dbconnect.php

<?php

define('DB\_SERVER', 'localhost');

define('DB\_USERNAME', 'webLogin');

define('DB\_PASSWORD', 'ThePasswordIsCarrot1');

define('DB\_NAME', 'games');

$conn = new mysqli(DB\_SERVER, DB\_USERNAME, DB\_PASSWORD, DB\_NAME);

if ($conn->connect\_error) {

    die("Connection failed: " . $conn->connect\_error);

}

include $\_SERVER['DOCUMENT\_ROOT'] . "/debugging.php";

?>

Admin.php

<?php

require\_once "dbinteraction.php";

$games = loadGamesFromDB();

$title;

$price;

$genre;

$platform;

$classification;

$acceptedFileTypes = array(".csv",".txt");

$file;

$goodFile;

function checkInputs() {

    if (!empty($\_GET["formID"])) {

        switch ($\_GET["formID"]) {

            case 1:

                if (!empty($\_GET["title"])) {

                    global $title;

                    $title = $\_GET["title"];

                }

                if (!empty($\_GET["price"])) {

                    global $price;

                    $price = $\_GET["price"];

                }

                if (!empty($\_GET["genre"])) {

                    global $genre;

                    $genre = $\_GET["genre"];

                }

                if (!empty($\_GET["platform"])) {

                    global $platform;

                    $platform = $\_GET["platform"];

                }

                if (!empty($\_GET["classification"])) {

                    global $classification;

                    $classification = $\_GET["classification"];

                }

                if (isset($title, $price, $genre, $platform, $classification)) {

                    $gameArray = array($title, $price, getGenreID($genre), getPlatformID($platform), getClassificationID($classification));

                    AddGame($gameArray);

                    header("Location: /admin.php");

                }

                break;

            case 3:

                if (!empty($\_GET["fileUpload"])) {

                    $fileinfo = pathinfo($\_GET["fileUpload"]);

                    global $file, $acceptedFileTypes;

                    foreach ($acceptedFileTypes as $key) {

                        if ("." . $fileinfo['extension'] == $key) {

                            $file = $\_GET["fileUpload"];

                            return;

                        }

                    }

                }

                break;

            default:

                header("Location: /admin");

            break;

        }

    }

}

function getAcceptedFileTypes(){

    global $acceptedFileTypes;

    return implode(",", $acceptedFileTypes);

}

function AddGame($game) {

    global $conn;

    $query = "INSERT INTO games (title, price, genre, platform, classification) VALUES (?,?,?,?,?);";

    if ($stmt = $conn->prepare($query)) {

        $stmt->bind\_param("sdiii", $game[0], $game[1], $game[2], $game[3], $game[4]);

        $result = $stmt->execute();

        echo $result;

    }

}

function refValues($arr){

    if (strnatcmp(phpversion(),'5.3') >= 0) //Reference is required for PHP 5.3+

    {

        $refs = array();

        foreach($arr as $key => $value)

            $refs[$key] = &$arr[$key];

        return $refs;

    }

    return $arr;

}

// FOR LATER IMPROVEMENT: https://dev.mysql.com/doc/refman/5.7/en/load-data.html

function bulkUpload($file, mysqli $conn) {

    $games = array();

    $directory = fopen($file, "r") or die("Unable to open file.");

    fgets($directory);

    // $columns = dbGetColumns($conn);

    $query = "INSERT INTO games (title, price, genre, platform, classification) VALUES ";

    $params = "";

    $valueTemp = "(?,?,?,?,?), ";

    $paramsTemp = "sdiii";

    while(!feof($directory)) {

        $line = explode(",", fgets($directory));

        array\_push($games, $line[0],

            sprintf("%.2f", floatval($line[1])),

            getGenreID($line[2]),

            getPlatformID($line[3]),

            getClassificationID(trim($line[4]))

        );

        $query .= $valueTemp;

        $params .= $paramsTemp;

    }

    $query = rtrim($query,", ");

    $query .= ";";

    if ($stmt = $conn->prepare($query)) {

        array\_unshift($games, $params);

        call\_user\_func\_array(array($stmt, 'bind\_param'), refValues($games));

        $stmt->execute();

    }

}

function createTableRow($game) {

    return

    '<tr id=' . $game["id"] . '>' .

        "<td class=\"col1\"><input type=\"checkbox\" onchange=\"checkRow(this.parentElement.parentElement.id, this.checked);\"></td>" .

        '<td class=\"col2\"><div contenteditable name="title">' . $game["title"] . '</div></td>' .

        '<td class="col3" onclick="this.children[1].focus();">' .

            '<p style="margin:0;display:inline-block">$</p>' .

            '<div contenteditable name="price" style="display:inline-block;text-align:left;">' . $game["price"] . '</div>' .

        '</td>' .

        '<td class="col3"><div contenteditable name="genre">' . $game["genre"] . '</div></td>' .

        '<td class="col4"><div contenteditable name="platform">' . $game["platform"] . '</div></td>' .

        '<td class="col5"><div contenteditable name="classification">' . $game["classification"] . '</div></td>' .

        '<td class="col6"><button class="button" onclick="updateRow(this.parentElement.parentElement.id);">Update</button></td>' .

        '<td class="col6"><button class="button" onclick="deleteRow(this.parentElement.parentElement.id);">Delete</button></td>' .

    '</tr>';

}

function createTbody($games) {

    $string = "<tbody>";

    foreach ($games as $selected) {

        $string .= createTableRow($selected);

    }

    $string .= "</tbody>";

    return $string;

}

checkInputs();

if (isset($file)) {

    bulkUpload($file, $conn);

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="shortcut icon" href="">

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

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

    <title>Game Store</title>

</head>

<body>

    <?php include 'header.html'?>

    <div style="text-align:center;margin-top: 6em;">

        <div>

            <h2>Individual Upload</h2>

            <form method="get" id="gameForm">

                <input type="hidden" name="formID" value=1>

                <div style="display:inline-block">

                    <label for="gameTitle">Game Title: </label>

                    <input type="text" id="gameTitle" name="title" value="<?= (isset($title)) ? $title: ''?>"/>

                    <?php

                        if (!empty($\_GET) && !isset($title) && $\_GET["formID"] == 1) {

                            echo "<span style=\"color: red\">Please enter a game title.</span>";

                        }

                    ?>

                </div>

                <div style="display:inline-block">

                    <label for="price">Price: </label>

                    <input type="text" id="price" name="price" value="<?= (isset($price)) ? $price: ""?>"/>

                    <?php

                        if (!empty($\_GET) && !isset($price) && $\_GET["formID"] == 1) {

                            echo "<span style=\"color: red\">Please enter a valid price.</span>";

                        }

                    ?>

                </div>

                <!-- TODO: Change Genre, Platform and Classification to use list of distinct categories -->

                <!-- TODO: Add way to add new Genre, Platform and Classifications... -->

                <div style="display:inline-block">

                    <label for="genre">Genre: </label>

                    <select name="genre">

                        <option value="" selected disabled style="display:none"></option>

                    <?php

                        $platforms = dbGetGenre();

                        $string = "";

                        foreach ($platforms as $key) {

                            $string .= '<option value="' . $key["name"] . '">' . $key["name"] . '</option>';

                        }

                        echo $string;

                    ?>

                    </select>

                    <?php

                        if (!empty($\_GET) && !isset($genre) && $\_GET["formID"] == 1) {

                            echo "<span style=\"color: red\">Please enter a genre.</span>";

                        }

                    ?>

                </div>

                <div style="display:inline-block">

                    <label for="platform">Platform: </label>

                    <select name="platform">

                        <option value="" selected disabled style="display:none"></option>

                    <?php

                        $platforms = dbGetPlatforms();

                        $string = "";

                        foreach ($platforms as $key) {

                            $string .= '<option value="' . $key["name"] . '">' . $key["name"] . '</option>';

                        }

                        echo $string;

                    ?>

                    </select>

                    <?php

                        if (!empty($\_GET) && !isset($platform) && $\_GET["formID"] == 1) {

                            echo "<span style=\"color: red\">Please enter a valid platform.</span>";

                        }

                    ?>

                </div>

                <div style="display:inline-block">

                    <label for="classification">Classification: </label>

                    <select name="classification">

                        <option value="" selected disabled style="display:none"></option>

                    <?php

                        $platforms = dbGetClassification();

                        $string = "";

                        foreach ($platforms as $key) {

                            $string .= '<option value="' . $key["initial"] . '">' . $key["initial"] . '</option>';

                        }

                        echo $string;

                    ?>

                    </select>

                    <?php

                        if (!empty($\_GET) && !isset($platform) && $\_GET["formID"] == 1) {

                            echo "<span style=\"color: red\">Please enter a valid classification.</span>";

                        }

                    ?>

                </div>

                <br><br>

                <span>

                    <input type="reset" value="Clear" class="button" onclick="clearForm(this.form.id)"/>

                    <input type="submit" value="Add Game" class="button"/>

                </span>

                <br>

                <?php

                    if (!empty($\_GET) && isset($title, $price)) {

                        echo "<span style=\"color: green\">All fields valid.</span>";

                    }

                ?>

            </form>

        </div>

        <hr>

        <div style="text-align:center">

            <h2>Change Game Entry</h2>

            <label for="updateSearch">Search: </label><input type="text" id="updateSearch"/>

            <br><br>

            <div style="display:inline-block">

                <table>

                    <thead>

                        <tr>

                            <th scope="col" class="col1"></th>

                            <th scope="col" class="col2">Title</th>

                            <th scope="col" class="col3">Price</th>

                            <th scope="col" class="col3">Genre</th>

                            <th scope="col" class="col4">Platform</th>

                            <th scope="col" class="col5">Classification</th>

                            <th scope="col" class="col6"></th>

                            <th scope="col" class="col6"></th>

                        </tr>

                    </thead>

                    <?php echo createTbody($games)?>

                </table>

            </div>

            <br><br>

            <button class="button all" onclick="UpdateAll();" disabled>Update Selected</button> <!-- TODO: Have Javascript execute post/get on mass -->

            <button class="button all" onclick="DeleteAll();" disabled>Delete Selected</button> <!-- TODO: Have Javascript execute post/get on mass-->

        </div>

        <hr>

        <div style="text-align:center">

            <h2>Bulk Upload</h2>

            <form id="bulkForm" method="get">

                <input type="hidden" name="formID" value=3>

                <label for="fileInput">Upload File:</label><br>

                <input type="file" name="fileUpload" id="fileInput" accept="<?= getAcceptedFileTypes() ?>" style="margin-left: 3em;"><br>

                <?php

                    if (!empty($\_GET) && $\_GET["formID"] == 2 && isset($goodFile) && $goodFile == false) {

                            echo "<span style=\"color: red\">Please upload a valid file type (" . getAcceptedFileTypes() . ").</span><br>";

                    }

                ?>

                <div style="margin-top: 0.5em">

                    <input type="reset" value="Clear" class="button" onclick="clearForm(this.form.id)"/>

                    <input type="submit" value="Upload" class="button"/>

                </div>

            </form>

        </div>

        <hr>

        <h2>Change Log</h2> <!-- TODO: Finish Change Log -->

        <textarea rows="8" cols="50" readonly>

        </textarea>

        <br>

        <button style="cursor:pointer;" onclick="exportLog();">Export</button>

    </div>

    <footer>

        <script src="script.js"></script>

    </footer>

</body>

</html>

createFilter.php

<?php

require\_once "dbinteraction.php";

// TODO: add aggregate counter of currently showing games matching this checkbox

function createLabel ($inputID, $displayName) {

    return "<label for=\"$inputID\">$displayName</label>";

}

function createButtons(){

    return "<input class=\"button\" type=\"submit\" value=\"Search\"/>

    <button type=\"button\" class=\"button\" onclick=\"clearForm(this.form.id);\">Clear</button>";

}

function createTextInput($displayName, $inputID, $inputName, $placeholder = NULL) {

    $string =

    createLabel($inputID, $displayName) .

        "\n<input id=\"$inputID\" type=\"text\" name=\"$inputName\"";

    if (!is\_null($placeholder)) {

        $string .= " placeholder=\"$placeholder\"";

    }

    $string .= "/>\n<br>\n";

    return $string;

}

function createNumberInput($displayName, $inputID, $inputName, $minNum = NULL, $maxNum = NULL, $placeholder = NULL) {

    $string =

    createLabel($inputID, $displayName) .

        "\n<input id =\"$inputID\" type=\"number\" name=\"$inputName\"";

    if (!is\_null($placeholder)) {

        $string .= " placeholder=\"$placeholder\"";

    }

    if(!is\_null($minNum)) {

        $string .= " min=\"$minNum\"";

    }

    if(!is\_null($maxNum)) {

        $string .= " max=\"$maxNum\"";

    }

    $string .= "/>\n<br>\n";

    return $string;

}

function createNumberRange($labelName, $inputID, $inputName, $range, bool $wasSelected) {

    $string = "";

    if ($wasSelected) {

        $string .= "<input id=\"$inputID\" type=\"radio\" name=\"$inputName\" value=\"$range\" checked/>";

    } else {

        $string .= "<input id=\"$inputID\" type=\"radio\" name=\"$inputName\" value=\"$range\"/>";

    }

    $string .= createLabel($inputID, $labelName) . "<br>";

    return $string;

}

//Create individual checkbox with label

function createCheckboxInput($labelName, $displayName, $inputID, $inputName, $isChecked) {

    $string =

    "<input type=\"checkbox\" id=\"$inputID\" name=\"$inputName\" value=\"$displayName\"";

    if ($isChecked) {

        $string .= " checked";

    }

    $string .= ">" .

    createLabel($inputID, $labelName) .

    "<br>\n";

    return $string;

}

function initialiseCheckboxGroup($groupName, $category, array $stickyValues = NULL) {

    $string = "<div id=\"$groupName\">\n";

    if (!is\_null($stickyValues) && isset($stickyValues[$groupName])) {

        if ($stickyValues[$groupName][0] == $groupName) {

            $string .= "<input type=\"checkbox\" id=\"" . $groupName . "Checkbox\" onchange=\"selectAll(this.value, this.checked)\" name=\"" . $category . "[" . $groupName . "][]\" value=\"$groupName\" checked>";

        } else {

            $string .= "<input type=\"checkbox\" id=\"" . $groupName . "Checkbox\" onchange=\"selectAll(this.value, this.checked)\" name=\"" . $category . "[" . $groupName . "][]\" value=\"$groupName\">";

        }

    } else {

        $string .= "<input type=\"checkbox\" id=\"" . $groupName . "Checkbox\" onchange=\"selectAll(this.value, this.checked)\" name=\"" . $category . "[" . $groupName . "][]\" value=\"$groupName\">";

    }

    return $string;

}

//Create grouped checkboxes e.g. manufacturer

//TODO: anonymize function

function createGroupedCheckboxes(array $groupNames, $category, $inputName, array $filterCount, array $stickyValues = NULL) {

    $platforms = dbGetPlatforms();

    $string = "";

    foreach ($groupNames as $groupName) {

        $string .= initialiseCheckboxGroup($groupName, $category, $stickyValues);

        $string .= createLabel(($groupName . "Checkbox"), $groupName . " (" . $filterCount[$groupName] . ")") . "<br>" .

            "<div id=\"" . $groupName . "Selection\" style=\"padding-left:0.75em\">";

        foreach ($platforms as $platform) {

            $count = (array\_key\_exists($platform["name"], $filterCount) ? " (" . $filterCount[$platform["name"]] . ")" : " (0)");

            if ($platform["name"] != $platform["manufacturer"]) {

                if ($platform["manufacturer"] == $groupName) {

                    $found = false;

                    if (!is\_null($stickyValues) && isset($stickyValues[$groupName])) {

                        for ($i=0; $i < count($stickyValues[$groupName]); $i++) {

                            if ($stickyValues[$groupName][$i] == $platform["name"]) {

                                $string .= createCheckboxInput($platform["name"] . $count, $platform["name"], $platform["initial"], str\_replace("[]", "[" . $groupName . "][]", $inputName), true);

                                $found = true;

                                break;

                            }

                        }

                    }

                    if (!$found) {

                    $string .= createCheckboxInput($platform["name"] . $count, $platform["name"], $platform["initial"], str\_replace("[]", "[" . $groupName . "][]", $inputName), false);

                    }

                }

            }

        }

        $string .= "</div></div>";

    }

    return $string;

}

?>

Dbinteraction.php

<?php

require\_once "dbconnect.php";

function loadGamesFromDB(){

    global $conn;

    $games = array();

    $query = "SELECT \* FROM games";

    if ($result = $conn->query($query)) {

        while ($row = $result->fetch\_assoc()) {

            $newGame = array(

                "id" => $row["gID"],

                "title" => $row["title"],

                "price" => $row["price"],

                "genre" => getGenreNameByID($row["genre"]),

                "platform" => getPlatformNameByID($row["platform"]),

                "classification" => getClassificationInitialByID($row["classification"])

            );

            $games[] = $newGame;

        }

    }

    return $games;

}

// A challenge I rose too

function dbGetColumns(){

    global $conn;

    $columns = array();

    $query = "SELECT COLUMN\_NAME

    FROM information\_schema.COLUMNS

    WHERE table\_name = 'games'

      AND COLUMN\_KEY != 'PRI'

    ORDER BY ordinal\_position;";

    if ($result = $conn->query($query)) {

        while ($row = $result->fetch\_assoc()) {

            $columns[] = $row['COLUMN\_NAME'];

        }

        return $columns;

    }

}

function dbGetPlatforms() {

    global $conn;

    $platforms = array();

    $query = "SELECT \* FROM platforms;";

    if ($result = $conn->query($query)) {

        while ($row = $result->fetch\_assoc()) {

            $newPlatform = array(

                "id" => $row["pID"],

                "initial" => $row["initial"],

                "name" => $row["pName"],

                "manufacturer" => $row["manufacturer"],

                "description" => (isset($row["description"]) ? $row["description"] : "") //see if isset is actually necessary

            );

            $platforms[] = $newPlatform;

        }

        return $platforms;

    }

}

function dbGetClassification() {

    global $conn;

    $classifications = array();

    $query = "SELECT \* FROM classification;";

    if ($result = $conn->query($query)) {

        while ($row = $result->fetch\_assoc()) {

            $newClassification = array(

                "id" => $row["cID"],

                "initial" => $row["initial"],

                "description" => (isset($row["description"]) ? $row["description"] : "") //see if isset is actually necessary

            );

            $classifications[] = $newClassification;

        }

        return $classifications;

    }

}

function dbGetGenre() {

    global $conn;

    $genres = array();

    $query = "SELECT \* FROM genres;";

    if ($result = $conn->query($query)) {

        while ($row = $result->fetch\_assoc()) {

            $newGenre = array(

                "id" => $row["gID"],

                "name" => $row["gName"],

                "description" => (isset($row["description"]) ? $row["description"] : "") //see if isset is actually necessary

            );

            $genres[] = $newGenre;

        }

        return $genres;

    }

}

function getGenreID($gName) {

    global $conn;

    $genre = dbGetGenre($conn);

    foreach ($genre as $key) {

        if ($key["name"] == $gName) {

            return $key["id"];

        }

    }

    // TODO Add contingency to add new genre when not found

}

function getPlatformID($pName) {

    global $conn;

    $platforms = dbGetPlatforms($conn);

    foreach ($platforms as $key) {

        if ($key["name"] == $pName) {

            return $key["id"];

        }

    }

    // TODO Add contingency to add new platform when not found

}

function getClassificationID($cName) {

    global $conn;

    $classification = dbGetClassification($conn);

    foreach ($classification as $key) {

        if ($key["initial"] == $cName) {

            return $key["id"];

        }

    }

    // TODO Add contingency to add new classification when not found

}

//functions that convert id's to names/initials

function getPlatformNameByID($pID) {

    global $conn;

    $query = "SELECT pName from platforms WHERE pID = $pID";

    if ($result = $conn->query($query)) {

        while ($row = $result -> fetch\_row()) {

            return $row[0];

        }

    }

}

function getClassificationInitialByID($cID) {

    global $conn;

    $query = "SELECT initial from classification WHERE cID = $cID";

    if ($result = $conn->query($query)) {

        while ($row = $result -> fetch\_row()) {

            return $row[0];

        }

    }

}

function getGenreNameByID($gID) {

    global $conn;

    $query = "SELECT gName from genres WHERE gID = $gID";

    if ($result = $conn->query($query)) {

        while ($row = $result -> fetch\_row()) {

            return $row[0];

        }

    }

}

function getManufacturers() {

    global $conn;

    $manufacturers = array();

    $query = "SELECT DISTINCT manufacturer FROM platforms";

    if ($result = $conn->query($query)) {

        while ($row = $result->fetch\_assoc()) {

            $manufacturers[] = $row["manufacturer"];

        }

    }

    return $manufacturers;

}

function getPlatformNamesByManufacturers() {

    global $conn;

    $manufacturers = getManufacturers();

    $platforms = array();

    $query = "SELECT pName FROM PLATFORMS WHERE manufacturer = ?";

    foreach ($manufacturers as $key) {

        if ($stmt = $conn->prepare($query)) {

            $result = $stmt->bind\_param("s", $key);

            if (false === $result) {//if bind\_param didn't work

                die('bind\_param() failed');

            }

            $result = $stmt->execute();

            if (false === $result) {//failed to execute

                die('execute() failed: '. $stmt->error);

            }

            $result = $stmt->get\_result();

            $platforms[$key] = array();

            while ($row = $result->fetch\_assoc()) {

                $platforms[$key][] = $row["pName"];

            }

        }

    }

    if (!empty($platforms)) {

        return $platforms;

    }

}

?>

Header.html

<div class="header">

    <h1 style="display:inline-block"><a href="/" class="homeLink">Game Store</a></h1>

    <a href="/admin" style="float: right;margin-right: 2em;padding-top: 1.75em;">Admin Page</a>

</div>

Index.php

<?php

require\_once "dbinteraction.php";

require\_once "createFilter.php";

$games;

$columns;

$genres;

$platforms;

$classifications;

$filterCount = array();

function createCard($game) {

    $newCard =

    "<div class=\"cardCard\" id=\"" . $game["id"] . "\">

            <h3 name=\"title\" style=\"color:blue;padding:5%;\">" . $game["title"] . "</h4>

            <h5 name=\"genre\">". $game["genre"] . "</h5>

            <h6 name=\"platform\">" . $game["platform"] . "</h4>

            <h6 name=\"classification\">" . $game["classification"] . "</h4>

            <h4 name=\"price\">$" . $game["price"] . "</h4>

    </div>";

    addToFilterCount($game["genre"]);

    addToFilterCount($game["platform"]);

    addToFilterCount($game["classification"]);

    return $newCard;

}

function createRow($game1 = NULL, $game2 = NULL, $game3 = NULL) {

    $newRow =

    "<div class=\"row\">";

        if (!is\_null($game1))

            $newRow = $newRow . createCard($game1);

        if (!is\_null($game2))

            $newRow = $newRow . createCard($game2);

        if (!is\_null($game3))

            $newRow = $newRow . createCard($game3);

    $newRow = $newRow . "</div>";

    return $newRow;

}

function LoadGames($games) {

    $rows = "";

    $completeRows = intval(count($games) / 3);

    $partialRows = count($games) % 3;

    $totalrows = $completeRows;

    if ($partialRows > 0) {

        $totalrows = $completeRows+1;

    }

    for ($i = 0,$x = 0; $i < $completeRows; $i++) {

    $rows = $rows . createRow($games[$x], $games[$x+1], $games[$x+2]);

        $x = $x + 3;

    }

    if ($partialRows == 1) {

        $rows = $rows . createRow($games[$x]);

    } else if ($partialRows == 2) {

        $rows = $rows . createRow($games[$x], $games[$x+1]);

    }

    return $rows;

}

function getGenres() {

    $genres = array();

    if (!empty($\_GET)) {

        if (!empty($\_GET['genre'])) {

            foreach ($\_GET['genre'] as $key) {

                $genres[] = $key;

            }

        }

    }

    if (!empty($genres)){

        return $genres;

    }

}

function getPlatforms() {

    $platforms = array();

    if (!empty($\_GET)) {

        if (!empty($\_GET['platform'])) {

            foreach ($\_GET['platform'] as $key) {

                $platforms[] = $key;

            }

        }

    }

    if (!empty($platforms)){

        return $platforms;

    }

}

function getClassification() {

    $classification = array();

    if (!empty($\_GET)) {

        if (!empty($\_GET['classification'])) {

            foreach ($\_GET['classification'] as $selected) {

                $classification[] = $selected;

            }

        }

    }

    if (!empty($classification)){

        return $classification;

    }

}

function getTitle() {

    if (!empty($\_GET)) {

        if (!empty($\_GET['title'])) {

            return $\_GET['title'];

        }

    }

}

function getMinPrice() {

    if (!empty($\_GET)) {

        if (!empty($\_GET['minPrice'])) {

            return $\_GET['minPrice'];

        }

    }

}

function getMaxPrice() {

    if (!empty($\_GET)) {

        if (!empty($\_GET['maxPrice'])) {

            return $\_GET['maxPrice'];

        }

    }

}

function getPriceRange() {

    if (!empty($\_GET)) {

        if (!empty($\_GET['pRange'])) {

            $range["val1"] = explode("-", $\_GET['pRange'])[0];

            $range["val2"] = explode("-", $\_GET['pRange'])[1];

            if (intval($range["val1"]) < intval($range["val2"])) {

                return array("min"=> intval($range["val1"]), "max" => intval($range["val2"]));

            } else {

                return array("min"=> intval($range["val2"]), "max" => intval($range["val1"]));

            }

        }

    }

}

function addToFilterCount($key) {

    global $filterCount;

    if (array\_key\_exists($key, $filterCount)) {

        $filterCount[$key]++;

    } else {

        $filterCount[$key] = 1;

    }

}

function setGroupFilterCount() {

    global $filterCount;

    $manufacturers = getPlatformNamesByManufacturers();

    foreach ($manufacturers as $key => $value) {

        $count = 0;

        for ($i = 0; $i < count($value); $i++) {

            if (array\_key\_exists($value[$i], $filterCount)) {

                $count += $filterCount[$value[$i]];

            }

        }

        $filterCount[$key] = $count;

    }

}

function FilterGames($games) {

    $title = getTitle();

    $minPrice = (isset(getPriceRange()["min"]) ? getPriceRange()["min"] : null);

    $maxPrice = (isset(getPriceRange()["max"]) ? getPriceRange()["max"] : null);

    $genres = getGenres();

    $platforms = getPlatforms();

    $classifications = getClassification();

    $filtered = array();

    foreach ($games as $selected) {

        //Title

        if (isset($title)) {

            if (!is\_numeric(strpos($selected['title'], $title))) {

                continue;

            }

        }

        //Min Price

        if (isset($minPrice)) {

            if ($selected['price'] < $minPrice) {

                continue;

            }

        }

        //Max Price

        if (isset($maxPrice)) {

            if ($selected['price'] > $maxPrice) {

                continue;

            }

        }

        //Genre

        if (!empty($genres)) {

            $found = false;

            foreach ($genres as $key) {

                if ($key == $selected["genre"]) {

                    $found = true;

                    break;

                }

            }

            if (!$found) {

                continue;

            }

        }

        //Platform

        if (!empty($platforms)) {

            $found = false;

            foreach ($platforms as $platform) {

                foreach ($platform as $key) {

                    if ($key === $selected["platform"]) {

                        $found = true;

                        break;

                    }

                }

            }

            if (!$found) {

                continue;

            }

        }

        //Classification

        if (isset($classifications)) {

            $found = false;

            foreach ($classifications as $classification) {

                if ($classification === $selected["classification"]) {

                    $found = true;

                    break;

                }

            }

            if (!$found) {

                continue;

            }

        }

        $filtered[] = $selected;

    }

    return $filtered;

}

$games = loadGamesFromDB();

$genres = getGenres();

$classifications = getClassification();

$cards = LoadGames(FilterGames($games));

setGroupFilterCount();

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="shortcut icon" href="">

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

    <title>Game Store</title>

</head>

<body>

    <?php include 'header.html';

    $string = "<div id=\"searchArea\" class=\"searchArea\"><form id=\"form1\" style=\"margin-left: 0.5em; padding-bottom: 7em\">" .

    createTextInput("Game Title", "titleInput", "title", "Game Title") . "<hr>" .

    createLabel("Price", "Price Range") . "<br>" .

    createNumberRange("$0-$20"/\*\*/, "0-20", "pRange", "0-20", (isset($\_GET["pRange"]) && $\_GET["pRange"] == "0-20")) .

    createNumberRange("$21-$40"/\*\*/, "21-40", "pRange", "21-40", (isset($\_GET["pRange"]) && $\_GET["pRange"] == "21-40")) .

    createNumberRange("$41-$60"/\*\*/, "41-60", "pRange", "41-60", (isset($\_GET["pRange"]) && $\_GET["pRange"] == "41-60")) .

    createNumberRange("$61-$80"/\*\*/, "61-80", "pRange", "61-80", (isset($\_GET["pRange"]) && $\_GET["pRange"] == "61-80")) .

    createNumberRange("$81-$100"/\*\*/, "81-100", "pRange", "81-100", (isset($\_GET["pRange"]) && $\_GET["pRange"] == "81-100"));

    $string .= "<hr>";

    //Platform

    $string .= createLabel("Platform", "Platform") . "<br>";

    if (!empty($\_GET) && isset($\_GET["platform"]))

        $string .= createGroupedCheckboxes(getManufacturers($conn), "platform", "platform[]", $filterCount, $\_GET["platform"]);

    else

        $string .= createGroupedCheckboxes(getManufacturers($conn), "platform", "platform[]", $filterCount);

    $string .= "<hr>";

    //Genres

    $string .= createLabel("Genre", "Genre") . "<br>";

    $dbgenres = dbGetGenre();

    foreach ($dbgenres as $key) {

        $count = "(0)";

        $found = false;

        if (isset($genres)) {

            for ($i = 0; $i < count($genres); $i++) {

                if ($genres[$i] == $key["name"]) {

                    $found = true;

                    break;

                }

            }

        }

        if (array\_key\_exists($key["name"], $filterCount)) {

            $count = "(" . $filterCount[$key["name"]] . ")";

        }

        $string .= createCheckboxInput($key["name"] . " " . $count, $key["name"], $key["name"], "genre[]", $found);

    }

    $string .= "<hr>";

    //Classification

    $string .= createLabel("Classification", "Classification") . "<br>";

    $dbclassification = dbGetClassification();

    foreach ($dbclassification as $key) {

        $count = "(0)";

        $found = false;

        if (isset($classifications)) {

            for ($i = 0; $i < count($classifications); $i++) {

                if ($classifications[$i] == $key["initial"]) {

                    $found = true;

                    break;

                }

            }

        }

        if (array\_key\_exists($key["initial"], $filterCount)) {

            $count = "(" . $filterCount[$key["initial"]] . ")";

        }

        $string .= createCheckboxInput($key["initial"] . " " . $count, $key["initial"], $key["initial"], "classification[]", $found);

    }

    $string .= "<hr>";

    $string .= '<input class="button" type="submit" value="Search" style="width: -webkit-fill-available;margin-right: 0.5em;height: 2em;"/><br>

    <div style="text-align:center">

        <button type="button" class="button" onclick="clearForm(this.form.id);">Clear</button>

    </div>';

    $string .= "</form></div>";

    echo $string;

    ?>

    <div style="text-align:center">

        <div id="viewArea" class="viewArea">

            <?php echo $cards;?>

        </div>

    </div>

    <footer>

        <script src="script.js"></script>

    </footer>

</body>

</html>

modifyData.php

<?php

require\_once "dbinteraction.php";

if (!empty($\_POST)) {

    if ($\_POST["type"] == "update") {

        $array = checkArray($\_POST["data"]);

        if (array\_keys($array)[0] != "title") {

            // has errors

            print\_r($array);

        } else {

            // all clear

            $array["id"] = $\_POST["rowID"];

            updateRow($array);

        }

    } else if ($\_POST["type"] == "delete") {

        deleteRow($\_POST["rowID"]);

    }

} elseif (!empty($\_GET)) { //DEBUGING

    if ($\_GET["type"] == "update") {

        $array = checkArray($\_GET["data"]);

        if (array\_keys($array)[0] != "title") {

            // has errors

            print\_r($array);

        } else {

            // all clear

            $array["id"] = $\_GET["rowID"];

            updateRow($array);

        }

    } else if ($\_GET["type"] == "delete") {

        deleteRow($\_GET["rowID"]);

    }

}

function checkArray($array) {

    $errorlog = array();

    $error = false;

    if (strlen($array["title"]) < 1) {

        $errorlog[] = "Game Title cannot be empty.";

        $error = true;

    }

    if (!is\_numeric($array["price"])) {

        $errorlog[] = "Price must be a number.";

        $error = true;

    } else {

        $array["price"] = sprintf("%.2f", floatval($array["price"]));

    }

    $genreID = getGenreID($array["genre"]);

    if (is\_NULL($genreID)) {

        $errorlog[] = "Invalid Genre.";

        $error = true;

    } else {

        $array["genre"] = $genreID;

    }

    $platID = getPlatformID($array["platform"]);

    if (is\_NULL($platID)) {

        $errorlog[] = "Invalid Platform.";

        $error = true;

    } else {

        $array["platform"] = $platID;

    }

    $classID = getClassificationID($array["classification"]);

    if (is\_NULL($classID)) {

        $errorlog[] = "Invalid Classification.";

        $error = true;

    } else {

        $array["classification"] = $classID;

    }

    if ($error)

        return $errorlog;

    else

        return $array;

}

function updateRow($array) {

    global $conn;

    $query = "UPDATE games SET title = ?, price = ?, genre = ?, platform = ?, classification = ? WHERE gID = ?";

    if ($stmt = $conn->prepare($query)) {

        $result = $stmt->bind\_param("sdiiii", $array["title"], $array["price"], $array["genre"], $array["platform"], $array["classification"], $array["id"]);

        if ( false===$result ) {

            die('bind\_param() failed');

        }

        $result = $stmt->execute();

        if ( false===$result ) {

            die('execute() failed: '.$stmt->error);

        }

    }

}

function deleteRow($rowID) {

    global $conn;

    $query = "DELETE FROM games WHERE gID = ?";

    if ($stmt = $conn->prepare($query)) {

        $result = $stmt->bind\_param("i", $rowID);

        if ( false===$result ) {

            die('bind\_param() failed');

        }

        $result = $stmt->execute();

        if ( false===$result ) {

            die('execute() failed: '.$stmt->error);

        }

    }

}

?>

Script.js

let changedEntries = [];

function clearForm(formID){

    let form = document.getElementById(formID);

    [...form.elements].forEach((input) => {

        if (input.type == "text") {

            if (input.value != null) {

                //console.log(input.id);

                document.getElementById(input.id).value = "";

            }

        } else if (input.type == "number") {

            if (input.value != null) {

                //console.log(input.id);

                document.getElementById(input.id).value = "";

            }

        } else if (input.type == "checkbox") {

            input.checked = false;

        } else if (input.type == "radio") {

            input.checked = false;

        }

    });

}

function checkRow(rowID, isRowIncluded) {

    if (isRowIncluded) {

        let found = false;

        for (let i = 0; i < changedEntries.length-1; i++) {

            if (changedEntries[i] === rowID) {

                found = true;

                break;

            }

        }

        if (!found) {

            changedEntries.push(rowID);

        }

    } else {

        var filtered = changedEntries.filter(function(value){ return value !== rowID;});

        changedEntries = filtered;

    }

    let buttons = document.getElementsByClassName("button all");

    if (changedEntries.length > 0) {

        if (buttons[0].disabled) {

            for (let i = 0; i < buttons.length; i++) {

                buttons[i].disabled = false;

            }

        }

    } else {

        for (let i = 0; i < buttons.length; i++) {

            buttons[i].disabled = true;

        }

    }

}

function getRowData(rowID) {

    let rowData = document.getElementById(rowID).children;

    let rowInfo = [];

    for (let i = 1; i < rowData.length-1; i++) {

        let innerElements = rowData[i].children;

        for (let x = 0; x < innerElements.length; x++) {

            if (innerElements[x].hasAttribute("name")) {

                rowInfo[innerElements[x].getAttribute("name")] = innerElements[x].innerHTML;

            }

        }

    }

    return rowInfo;

}

function updateRow(rowID) {

    let data = getRowData(rowID);

    let string = buildHTTPquery(data);

    var xhttp = new XMLHttpRequest();

    xhttp.open("POST", "modifyData.php", true);

    xhttp.responseType = "text";

    xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

    xhttp.send("type=update&rowID=" + rowID + "&" + string);

    xhttp.onload = function () {

        if (xhttp.readyState === xhttp.DONE) {

            if (xhttp.status === 200) {

                // If successful response

                console.log(xhttp.response);

            }

        }

    };

}

function deleteRow(rowID) {

    var xhttp = new XMLHttpRequest();

    xhttp.open("POST", "modifyData.php", true);

    xhttp.responseType = "text";

    xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

    xhttp.send("type=delete&rowID=" + rowID);

    xhttp.onload = function () {

        if (xhttp.readyState === xhttp.DONE) {

            if (xhttp.status === 200) {

                console.log(xhttp.response);

            }

        }

    };

}

function UpdateAll() {

    changedEntries.forEach(element => {

        updateRow(element);

    });

}

function DeleteAll() {

    changedEntries.forEach(element => {

        deleteRow(element);

    });

}

function buildHTTPquery(array) {

    let string = "";

    let keys = Object.keys(array);

    keys.forEach(element => {

        string += "data[" + element + "]=" + array[element] + "&";

    });

    string = string.substring(0, string.length - 1);

    return encodeURI(string);

}

function selectAll(divID, isSelected) {

    var inputs = document.getElementById(divID + "Selection").getElementsByTagName("input");

    for (let index = 0; index < inputs.length; index++) {

        inputs[index].checked = isSelected;

    }

}

Style.css

body {

    /\* padding-bottom: 6em; \*/

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    overflow-x: hidden;

}

.searchArea {

    margin-top: 4.85em;

    margin-left: -0.47em;

    padding: 0;

    width: 13em;

    background-color: #f1f1f1;

    position: fixed;

    height: 100%;

    overflow: auto;

}

.viewArea {

    padding-top: 6em;

    margin-left: 16em;

}

textarea {

    outline: none;

    overflow: auto;

}

.row {

    display: flex;

    justify-content: center;

    margin-right: -10%;

    margin-left: -10%;

}

.cardCard {

    text-align:center;

    min-width: 25%;

    max-width: 25%;

    margin:1.25%;

    border: 1px solid black;

    border-radius: 5px;

}

.header {

    background-color: forestgreen;

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    padding-left: 1em;

}

.button {

    cursor:pointer;

    margin-left: .5em;

}

.priceFields {

    width:4em;

}

.column {

    text-align: left;

    float: left;

    width: auto;

    margin-left: 1em;

}

.row::after {

    max-width: 50%;

    content: "";

    display: table;

    clear: both;

}

.homeLink {

    color: black;

    text-decoration:none;

}

.homeLink:visited {

    color: black;

}

@media screen and (max-width: 600px) {

    .column {

      width: 100%;

    }

}

table {

    width: 100%;

}

thead, tbody tr {

    display: table;

    width: 60em;

    table-layout: fixed;

    border-collapse: collapse;

}

tbody {

    display: block;

    overflow-x: hidden;

    overflow-y: scroll;

    table-layout: fixed;

    max-height: 15em;

}

th, td {

    border: 1px solid black;

}

th {

    border-bottom: 5px solid black;

}

td {

    border-top: 0px;

}

.col1 {

    border:0px;

    width:2em;

}

.col2 {

    min-width:3.5em;

}

.col3 {

    min-width:3.5em;

    width:5em;

}

.col4 {

    min-width:4.5em;

    width:5.5em;

}

.col5 {

    min-width:6.5em;

    width:6.5em

}

.col6 {

    border:0px;

    width: 4.5em;

}

Table.css

table {

    width: 100%;

}

thead, tbody tr {

    display: table;

    width: 60em;

    table-layout: fixed;

    border-collapse: collapse;

}

tbody {

    display: block;

    overflow-x: hidden;

    overflow-y: scroll;

    table-layout: fixed;

    max-height: 15em;

}

th, td {

    border: 1px solid black;

}

th {

    border-bottom: 5px solid black;

}

td {

    border-top: 0px;

}

.col1 {

    border:0px;

    width:2em;

}

.col2 {

    min-width:3.5em;

}

.col3 {

    min-width:3.5em;

    width:5em;

}

.col4 {

    min-width:4.5em;

    width:5.5em;

}

.col5 {

    min-width:6.5em;

    width:6.5em

}

.col6 {

    border:0px;

    width: 4.5em;

}