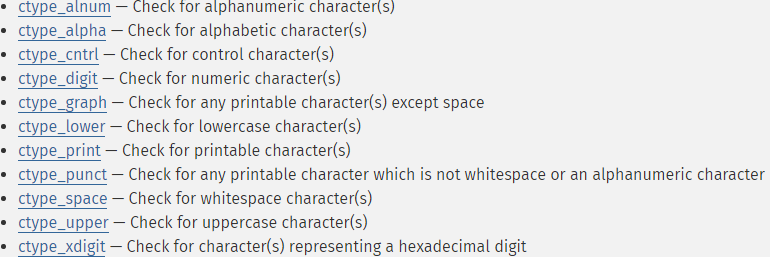
## Security measures

### General

Any type of variable that was used in a query was checked via three checks (PHP syntax). The examples are in the form of screenshots of the official PHP Manual ([link](http://php.net/manual/en/)) as of 12/12/16:

1. 
2. 
3. 

**Explanation:** The first comparator checks if value A is equal to value B. In the case of a textbox value for example, value A would be the textbox’s text and value B would be an empty string (“”). This is the initial test if there is text in the textbox.

The second comparator checks if value A and value B are of the same type. Using the example of a textbox text again, value A would be the text and value B would be an empty string (“”). This time PHP would compare the types, rather than the values of the two variables.

The third test is a selection of the **ctype** check relevant to the value type. For example, if we are expecting a number in the textbox, we would use **ctype\_digit().** This is an inbuilt function that returns false if the value is not the type expected.

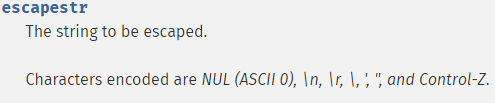
During the different stages of the project various combinations of these security measures were implemented for all values.

### SQL queries

Whenever we generate dynamic sql queries within the project, besides implementing the above described security measures, we also include an extra inbuilt security measure:



This inbuilt statement escapes any special characters found in the value. This is an important step to prevent SQL injection. Here’s an example of the values escaped by the function taken from the PHP Manual (find link at the top of this document):



One of the key characters here is the single quotation mark. This is a character that is crucial when attempting to execute a SQL injection attack. A basic explanation for the purpose of this character is that it completes the SQL query in question and the attacker can then attach their own query. For example in a “GET” usage in PHP the URL could look like:

http://www.url.com/getme.php?index=1

The SQL query attached would go something like this:

$sql = “SELECT \* FROM table WHERE i=’” . $\_GET[“index”].”’”;

As you can see from the example above, the single quotation marks surround the value PHP gets from the URL. Without using a character escape function an attacker could try the following:

http://www.url.com/getme.php?index=’SELECT \* FROM x

Parsing the value in the “GET” through the escape function will change the single quotation mark from such into: &#39. When displaying the character as a value, it will still show up as a quotation mark, while protecting the website from malicious attempts.