | Yes |
| Make use of white space for further readability | Yes |
| Use a pleasing and consistent colour palette for visual clarity | Yes |
| Implement a intuitive navigation bar so the website is easy-to-navigate | Yes |
| Make the functions of buttons, links, etc clear so users know what to expect when they use them | Yes |
| Validate user inputs for account facilities | Yes |
| Show different user interfaces based on whether user is on mobile or desktop | Yes |
| Authenticate that users if logging in, exist within the external database | Yes |
| Authenticate that users if registering, the username they have inputted does not exist within the external database | Yes |
| Redirect to the appropriate error page when users who are not registered try to access the quiz | Yes |
| When the countdown times out or when the user submits the quiz, calculate the score and update the leaderboard by querying the external database | Yes |
| Sort the leaderboard details so that it is ordered by each user’s score and the time they completed it (both in descending order) and for each user, give them a numbered rank in ascending order | Yes |
| Logout users from the account they are using. If they are not using an account, do not perform the logout. | Yes |

##### Results of Testing

In my requirements testing, all of my test cases returned my expected or an acceptable result. The test cases also helped prove that my program was robust as my program did not crash even when given exceptional data. All exceptional data would inform the user that the data they had inputted was invalid. All exceptional data would send an error message to the user and was not accepted by the program. All extreme data would return the expected result - showing that logic errors would not be returned when using the upper and lower values of my program’s conditions. All normal data would return the expected result so the program is fit for purpose as it can handle potential end-user inputs correctly and without error. Thus, from the results of my testing, I can include that my solution meets all functional requirements.

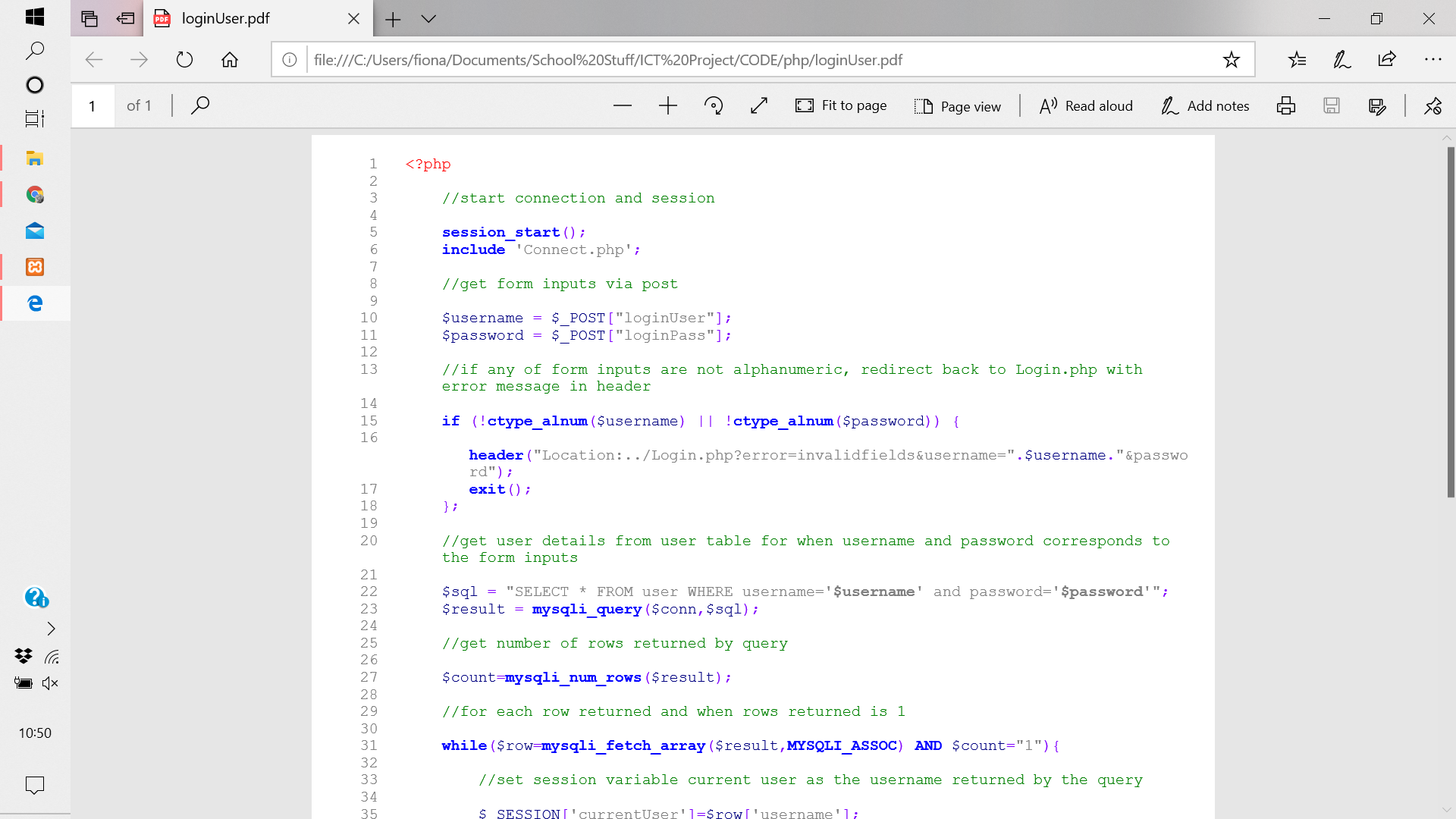
In my persona and test case testing, each tester was able to carry out the scenario with ease with the persona they had taken on. This proved to me that end-users would be able to use the website without any problems or issues, showing that my solution is usable for end-users. The answers I got from my user survey that I gave to all my testers proved that my solution met all end-user requirements as I got the expected result from all my responses.

### Maintainability

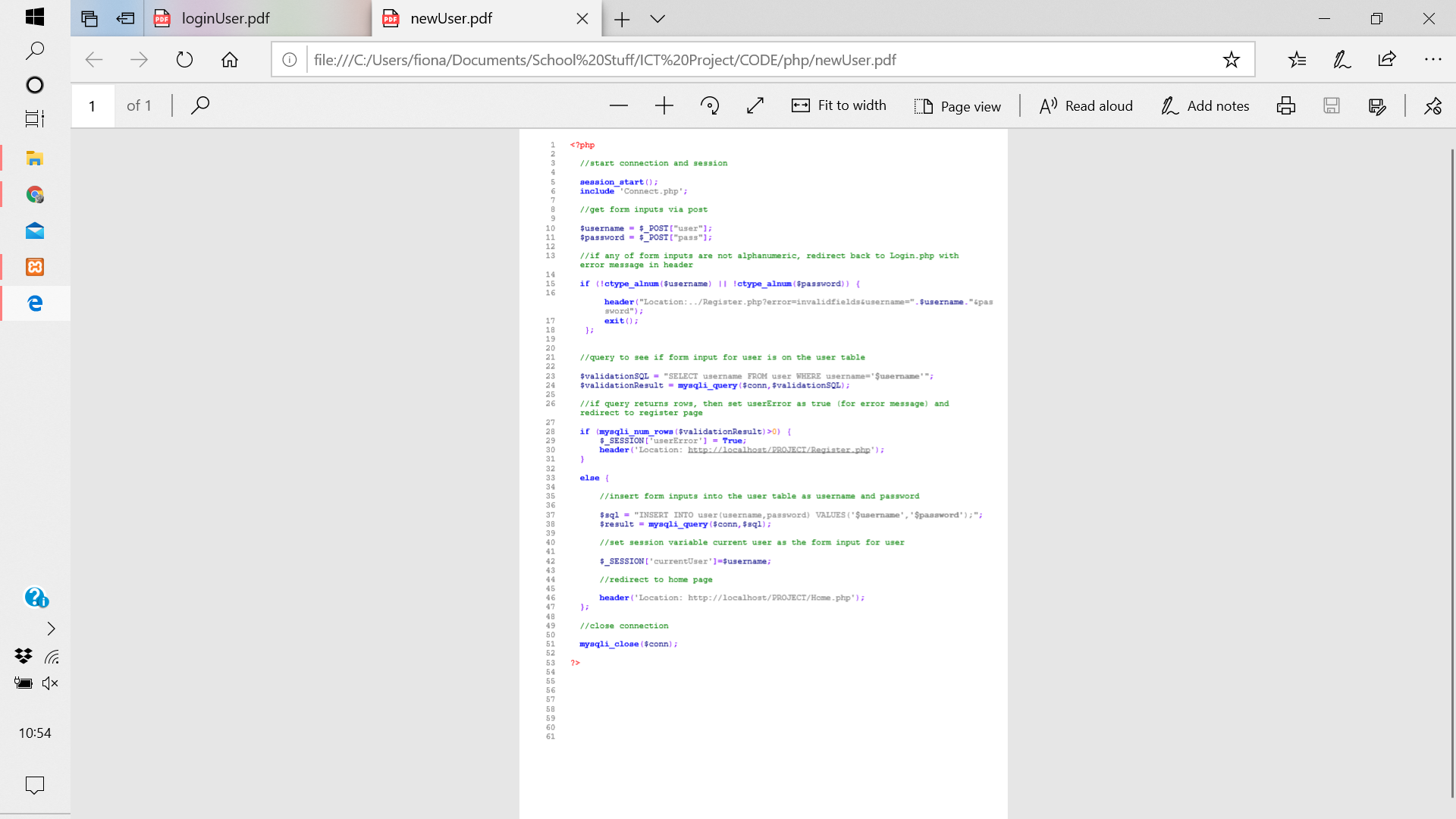
I believe that my project is maintainable for adaptive, corrective and perfective maintenance.

To ensure that my code was readable, I made use of meaningful identifiers for my variables and functions, indention, white space and internal commentary.

I used meaningful identifiers majorly throughout my code. Most of my functions’ and variables’ purposes were described in its name. Variables were also named after what kind of data they would contain when applicable. An example of this is shown below:

Here, I made sure that both the variables for the PHP code and the form were both meaningful and readable. 

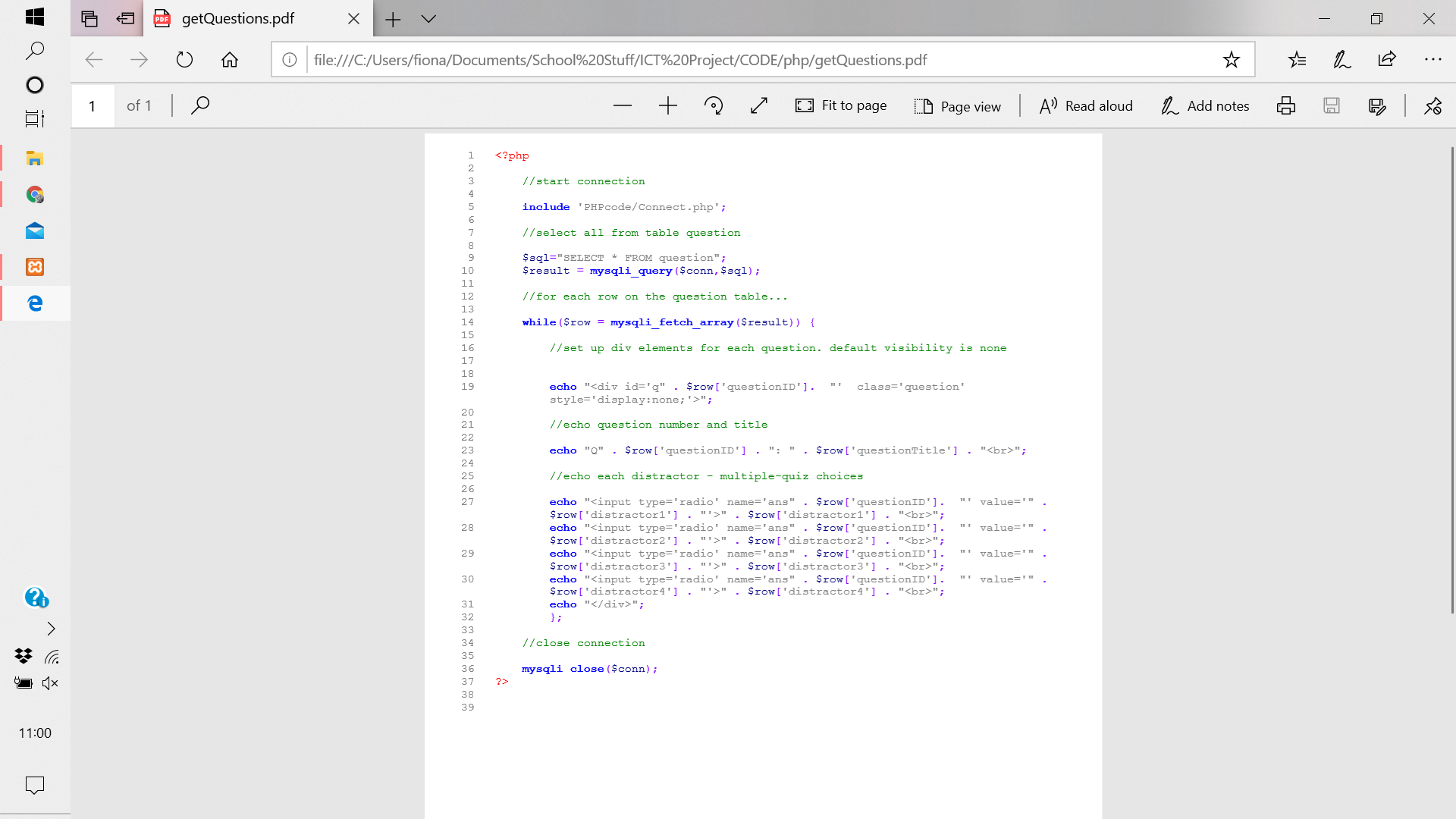
I made sure to use indentation and white space when appropriate in my code so that the overall structure of the code would become clearer. An example of this is shown below:

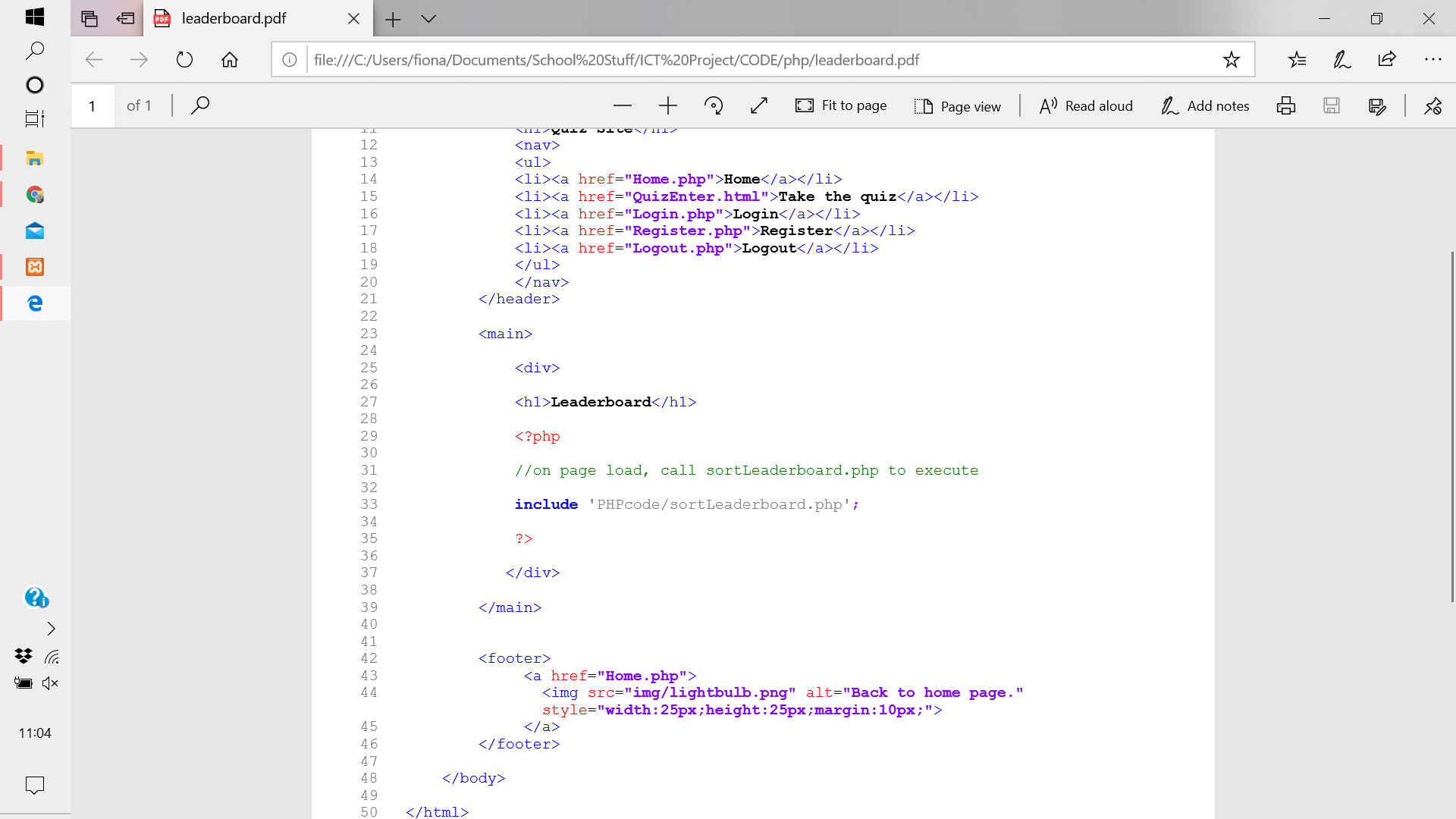


Here, I put white space between each piece of commentary and indented code when there was a loop/conditional statement.

I used internal commentary throughout my code so that my code could be translated and explained to anyone with access to my code. An example of this is shown below:

Here, I have described each step of the code so anyone looking at the code can get an idea on what I am trying to achieve.

I also integrated modularity in my code when possible. I tried to minimise the amount of PHP code on each page by making use of the ‘include’ and ‘require’ functions. These functions also helped the efficiency of my code as it would save me rewriting some pieces of code. An example of this is shown below:



Here, I have used an include statement to call external PHP code to be executed. This will make my code more maintainable as I have broken down my solution into smaller, more workable parts.

I have used an external stylesheet too so that my webpages are not only consistent in appearance for visual clarity, but so that only one change will have to be made to the stylesheet to change the appearance of the page instead of multiple changes if I used an internal/inline styles - making it less repetitive for anyone else working on the code to change the style of the website.

However, there were some places I could improve the maintainability of my code. For example, although I had explained what steps of code would do in each line of commentary, I would assign some unmeaningful identifiers to variables/functions. Instead of relying on internal commentary to explain my variables, I should have made the variables’ function clear in its name. I have implemented conditional statements, arrays and loops so that my program is primarily efficient but my code wasn’t wholly efficient too as I could have made use of loops more in places to minimise repetition.

### Robustness

I believe that my project is robust.

Through my testing, I have identified places in which I can implement error messages/pages to the user so they are aware that they have given an invalid input. An example of this is shown below:

Here, I have used a session variable to store if there has been an error in my PHP code so I can display an error message to the user so they know what they have done wrong.

So my project is able to cope with incorrect input, I have implemented validation when applicable. An example of this is shown below:

