Assignment: Automobile Database CRUD

In this assignment you will build a web based application to track data about automobiles. We will only allow logged in users to track automobiles.

Resources

There are several resources you might find useful:

- Recorded lectures, sample code and chapters from www.wa4e.com:
  - Understanding CRUD
- The sample code from the CRUD lecture is a simple working CRUD application.

  http://www.wa4e.com/code/crud.zip

General Specifications

Here are some general specifications for this assignment:

- Your name must be in the title tag of the HTML for all of the pages for this assignment.
- All data that comes from the users must be properly escaped using the **htmlentities()** function in PHP. You do not need to escape text that is generated by your program.
- You must follow the POST-Redirect-GET pattern for all POST requests. This means when your program receives and processes a POST request, it must not generate any HTML as the HTTP response to that request. It must use the "header('Location: ...');" function and "return" to send the location header and redirect the browser to the same or a different page.
- Please do not use HTML5 in-browser data validation (i.e. type="number") for the fields in this assignment as we want to make sure you can properly do server side data validation. And in general, even when you do client-side data validation, you should still validate data on the server in case the user is using a non-HTML5 browser.

Sample Implementation

You can experiment with a reference implementation at:

http://www.wa4e.com/solutions/autoscrud

Using the Autograder

This assignment will be automatically graded and so your web server will need an Internet-accessible URL so you can submit it for autograding. To achieve this you will need to install and use a piece of software called "ngrok". Instructions for installing and using ngrok are available at:

http://www.wa4e.com/ngrok.php

Please see the process for handing in the assignment at the end of this document.

Important: The autograder will demand that your name is in the <title> tag in the head are of your document. If the autograder does not find your name, it will run all the tests but will not treat the grade as official.

Creating Automobile table

You can reuse or adapt a table from a previous assignment. This assignment will need a table as follows:

```
CREATE TABLE autos (
    autos_id INTEGER NOT NULL KEY AUTO_INCREMENT,
    make VARCHAR(255),
    model VARCHAR(255),
    year INTEGER,
    mileage INTEGER
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Protecting the add.php and edit.php

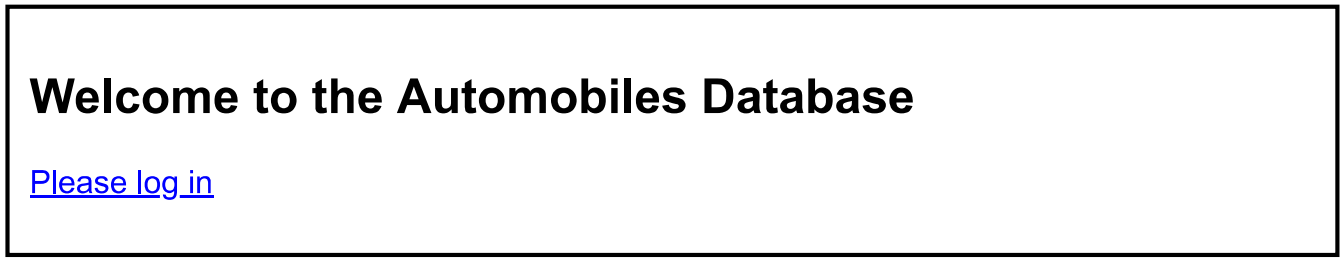
In order to protect the database from being modified without the user properly logging in, the **add.php** and **edit.php** must first check the session to see if the user's name is set and if the user's name is not set in the session the they must stop immediately using the PHP die() function:

```
die("ACCESS DENIED");
```

To test, navigate to **add.php** manually without logging in - it should fail with "ACCESS DENIED".

Log in

If the user is not logged in, they will be presented a screen with a welcome and a link to **login.php** - they should not see the table of data.



The autograder will log in to your program with the following account and password:

```
Account: umsi@umich.edu
Password: php123
```

The login screen needs to have some error checking on its input data. If either the name or the password field is blank, you should display a message of the form:

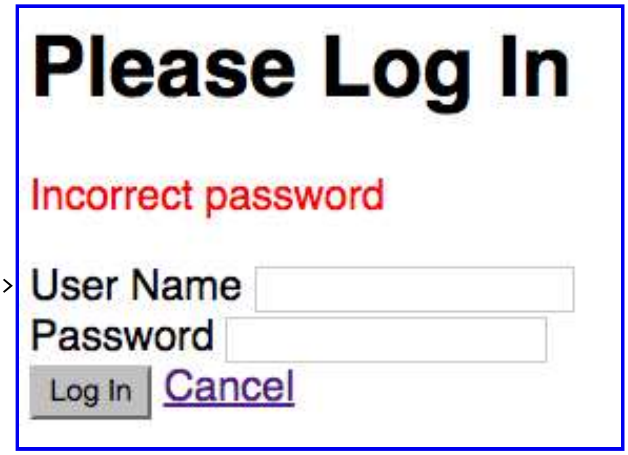
```
User name and password are required
```

If the password is non-blank and incorrect, you should put up a message of the form:

```
Incorrect password
```

Note: Please name your form fields in **login.php** exactly as follows for autograding:

```
User Name <input type="text" name="email"><br/>
Password <input type="text" name="pass"><br/>
```



Automobile Database Main List

Once the user is logged in, they should should be redirected to **index.php** where they will be shown see a list of the automobiles in the database in a table similar to the following:

Welcome to the Automobiles Database

Make	Model	Year	Mileage	Action
Thistly	Vowelising	80	8	Edit / Delete
Harvests	Thinkingly	83	52	Edit / Delete
Eigentones	Upcurled	52	39	Edit / Delete
Gainly	Czarevna	44	30	Edit / Delete

Add New Entry

Logout

If there are no rows in the table, do not print out the table, but simply print out **"No rows found"**.

There should also be options to **Add a New Entry** and **Log Out** presented after the table.

If the **Logout link** is pressed the user should be sent to the **logout.php** page which clears session variables and redirects back to **index.php**.

Adding new Records

When the user asks to add a new record, they should be presented with a screen that allows them to append a new automobile. Each automobile will have the following fields:

- Make
- Model
- Year - must be an integer - use is_numeric() to validate
- Mileage - must be an integer - use is_numeric() to validate

Important: Make sure to name the fields in your forms using the lower case version of the field names so that the autograder can work:

```
<input type="text" name="make">
```

When processing an incoming POST, data must be validated. All fields are required, if there is a missing (i.e. blank) field, issue a message like:

```
All fields are required
```

If the user enters a non-numeric field, you should issue a message like:

```
Year must be an integer
```

If there are any errors on the input, do not add the record to the stored data. Redirect the user back to the **add.php** script and display the error message "flash style".

```
if ( ... at least one of the fields is empty ... ) {
    $_SESSION['error'] = "All fields are required";
    header("Location: add.php");
    return;
}

...

if ( isset($_SESSION['error']) ) {
    echo('<p style="color: red;">'.htmlentities($_SESSION['error'])."</p>\n");
    unset($_SESSION['error']);
}
```

Note that only one of the error messages need to come out regardless of how many errors the user makes in their input data. Once you detect one error in the input data, you can stop checking for further errors.

If the data validates and the add is successful, redirect to **index.php** with a successful flash message:

```
Record added
```

Editing Records and Validation Errors

When you edit a record, the prior data must be shown and properly escaped. All of the data validation must be performed on the edit data as required in the **add.php**. Make sure to include the "id" parameter (you may name this variable differently) on the redirect statement in the **edit.php** when an error is detected:

```
if ( ... a field is missing ... ) {
    $_SESSION['error'] = "All fields are required";
    header("Location: edit.php?autos_id=".$_REQUEST['id']);
    return;
}
```

If the data validates and the edit is successful, redirect to **index.php** with a successful flash message:

```
Record edited
```

Deleteing Records

When the user selects the "Delete" link from the list of Automobiles you should put up a form with "Delete" and "Cancel" options.

If the "Delete" button is pressed, the record is deleted and the user is redirected to **index.php** with a success message:

```
Record deleted
```

What To Hand In

This assignment will be autograded by a link that you will be provided with in the LMS system. When you launch the autograder, it will prompt for a web-accessible URL where it can access your web application. Since your application is running on localhost, you will need the Ngrok application installed.
