Week 5 Commentary

Show how you have made the form safe (200 words)

The form has been made safe by utilising the proper methods of form validation. One way in which this has been done is by using a check to ensure that the user is sure that their password they have entered whilst registering for an account is what they think they have entered. This has been accomplished by creating both a password and a confirm password, and when the form is actioned (ran) a function is run that checks both text boxes have been entered with the same string, ensuring the integrity of the site's passwords.

Another method I have utilised to make sure that the form is safe is that. I have used a feature from HTML5, which disallows the form from being run unless all fields have been filled in. this is done by adding the word 'required' where the inout box is created in the form's HTML code:

```
<input type="password" class="formcontrol" name="confirm" id="confirm" placeholder="Confirm y
our Password" minlength="8" required/>
```

The word for this can be seen at the end of the code, just before the closing tag.

Highlight 4 security features that add to your site or are weak within your site. (50 each)

A security feature that ensures the security of my site is the passwords have been hashed with MD5, so that if the database is ever compromised in any way then the passwords of the users of the website are still safe. If this could be improved upon, I would change the hashing algorithm to a newer and further secure one, such as SHA-1, or SHA-256.

Another security aspect is the whilst the server settings for the website to connect to the database are stored in a text file on the server, as they are not directly included in the web page that the user sees (and instead in a separate folder called model) this means that there is no way for a would be hacker to obtain the database username and password.

At time of writing however, I do not have the Google reCAPTCHA API in use on the forms on the web site. This would be a good idea to implement at some point, as it would guarantee that bots and crawlers cannot create accounts, as the API uses a simple but effective method to ensure that the person on the website is indeed, a human.

Another potential vulnerability is the that the database table and field names are what they say they are. Whilst this may not seem like a condemnation or potential vulnerability, then let me elaborate. By giving the `user` table the name 'user' this means that if a hacker tried an SQL Injection, and didn't know the actual name of the table, they could make the (correct assumption) that the table is called something easily guessable, such as user, which it is. From this they could then perform and SQL Injection and extrapolate some private user information, such as their email address and phone number.