Working with the MySQL connection object

The connection object is shared between various callbacks in `app.js`. These callbacks run asynchronously, in separate threads. If the connection object is used incorrectly, you may run into errors caused by thread synchronization problems, such as race conditions.

For example, if you run into the error

`"Error: Cannot enqueue Query after invoking quit."`

it means you've closed the connection with `connection.end()`, but you are still trying to use the connection object somewhere to run `connection.query()`.

However, you should avoid all these problems if you follow these guidelines:

- Make the `connection` object global - declare it at the top of `app.js`, and not in any of the callbacks. Initialize it to null.
- Create the connection object by calling `connection = mysql.createConnection()` as soon as you have the user's credentials. Never re-create this object again.
- Call `connect()` method on the `connection` object as soon as you have created it. Never call `connect()` again.
- Never call `connection,end()` after any of the queries that you run.

This should run without any issues.

However, if you want to be completely sure, you can also create a button "Close DB connection" in the UI. The route for this button - i.e. the `app.get()` callback in `app.js` - would simply call `connection.end()`.

Do not press this button unless you see some sort of weird database error indicating too many open connections.

Just remember: if you click on "Close DB connection", you'll need to re-connect to the database. The easiest way to do this is by reloading the web app, and re-entering the DB connection details (username/password/DB name), which will force the server to re-create the connection, and re-connect.