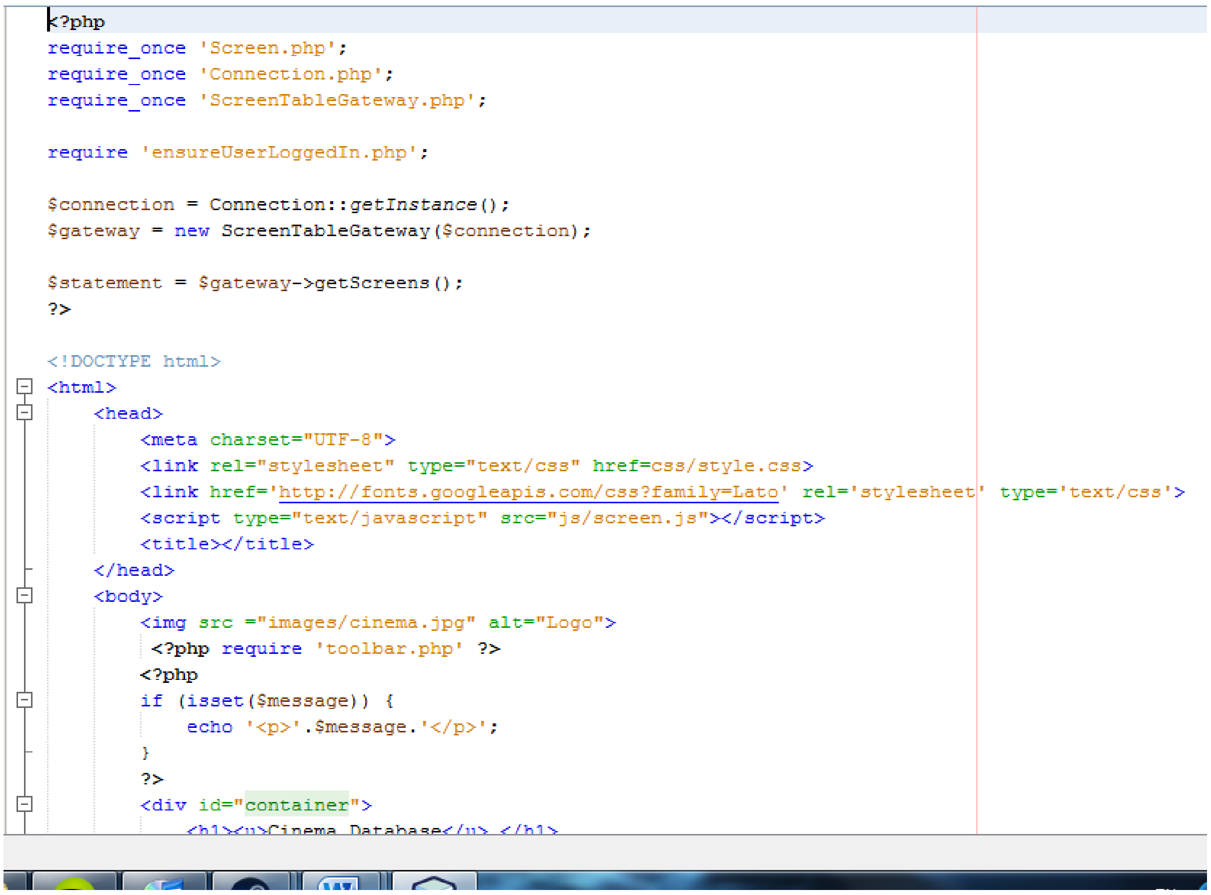
CA 2 Web Programming PHP Database

N00132599
Sam Brooks

Overview

For this assignment I was required to implement a HTML/CSS/JavaScript/PHP application that allows a user to access and view the contents of a database and its subsequent tables in MySQL. Other requirements were to include options to insert new data, update current information in the database, and to delete information/ fields in the database table.

Home



```
<?php
require_once 'Screen.php';
require_once 'Connection.php';
require_once 'ScreenTableGateway.php';

require 'ensureUserLoggedIn.php';

$connection = Connection::getInstance();
$gateway = new ScreenTableGateway($connection);

$statement = $gateway->getScreens();
?>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <link rel="stylesheet" type="text/css" href=css/style.css>
        <link href='http://fonts.googleapis.com/css?family=Lato' rel='stylesheet' type='text/css'>
        <script type="text/javascript" src="js/screen.js"></script>
        <title></title>
    </head>
    <body>
        <img src ="images/cinema.jpg" alt="Logo">
         <?php require 'toolbar.php' ?>
        <?php
        if (isset($message)) {
            echo '<p>'.$message.'</p>';
        }
        ?>
        <div id="container">
            <h1><u>Cinema Database</u> </h1>
```

I'll begin with the home page, as that is the start page for the website which displays the Screen table from the database, and the login button. As seen at the top, the require_once function includes other php pages you have created within the same php project. It includes them and fetches the information from them once and does not have to include them several times.

require 'ensureUserLoggedIn.php';
$connection = Connection::getInstance();
$gateway = new ScreenTableGateway($connection);

```
$statement = $gateway->getScreens();
```

The next line of code includes the require function that will constantly check if the user on the database is logged into the database at any time in order to use access the information stored in the tables/database.
An instance of the connection is retrieved to connect the webpage to the database on Daneel, and the php class ScreenTableGateway that holds the SQL code to access the database. The statement is to use the gateway to fetch and display the variables shown in the screen table using the getScreens() function.
Below that is the design of the HTML page to display within a browser.

```
<link rel="stylesheet" type="text/css" href=css/style.css>

<link href='http://fonts.googleapis.com/css?family=Lato' rel='stylesheet' type='text/css'>

<script type="text/javascript" src="js/screen.js"></script>

<title></title>

</head>

<body>

<img src ="images/cinema.jpg" alt="Logo">

<?php require 'toolbar.php' ?>

<?php
```

The code above links the stylesheets made within the same project to style the individual HTML pages using cascading style sheets. The next link reference is to use different fonts that are not already pre programmed into html.
In the body of the html page is an image relating to my case study which is a Cinema (hence the screen table). The image sits at the top of the page with the screen table sat underneath, and the link buttons there also.

```
<div id="container">
<h1><u>Cinema Database</u> </h1>
<table border="1" width="75%">
<thead>
<tr>
<th>Screen</th>
<th>Fire Exits</th>
<th>Number of Seats</th>
<th>Projector Type</th>
<th>Options</th>
</tr>
</thead>
<tbody>
```

The table is wrapped inside a container div so that the container can be styled with background colours, margins, etc. The table headers are produced, while php code fetches the table from the database and displays it using the following code.

```
<?php
$row = $statement->fetch(PDO::FETCH_ASSOC);
while ($row) {


echo '<td>' . $row['screenNumber'] . '</td>';
echo '<td>' . $row['noOfFireExits'] . '</td>';
echo '<td>' . $row['noOfSeats'] . '</td>';
echo '<td>' . $row['projectorType'] . '</td>';
echo '<td>'
. '<a href="viewScreen.php?id='.$row['screenID'].'">View</a> '
. '<a href="editScreenForm.php?id='.$row['screenID'].'">Edit</a> '
. '<a class="deleteScreen" href="deleteScreen.php?id='.$row['screenID'].'">Delete</a> '
. '</td>';
echo '</tr>';

$row = $statement->fetch(PDO::FETCH_ASSOC);
}
?>
```

(PDO::FETCH_ASSOC) – this returns an array indexed by column name as returned in your result set, in this instance, the screen table. The a href tags reference the view, edit and delete screen pages and contain links to those webpages that are displayed next to the table.

ScreenTableGateway

This page access the database so information can be altered, deleted and added.

```
public function getScreens() {
//executes query to get screens
$sqlQuery = "SELECT * FROM screen";

$statement = $this->connection->prepare($sqlQuery);
$status = $statement->execute();

if (!$status) {
die("Unable to retrieve screens");
}

return $statement;

}
```

The above code is SQL code that will access the database, using the public function getScreens() which will then retrieve the information in the screen table.
The SQL query will fetch all fields and information from the table using
- $sqlQuery = “SELECT * FROM screen”; -
Which will select all fields from the screen table and display them.

```
public function getScreenById($screenID) {
// execute a query to get the user with the specified id
$sqlQuery = "SELECT * FROM screen WHERE screenID = :screenID";

$statement = $this->connection->prepare($sqlQuery);
$params = array(
"screenID" => $screenID
);

$status = $statement->execute($params);

if (!$status) {
die("Could not retrieve user");
}

return $statement;
}
```

The above code prepares the query to gather information from the array using the parameter “screenID” in this case.

```
public function insertScreen($s, $f, $se, $pr) {
$sqlQuery = "INSERT INTO screen " .
"(screenNumber, noOfFireExits, noOfSeats, projectorType) " .
"VALUES (:screenNumber, :noOfFireExits, :noOfSeats, :projectorType)";

$statement = $this->connection->prepare($sqlQuery);
$params = array(
"screenNumber" => $s,
"noOfFireExits" => $f,
"noOfSeats" => $se,
"projectorType" => $pr
);

$status = $statement->execute($params);

if (!$status) {
die("Unable to insert screen");
}

$id = $this->connection->lastInsertId();
return $id;
}
```

The above code is used to insert a screen into the database. The letters ($s, $f, etc) are variables declared in the Screen.php class.

```
$sqlQuery = "INSERT INTO screen " .
```

This will perform a query which will insert any data input by the user into the ‘screen’ table. The values in the table are declared and then any data entered to create a new screen are entered in the values of the fields in the table, in this case, screenNumber, noOfFireExits, noOfSeats, and projectorType. If for some reason the query cannot be executed the webpage will display the message “Unable to insert screen”.

```
public function deleteScreen($id) {
$sqlQuery = "DELETE FROM screen WHERE screenID = :screenID";

$statement = $this->connection->prepare($sqlQuery);
$params = array(
"screenID" => $id
);

$status = $statement->execute($params);

if (!$status) {
die("Could not delete screen");
}

return ($statement->rowCount() == 1);
}
```

The code above is much the same as the aforementioned functions instead this is to delete fields from the table.

```
public function updateScreen($id, $s, $f, $se, $pr) {
$sqlQuery =
"UPDATE screen SET " .
"screenNumber = :screenNumber, " .
"noOfFireExits = :noOfFireExits, " .
"noOfSeats = :noOfSeats, " .
"projectorType = :projectorType " .
"WHERE screenID = :screenID";

$statement = $this->connection->prepare($sqlQuery);
$params = array(
"screenID" => $id,
"screenNumber" => $s,
"noOfFireExits" => $f,
"noOfSeats" => $se,
"projectorType" => $pr,
);

$status = $statement->execute($params);

return ($statement->rowCount() == 1);
```

This function is to update/alter/edit data in the table, and is a little more complex than the previous one.

```
$sqlQuery =
"UPDATE screen SET " .
"screenNumber = :screenNumber, " .
"noOfFireExits = :noOfFireExits, " .
"noOfSeats = :noOfSeats, " .
"projectorType = :projectorType " .
"WHERE screenID = :screenID";
```

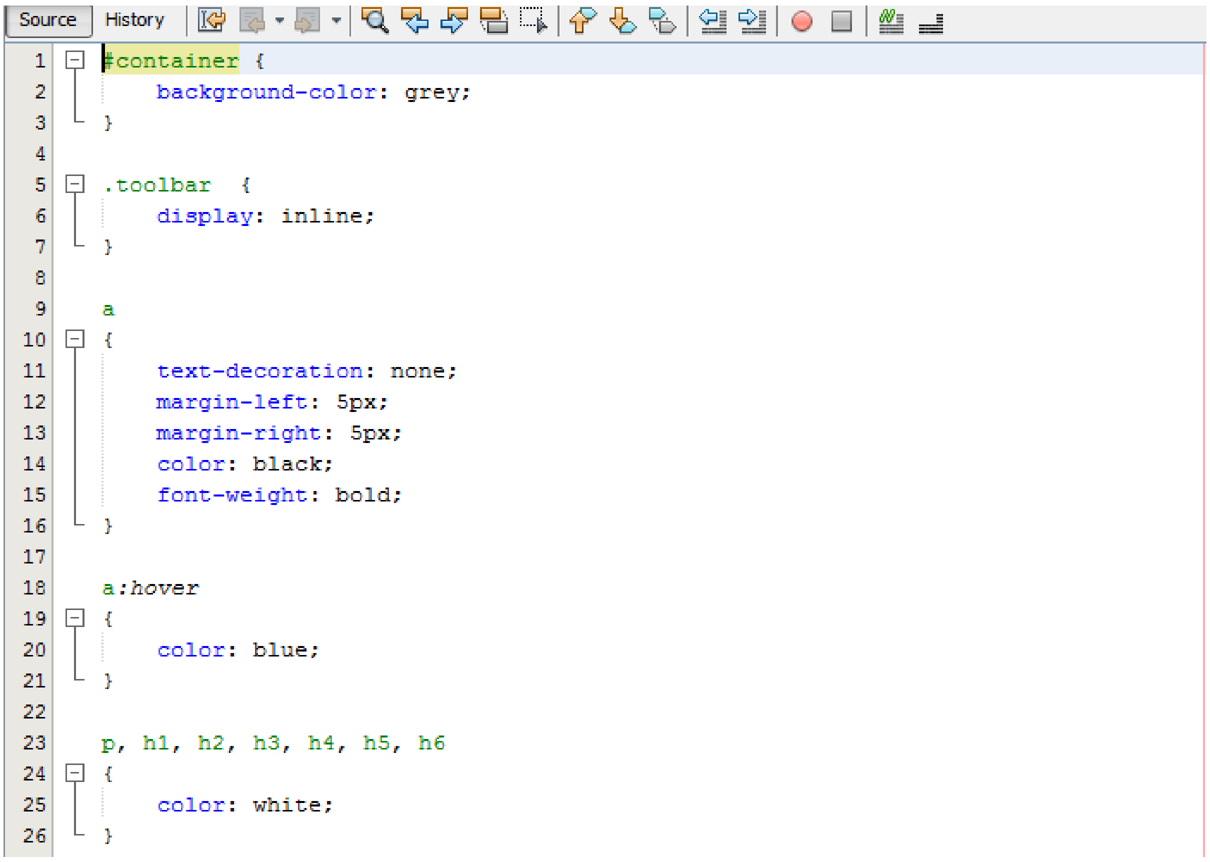
The SET is the fields in the table, and it locates the fields starting with screenID, displays them to the user and the user must input the new values and click save. A javascript message will pop up if the fields are left blank, and an alert box will be displayed to ask the user whether the data is correct.

Toolbar

```
<?php
$MySession_id = session_id();
if ($MySession_id == "") {
session_start();
}

if (isset($_SESSION['username'])) {
echo '<br><p class="toolbar"><a href="home.php">Home</a></p>';
echo '<p class="toolbar"><a href="logout.php">Logout</a></p><br>';
}
else {
echo '<br><p class="toolbar"<a href="index.php">Home</a></p>';
echo '<p class="toolbar"><a href="login.php">Login</a></p><br>';
}
```

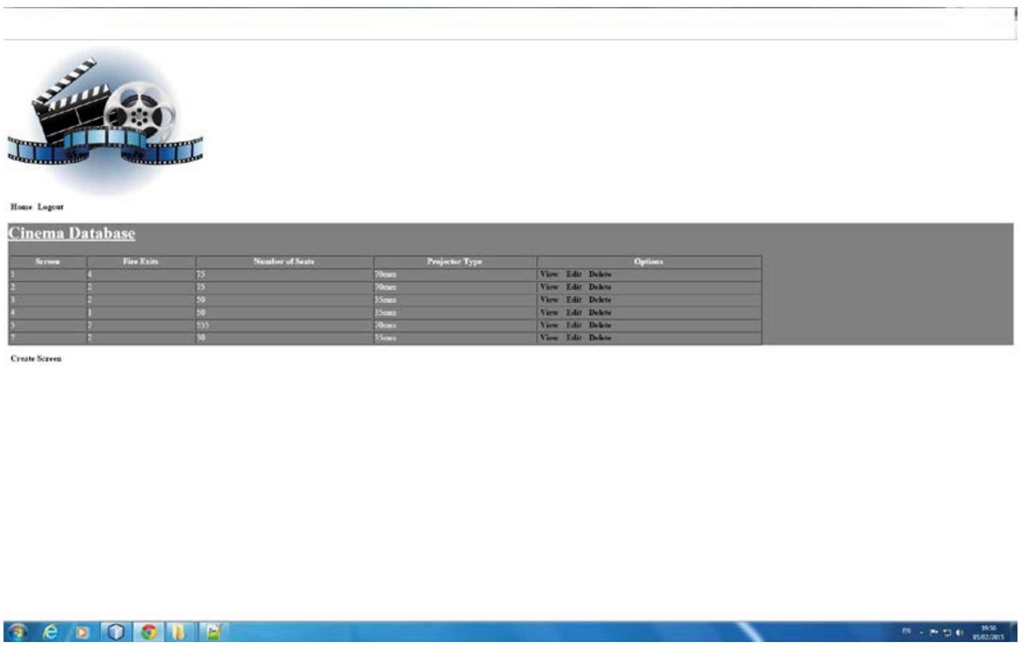
This php code snippet basically constructs a toolbar for users to navigate through the site once they have logged in and displays clickable buttons which is then retrieved using code to require the toolbar.php page into the home page and the index page.

Style.css



```css
#container {
    background-color: grey;
}

.toolbar  {
    display: inline;
}

a
{
    text-decoration: none;
    margin-left: 5px;
    margin-right: 5px;
    color: black;
    font-weight: bold;
}

a:hover
{
    color: blue;
}

p, h1, h2, h3, h4, h5, h6
{
    color: white;
}
```

The screenshot above is the cascading style sheet to style the container div used in the homepage and the subsequent form pages that adds a grey background and white coloured font to everything inside the container div. The end result looks like this:



Home Logout

Cinema Database

Screen	Fire Exits	Number of Seats	Projector Type	Options
1	4	75	70mm	View Edit Delete
2	2	75	70mm	View Edit Delete
3	2	50	35mm	View Edit Delete
4	1	50	35mm	View Edit Delete
5	2	555	70mm	View Edit Delete
7	2	30	35mm	View Edit Delete

Create Screen