|  |  |
| --- | --- |
| **CM2010 Software design and development** | |
|  | |
| **Student Name:** | Owen Lee Wei Hern |
| **Date Submitted:** | 24th December 2022 |
| **Degree Title:** | Computer Science |
| **Local Institution:** | Singapore Institute of Management |
| **Student ID:** | 220218799 |

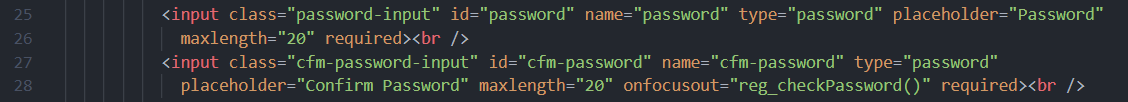
**Part 3**

The example project I will be looking at is my end semester Agile Software Projects Coursework. In this project, we planned and developed a charity awareness portal using Express.JS.

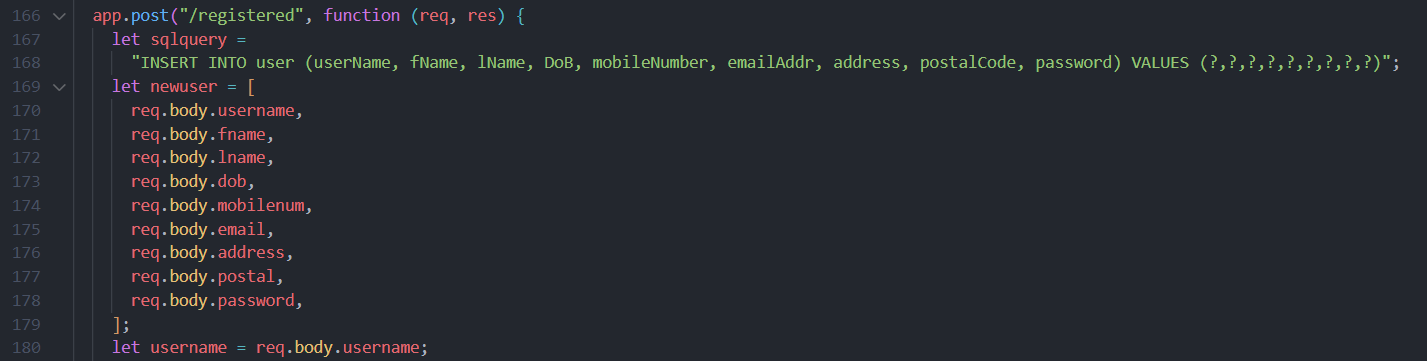
**Secure Programming Recommendation 1: Validating Inputs**

I found is that the user’s password is saved as plaintext upon registering. This is very dangerous as if the database were to be compromised, user information and passwords would be leaked to malicious targets.

As can be seen below, although there is a function to validate that the passwords in the password and confirm fields are the same,



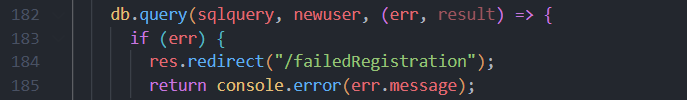
When the inputs are being registered as a new user into the SQL database, no further validation is done.



Sensitive fields such as passwords could be hashed when they are being stored. For example, the MD5 algorithm exists as a npm package and can be used to hash passwords before storing.

**Secure Programming Recommendation 2: Sending data back judiciously**

Another issue I found in this project was that error logs were left in the console and could be printed out. Examples of this can be found in the routes in main.js. While it is good to use error logs to debug issues during development, leaving such errors leave security implications as in this particular case, the error message is linked to failing to register.



A solution for this is to always scrub production code that will be live or remove such statements when development on this feature is complete. This is done as not only will it be cleaner as clients would not run into console error messages, but error logs also can be manipulated by malicious users to gain information about our program that they should not have.

**Secure Programming Recommendation 3: Design program for security**

Through looking at some of my javascript functions, simplicity can be adopted in them, so that in the event there are errors, they will be easier to de-bug.

For instance, the checkFilled() function is responsible for making sure inputs are filled before a user can submit a form.



As can be seen above, this function checks for both forms with a “form” tag name and a ‘div’ tag name. Splitting these into 2 separate functions would have been much simpler, as even though they are fundamentally similar, from the if-conditions we can see how they work is different and it may have been better to split them apart.