**CSYM019: Internet Programming  
Assignment**   
  
Name : Rathnayake Mudiyanselage Sanindu Madhushan Rathnayake

Student ID: 23849711

Code Explanation

Part – 02

**Part 02**

**report.php**

<!DOCTYPE html>

<html lang="en">

<head>

    <!-- Meta tags for character set and viewport -->

    <meta charset="UTF-8">

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

     <!-- Title of the page -->

    <title>Premier League Report</title>

     <!-- Link to external stylesheet -->

    <link rel="stylesheet" href="layout.css">

     <!-- Link to Chart.js library -->

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<header>

     <!-- Header displaying the title -->

    <h3>CSYM019 - Premier League Results</h3>

</header>

<nav>

    <!-- Navigation menu -->

    <ul>

        <!-- Links to other pages -->

        <li><a href="./report.php">Premier League Report</a></li>

        <li><a href="./entryForm.php">Add New Football Team</a></li>

    </ul>

</nav>

<main>

    <!-- Main content -->

    <h3>Football Teams Selection Form</h3>

     <!-- Success message for edit -->

    <div id="editSuccessMessage" style="display: none; color: green;">Successfully edited!</div>

     <!-- Form for selecting teams -->

    <div class="sketch">

        <form action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" method="post" id="teamsForm">

         <!-- Hidden input field for deleting a team -->

         <input type="hidden" name="delete\_team\_iden" id="delete\_team\_iden">

          <!-- Table for displaying team data -->

            <table>

                 <!-- Table headers -->

                <thead>

                <tr>

                     <!-- Checkbox for selecting all rows -->

                    <th><input type="checkbox" id="select-all"></th>

                     <!-- Column headers -->

                    <th>Position</th>

                    <th>Team</th>

                    <th>Won</th>

                    <th>Drawn</th>

                    <th>Lost</th>

                    <th>For</th>

                    <th>Against</th>

                    <th>GD</th>

                    <th>Points</th>

                    <th>Remaining</th>

                    <th>Actions</th>

                </tr>

                </thead>

                <tbody>

                <!-- Table body -->

                <?php

                // Database connection

                $servername = "localhost";

                $username = "root";

                $password = "";

                $dbname = "league";

                // Referance :  MySQLi Extension Documentation. Available online at: [MySQLi Extension Documentation](https://www.php.net/manual/en/book.mysqli.php)

                $conn = new mysqli($servername, $username, $password, $dbname);

                // Check connection

                if ($conn->connect\_error) {

                    echo "<p> Connection Failed<p>";

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

                }

                // Handle deletion of selected records

                if ($\_SERVER["REQUEST\_METHOD"] == "POST") { // Check if the request method is POST

                    if (isset($\_POST['delete\_selected']) && isset($\_POST['teams']) && is\_array($\_POST['teams'])) { // Check if 'delete\_selected' field is set and 'teams' field is set and is an array

                        $selected\_teams = $\_POST['teams']; // Retrieve the selected teams from the form data

                        foreach ($selected\_teams as $team\_id) { // Loop through each selected team ID

                            $sql = "DELETE FROM teams WHERE id = $team\_id"; // Construct SQL query to delete the team with the specified ID

                            $conn->query($sql); // Execute the SQL query to delete the team

                        }

                        header("Location: " . $\_SERVER["PHP\_SELF"]); // Redirect back to the current page after successful deletion

                        exit; // Exit the script to prevent further execution

                    }

                }

                // Initialize teams data array

                $teams\_data = [];

                // Handle report generation

                if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['generate\_report']) && isset($\_POST['teams']) && is\_array($\_POST['teams'])) {

                    // Check if the request method is POST and the form has been submitted to generate a report

                    // Also, check if the 'teams' field is set and is an array

                    $selected\_teams = $\_POST['teams']; // Retrieve the selected teams from the form data

                    // Fetch data for selected teams

                    foreach ($selected\_teams as $team\_id) {

                        // Loop through each selected team ID

                        $sql = "SELECT \* FROM teams WHERE id = $team\_id"; // Construct SQL query to fetch data for the team with the specified ID

                        $result = $conn->query($sql); // Execute the SQL query

                        if ($result->num\_rows > 0) { // Check if any rows are returned from the query

                            $row = $result->fetch\_assoc(); // Fetch the row of data from the result set

                            $teams\_data[] = $row; // Add the fetched data to the $teams\_data array

                        }

                    }

                }

if ($\_SERVER["REQUEST\_METHOD"] == "POST") { // Check if the request method is POST

    if (isset($\_POST['delete\_selected']) && isset($\_POST['teams']) && is\_array($\_POST['teams'])) { // Check if 'delete\_selected' field is set and 'teams' field is set and is an array

        $selected\_teams = $\_POST['teams']; // Retrieve the selected teams from the form data

        foreach ($selected\_teams as $team\_id) { // Loop through each selected team ID

            $sql = "DELETE FROM teams WHERE id = $team\_id"; // Construct SQL query to delete the team with the specified ID

            $conn->query($sql); // Execute the SQL query to delete the team

        }

        header("Location: " . $\_SERVER["PHP\_SELF"]); // Redirect back to the current page after successful deletion

        exit; // Exit the script to prevent further execution

    }

}

                    // Handle individual deletion

                  // Referance :  Exception Handling in PHP. Available online at: [Exception Handling in PHP](https://www.php.net/manual/en/language.exceptions.php)

                    if (isset($\_POST['delete\_team\_iden'])) {

                        try {

                            // Establish database connection

                            // Assuming $conn is already initialized earlier

                            // Retrieve the team ID to be deleted from the POST data

                            $team\_id = $\_POST['delete\_team\_iden'];

                            // Construct SQL query to delete the team with the specified ID from the 'teams' table

                            $sql = "DELETE FROM teams WHERE id = $team\_id";

                            // Execute the SQL query

                            if ($conn->query($sql) === TRUE) {

                                // If deletion is successful, redirect back to the current page to reflect changes

                                header("Location: " . $\_SERVER["PHP\_SELF"]);

                                exit;

                            } else {

                                // If there's an error during deletion, throw an exception with the error message

                                throw new Exception("Error deleting record: " . $conn->error);

                            }

                        } catch (Exception $e) {

                            // Catch any exceptions thrown during the deletion process and display the error message

                            echo "Error: " . $e->getMessage();

                        }

                    }

// Check if the form has been submitted using POST method and if the 'edit\_team' field is set

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['edit\_team'])) {

    // Retrieve the data submitted via POST

    $team\_id = $\_POST['team\_id'];

    $team = $\_POST['team'];

    $won = $\_POST['won'];

    $drawn = $\_POST['drawn'];

    $lost = $\_POST['lost'];

    $goalsfor = $\_POST['goalsfor'];

    $against = $\_POST['against'];

    $gd = $\_POST['gd'];

    $points = $\_POST['points'];

    $remaining = $\_POST['remaining'];

    // Construct the SQL query to update the team information in the database

    $sql = "UPDATE teams SET

            team='$team',

            won='$won',

            drawn='$drawn',

            lost='$lost',

            goalsfor='$goalsfor',

            against='$against',

            gd='$gd',

            points='$points',

            remaining = '$remaining'

            WHERE id=$team\_id";

    // Execute the SQL query

    $conn->query($sql);

    // Redirect back to the current page after successful update

    header("Location: " . $\_SERVER["PHP\_SELF"]);

    exit;

}

// SQL query to fetch teams from the database sorted by points in descending order

$sql = "SELECT id, team, won, drawn, lost, goalsfor, against, gd, points , remaining FROM teams ORDER BY points DESC";

$result = $conn->query($sql);

// Check if there are any rows returned from the query

        if ($result->num\_rows > 0) {

            // Variable to keep track of team position

            $position = 1;

            // Loop through each row of the result set

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

                // Display each team's information in a table row

                echo "<tr>";

                echo "<td><input type='checkbox' name='teams[]' class='select-row' value='" . $row["id"] . "'></td>";

                echo "<td>" . $position++ . "</td>";

                echo "<td>" . $row["team"] . "</td>";

                echo "<td>" . $row["won"] . "</td>";

                echo "<td>" . $row["drawn"] . "</td>";

                echo "<td>" . $row["lost"] . "</td>";

                echo "<td>" . $row["goalsfor"] . "</td>";

                echo "<td>" . $row["against"] . "</td>";

                echo "<td>" . $row["gd"] . "</td>";

                echo "<td>" . $row["points"] . "</td>";

                echo "<td>" . $row["remaining"] . "</td>";

                // Buttons for editing and deleting team data

                echo "<td><button type='button' name='editbuttonaction' onclick='editTeam(" . json\_encode($row) . ")'>Edit</button>

                <button type='button' name='deletebuttonaction' onclick='deleteTeam(" . $row["id"] . ")'>Delete</button></td>";

                echo "</tr>";

            }

        } else {

            // If no teams are found in the database, display a message

            echo "<tr><td colspan='11'>No teams found</td></tr>";

        }

        // Close the database connection

        $conn->close();

                ?>

                </tbody>

            </table>

            <input type="submit" value="Create Report" id ="createreportbutton" name="generate\_report"/>

        </form>

        <br>

    </div>

    <?php if (!empty($teams\_data)): ?> <!-- Check if $teams\_data is not empty -->

    <div id="charts-container" style="width: 50%; height: 50%;"> <!-- Display a div element with ID "charts-container" -->

        <?php foreach ($teams\_data as $index => $team): ?> <!-- Iterate over each element in $teams\_data -->

            <!-- Display team name -->

            <h4><?php echo $team['team']; ?></h4>

            <!-- Create canvas element for pie chart with unique ID -->

            <canvas id="pieChart<?php echo $index; ?>"></canvas>

        <?php endforeach; ?>

        <!-- If there are more than one team, create canvas element for bar chart -->

        <?php if (count($teams\_data) > 1): ?>

            <canvas id="barChart"></canvas>

        <?php endif; ?>

    </div>

    <!-- JavaScript code for chart creation -->

    <script>

        // Encode PHP array $teams\_data to JSON format

        const teamsData = <?php echo json\_encode($teams\_data); ?>;

        // Loop through each team data and create pie chart

        teamsData.forEach((team, index) => {

            // Get canvas context for pie chart

            const pieCtx = document.getElementById('pieChart' + index).getContext('2d');

            // Calculate total matches

            const totalMatches = team.won + team.drawn + team.lost + team.remaining;

            // Data for pie chart

            const pieData = {

    labels: ['Won', 'Drawn', 'Lost', 'Remaining'], // Labels for each segment of the pie chart

    datasets: [{ // Array of datasets, containing data and styling information for the chart

        data: [team.won, team.drawn, team.lost, team.remaining], // Array of numerical data values for each segment

        backgroundColor: ['#006400', '#F4BC1C', '#8b0000', '#1260CC'], // Array of background colors for each segment

    }]

};

            // Create new pie chart instance

            new Chart(pieCtx, { // Creates a new Chart instance with the specified canvas context (pieCtx)

    type: 'pie', // Specifies the type of chart to be created (pie chart)

    data: pieData, // Specifies the data to be used for rendering the chart (defined earlier in the code)

    maintainAspectRatio: false, // Specifies whether to maintain the aspect ratio of the chart (set to false)

    options: { // Specifies additional options for configuring the chart's appearance and behavior

        responsive: true, // Specifies whether the chart should be responsive (resize to fit its container)

        plugins: { // Specifies plugins to be used for the chart (in this case, legend and tooltip plugins)

            legend: { // Configures the legend plugin

                position: 'top', // Specifies the position of the legend (top of the chart)

            },

            tooltip: { // Configures the tooltip plugin

                callbacks: { // Specifies callback functions for customizing tooltip behavior

                    label: function (context) { // Configures the label callback for tooltips

                        const label = context.label || ''; // Retrieves the label of the current tooltip item

                        const value = context.raw; // Retrieves the raw value of the current tooltip item

                        const percentage = ((value / totalMatches) \* 100).toFixed(2); // Calculates the percentage of the total matches

                        return `${label}: ${value} (${percentage}%)`; // Returns the customized tooltip label

                    }

                }

            }

        }

    }

});

        });

        // If there are more than one team, create bar chart

        if (teamsData.length > 1) {

            // Get canvas context for bar chart

            const barCtx = document.getElementById('barChart').getContext('2d');

            // Data for bar chart

                    const barData = {

                   labels: teamsData.map(team => team.team), // Array of labels for the bars (team names)

                   datasets: [ // Array of datasets, each representing a set of bars

                {

                    label: 'Won', // Label for the first dataset

                    data: teamsData.map(team => team.won), // Array of numerical data values for the first dataset (number of matches won for each team)

                    backgroundColor: '#006400' // Background color for the bars in the first dataset (green)

                },

                {

                    label: 'Drawn', // Label for the second dataset

                    data: teamsData.map(team => team.drawn), // Array of numerical data values for the second dataset (number of matches drawn for each team)

                    backgroundColor: '#F4BC1C' // Background color for the bars in the second dataset (yellow)

                },

                {

                    label: 'Lost', // Label for the third dataset

                    data: teamsData.map(team => team.lost), // Array of numerical data values for the third dataset (number of matches lost for each team)

                    backgroundColor: '#8b0000' // Background color for the bars in the third dataset (dark red)

                },

                {

                    label: 'Remaining', // Label for the fourth dataset

                    data: teamsData.map(team => team.remaining), // Array of numerical data values for the fourth dataset (number of matches remaining for each team)

                    backgroundColor: '#1260CC' // Background color for the bars in the fourth dataset (blue)

                }

            ]

        };

                     // Create new bar chart instance

                        new Chart(barCtx, { // Creates a new Chart instance with the specified canvas context (barCtx)

                type: 'bar', // Specifies the type of chart to be created (bar chart)

                data: barData, // Specifies the data to be used for rendering the chart (defined earlier in the code)

                options: { // Specifies additional options for configuring the chart's appearance and behavior

                    responsive: true, // Specifies whether the chart should be responsive (resize to fit its container)

                    scales: { // Specifies scale options for the axes

                        x: { // Configures the x-axis

                            stacked: false // Specifies whether bars should be stacked on the x-axis (set to false)

                        },

                        y: { // Configures the y-axis

                            stacked: false // Specifies whether bars should be stacked on the y-axis (set to false)

                        }

                    }

                }

            });

        }

    </script>

<?php endif; ?>

<!-- Edit Team Modal -->

<div id="editModal" style="display:none;">

    <!-- Form for editing team data -->

    <form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>">

        <!-- Hidden input field to store the team ID -->

        <input type="hidden" name="team\_id" id="edit\_team\_id">

        <!-- Input field for team name -->

        <label>Team: <input type="text" name="team" id="edit\_team"></label><br>

        <!-- Input field for number of matches won -->

        <label>Won: <input type="number" name="won" id="edit\_won"></label><br>

        <!-- Input field for number of matches drawn -->

        <label>Drawn: <input type="number" name="drawn" id="edit\_drawn"></label><br>

        <!-- Input field for number of matches lost -->

        <label>Lost: <input type="number" name="lost" id="edit\_lost"></label><br>

        <!-- Input field for goals scored -->

        <label>For: <input type="number" name="goalsfor" id="edit\_goalsfor"></label><br>

        <!-- Input field for goals conceded -->

        <label>Against: <input type="number" name="against" id="edit\_against"></label><br>

        <!-- Input field for goal difference -->

        <label>GD: <input type="number" name="gd" id="edit\_gd"></label><br>

        <!-- Input field for points -->

        <label>Points: <input type="number" name="points" id="edit\_points"></label><br>

        <!-- Input field for remaining matches -->

        <label>Remaining: <input type="number" name="remaining" id="edit\_remaining"></label><br>

        <!-- Button to submit the form and save changes -->

        <input type="submit" name="edit\_team" value="Save Changes">

    </form>

</div>

</main>

<!-- Footer indicating the copyright -->

<footer>&copy; CSYM019 2024</footer>

<script>

// Event listener for the 'select-all' checkbox

document.getElementById('select-all').addEventListener('change', function() {

    // Select all checkboxes with the class 'select-row' and set their checked status to match the 'select-all' checkbox

    var checkboxes = document.querySelectorAll('.select-row');

    for (var checkbox of checkboxes) {

        checkbox.checked = this.checked;

    }

    // Show or hide the 'delete-selected' button based on whether any checkboxes are checked

    document.getElementById('delete-selected').style.display = this.checked ? 'inline-block' : 'none';

});

// Event listener for individual checkboxes with class 'select-row'

document.querySelectorAll('.select-row').forEach(function(checkbox) {

    checkbox.addEventListener('change', function() {

        // Check if any checkbox with class 'select-row' is checked

        var anyChecked = document.querySelectorAll('.select-row:checked').length > 0;

        // Show or hide the 'delete-selected' button based on whether any checkboxes are checked

        document.getElementById('delete-selected').style.display = anyChecked ? 'inline-block' : 'none';

    });

});

// Function to populate the edit modal with team data

function editTeam(team) {

    // Populate form fields with team data

    document.getElementById('edit\_team\_id').value = team.id;

    document.getElementById('edit\_team').value = team.team;

    document.getElementById('edit\_won').value = team.won;

    document.getElementById('edit\_drawn').value = team.drawn;

    document.getElementById('edit\_lost').value = team.lost;

    document.getElementById('edit\_goalsfor').value = team.goalsfor;

    document.getElementById('edit\_against').value = team.against;

    document.getElementById('edit\_gd').value = team.gd;

    document.getElementById('edit\_points').value = team.points;

    document.getElementById('edit\_remaining').value = team.remaining;

    // Show the edit modal

    document.getElementById('editModal').style.display = 'block';

}

// Event listener for form submission

document.querySelector('form[action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>"]').addEventListener('submit', function(event) {

    // After form submission, show the success message if it was a successful edit

    document.getElementById('editSuccessMessage').style.display = <?php echo isset($\_POST['edit\_team']) ? 'block' : 'none'; ?>;

});

// Function to delete a team

function deleteTeam(id) {

    // Display a confirmation dialog before proceeding with deletion

    const confirmation = confirm("Are you sure you want to delete this team?");

    if (confirmation) {

        // Set the value of the hidden input field to the team ID to be deleted

        document.getElementById('delete\_team\_iden').value = id;

        // Submit the form for team deletion

        document.getElementById('teamsForm').submit();

    }

}

</script>

</body>

<!--

References:

1. PHP Official Documentation. Available online at: [PHP Official Documentation](https://www.php.net/manual/en/)

2. MySQLi Extension Documentation. Available online at: [MySQLi Extension Documentation](https://www.php.net/manual/en/book.mysqli.php)

3. Exception Handling in PHP. Available online at: [Exception Handling in PHP](https://www.php.net/manual/en/language.exceptions.php)

4. PHP Header Function. Available online at: [PHP Header Function](https://www.php.net/manual/en/function.header.php)

5. PHP Arrays. Available online at: [PHP Arrays](https://www.php.net/manual/en/language.types.array.php)

6. Chart.js library

 -->

</html>

**Layout.css**

**Layout.css**body {

    display: grid; /\* Use CSS grid layout for the body \*/

    font-family: Arial, sans-serif; /\* Set font family for the body \*/

    grid-template-columns: 15% 85%; /\* Define the width ratio of grid columns \*/

    grid-template-rows: auto; /\* Automatically size grid rows based on content \*/

    grid-template-areas: "header header" "nav main" "footer footer"; /\* Define the grid areas for the body \*/

    margin: 20px; /\* Set margin for the body \*/

    background-color: #f4f4f9; /\* Set background color for the body \*/

    color: #333; /\* Set text color for the body \*/

}

header {

    background-color: #555; /\* Set background color for the header \*/

    color: white; /\* Set text color for the header \*/

    padding: 10px; /\* Set padding for the header \*/

    grid-area: header; /\* Assign the header to the grid area defined in the body \*/

}

nav {

    background-color: whitesmoke; /\* Set background color for the navigation \*/

    color: black; /\* Set text color for the navigation \*/

    grid-area: nav; /\* Assign the navigation to the grid area defined in the body \*/

}

nav>ul {

    list-style-type: none; /\* Remove list bullet points from the navigation \*/

    margin: 0; /\* Remove margin from the navigation \*/

    padding: 0; /\* Remove padding from the navigation \*/

    border-right: 2%; /\* Set a right border for the navigation \*/

    border-color: whitesmoke; /\* Set border color for the navigation \*/

    border-style: solid; /\* Set border style for the navigation \*/

    background-color: whitesmoke; /\* Set background color for the navigation \*/

}

nav>ul li a {

    display: block; /\* Display links as block elements within the navigation \*/

    color: #000; /\* Set text color for the navigation links \*/

    padding: 8px 16px; /\* Set padding for the navigation links \*/

    text-decoration: none; /\* Remove underline from the navigation links \*/

}

/\* Change the link color on hover \*/

nav>ul li a:hover {

    background-color: #555; /\* Set background color for the navigation links on hover \*/

    color: whitesmoke; /\* Set text color for the navigation links on hover \*/

}

main {

    background-color: whitesmoke; /\* Set background color for the main content \*/

    color: black; /\* Set text color for the main content \*/

    margin-left: 1em; /\* Set left margin for the main content \*/

    padding-left: 1em; /\* Set left padding for the main content \*/

    grid-area: main; /\* Assign the main content to the grid area defined in the body \*/

}

footer {

    padding-top: 2em; /\* Set top padding for the footer \*/

    background-color: #555; /\* Set background color for the footer \*/

    color: white; /\* Set text color for the footer \*/

    grid-area: footer; /\* Assign the footer to the grid area defined in the body \*/

}

.note {

    color: red; /\* Set text color for elements with class 'note' \*/

    font-size: 12pt; /\* Set font size for elements with class 'note' \*/

    font-family: Arial, Helvetica, sans-serif; /\* Set font family for elements with class 'note' \*/

    padding-top: 1em; /\* Set top padding for elements with class 'note' \*/

}

.blueNote {

    color: blue; /\* Set text color for elements with class 'blueNote' \*/

    font-size: 12pt; /\* Set font size for elements with class 'blueNote' \*/

    font-family: Arial, Helvetica, sans-serif; /\* Set font family for elements with class 'blueNote' \*/

    padding-top: 1em; /\* Set top padding for elements with class 'blueNote' \*/

    font-weight: bold; /\* Set font weight for elements with class 'blueNote' \*/

}

.sketch img {

    width: 60%; /\* Set width for images within elements with class 'sketch' \*/

    height: auto; /\* Set height to auto for images within elements with class 'sketch' \*/

}

.addmore>input {

    display: block; /\* Display inputs as block elements within elements with class 'addmore' \*/

    float: left; /\* Float inputs to the left within elements with class 'addmore' \*/

    margin: 0 2em 2em 2em; /\* Set margin for inputs within elements with class 'addmore' \*/

    padding: 0.5em 5em; /\* Set padding for inputs within elements with class 'addmore' \*/

    text-decoration: none; /\* Remove underline from inputs within elements with class 'addmore' \*/

    background-color: whitesmoke; /\* Set background color for inputs within elements with class 'addmore' \*/

    color: #555; /\* Set text color for inputs within elements with class 'addmore' \*/

}

/\* Change the link color on hover \*/

.addmore>input:hover {

    background-color: #555; /\* Set background color for inputs within elements with class 'addmore' on hover \*/

    color: whitesmoke; /\* Set text color for inputs within elements with class 'addmore' on hover \*/

}

/\* Styles for h1 \*/

h1 {

    text-align: center; /\* Center align h1 \*/

    margin-bottom: 20px; /\* Add bottom margin to h1 \*/

    color: #4CAF50; /\* Set color for h1 \*/

}

/\* Styles for table \*/

table {

    width: 100%; /\* Set table width to 100% \*/

    border-collapse: collapse; /\* Collapse table borders \*/

    margin: 20px 0; /\* Add margin above and below the table \*/

    font-size: 16px; /\* Set font size for table \*/

    box-shadow: 0 2px 3px rgba(0, 0, 0, 0.1); /\* Add box shadow to table \*/

    background-color: #ffffff; /\* Set background color for table \*/

}

/\* Styles for th and td \*/

th,

td {

    padding: 12px; /\* Set padding for th and td \*/

    text-align: left; /\* Align text to the left for th and td \*/

    border-bottom: 1px solid #ddd; /\* Add bottom border to th and td \*/

}

/\* Styles for thead th \*/

thead th {

    background-color: #4CAF50; /\* Set background color for thead th \*/

    color: #ffffff; /\* Set text color for thead th \*/

}

/\* Styles for even rows in tbody \*/

tbody tr:nth-child(even) {

    background-color: #f9f9f9; /\* Set background color for even rows in tbody \*/

}

/\* Styles for hovered rows in tbody \*/

tbody tr:hover {

    background-color: #f1f1f1; /\* Set background color for hovered rows in tbody \*/

}

/\* Styles for checkboxes in th \*/

th input[type="checkbox"] {

    margin: 0; /\* Remove margin for checkboxes in th \*/

}

/\* Styles for all checkboxes \*/

input[type="checkbox"] {

    transform: scale(1.2); /\* Increase checkbox size \*/

    margin-right: 10px; /\* Add margin to the right of checkboxes \*/

}

th,

td {

    border-right: 1px solid #ddd; /\* Add right border to th and td \*/

}

th:last-child,

td:last-child {

    border-right: none; /\* Remove right border for last-child th and td \*/

}

h4.team-name {

    font-size: 40px; /\* Set font size for h4 \*/

    text-align: center; /\* Center align text for h4 \*/

    margin-bottom: 10px; /\* Add bottom margin to h4 \*/

}

/\* Styles for Report Button \*/

input[type="submit"][name="generate\_report"] {

    background-color: #007bff; /\* Set background color for generate\_report button \*/

    color: white; /\* Set text color for generate\_report button \*/

    padding: 10px 20px; /\* Set padding for generate\_report button \*/

    border: none; /\* Remove border for generate\_report button \*/

    border-radius: 5px; /\* Add border radius to generate\_report button \*/

    cursor: pointer; /\* Set cursor to pointer for generate\_report button \*/

    transition: background-color 0.3s; /\* Add transition for smoother hover effect \*/

    margin-top: 10px; /\* Add top margin to generate\_report button \*/

}

input[type="submit"][name="generate\_report"]:hover {

    background-color: #0056b3; /\* Change background color on hover for generate\_report button \*/

}

/\* Styles for the edit form modal \*/

#editModal {

    display: none; /\* Hide edit modal by default \*/

    position: fixed; /\* Position edit modal as fixed \*/

    top: 50%; /\* Align edit modal to the center vertically \*/

    left: 50%; /\* Align edit modal to the center horizontally \*/

    transform: translate(-50%, -50%); /\* Center edit modal \*/

    background-color: white; /\* Set background color for edit modal \*/

    padding: 20px; /\* Set padding for edit modal \*/

    border: 2px solid #ccc; /\* Add border to edit modal \*/

    border-radius: 10px; /\* Add border radius to edit modal \*/

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /\* Add box shadow to edit modal \*/

    z-index: 1000; /\* Set z-index for edit modal \*/

    width: 300px; /\* Set width for edit modal \*/

}

#editModal h2 {

    margin-top: 0; /\* Remove top margin for h2 \*/

    margin-bottom: 10px; /\* Add bottom margin for h2 \*/

}

#editModal form {

    display: flex; /\* Display form as flex container \*/

    flex-direction: column; /\* Set flex direction to column \*/

}

#editModal form label {

    margin-bottom: 5px; /\* Add bottom margin for labels \*/

}

#editModal form input[type="text"],

#editModal form input[type="number"] {

    width: calc(100% - 20px); /\* Set width for inputs with padding \*/

    padding: 8px; /\* Set padding for inputs \*/

    margin-top: 5px; /\* Add top margin for inputs \*/

    margin-bottom: 10px; /\* Add bottom margin for inputs \*/

    border: 1px solid #ccc; /\* Add border to inputs \*/

    border-radius: 5px; /\* Add border radius to inputs \*/

}

#editModal form input[type="submit"] {

    background-color: #28a745; /\* Set background color for submit button \*/

    color: white; /\* Set text color for submit button \*/

    padding: 10px; /\* Set padding for submit button \*/

    border: none; /\* Remove border for submit button \*/

    border-radius: 5px; /\* Add border radius to submit button \*/

    cursor: pointer; /\* Set cursor to pointer for submit button \*/

    transition: background-color 0.3s; /\* Add transition for smoother hover effect \*/

}

#editModal form input[type="submit"]:hover {

    background-color: #218838; /\* Change background color on hover for submit button \*/

}

/\* CSS for the add new Team Form \*/

.sketchc form {

    max-width: 400px; /\* Set maximum width for form \*/

    margin: 0 auto; /\* Center the form horizontally \*/

    padding: 20px; /\* Add padding around the form \*/

    border: 1px solid #ccc; /\* Add border to the form \*/

    border-radius: 5px; /\* Add border radius to the form \*/

    background-color: #f9f9f9; /\* Set background color for the form \*/

}

.sketchc form h3 {

    text-align: center; /\* Center align text for h3 \*/

}

.sketchc form input[type="text"],

.sketchc form input[type="number"],

.sketchc form button {

    width: 100%; /\* Make inputs and button full width \*/

    margin-bottom: 10px; /\* Add bottom margin for inputs and button \*/

    padding: 10px; /\* Add padding to inputs and button \*/

    box-sizing: border-box; /\* Include padding and border in the element's total width \*/

    border: 1px solid #ccc; /\* Add border to inputs and button \*/

    border-radius: 5px; /\* Add border radius to inputs and button \*/

}

.sketchc form input[type="text"]:focus,

.sketchc form input[type="number"]:focus,

.sketchc form button:focus {

    outline: none; /\* Remove default focus outline \*/

    border-color: #007bff; /\* Change border color on focus \*/

}

.sketchc form button {

    background-color: #007bff; /\* Set background color for button \*/

    color: #fff; /\* Set text color for button \*/

    cursor: pointer; /\* Set cursor to pointer for button \*/

    transition: background-color 0.3s; /\* Add transition for smoother hover effect \*/

}

.sketchc form button:hover {

    background-color: #0056b3; /\* Change background color on hover for button \*/

}

**Entryform.php**

**Entryform.php**<!DOCTYPE html>

<!DOCTYPE html>

<html>

<head>

    <title>New Football Team</title>

    <!-- Link to an external stylesheet -->

    <link rel="stylesheet" href="layout.css">

</head>

<body>

    <!-- Header section -->

    <header>

        <h3>CSYM019 - Premier League Results</h3>

    </header>

    <!-- Navigation section -->

    <nav>

        <ul>

            <!-- List item with a link to the 'report.php' page -->

            <li><a href="./report.php">Premier League Report</a></li>

            <!-- List item with a link to the 'entryForm.php' page (current page) -->

            <li><a href="./entryForm.php">Add New Football Team</a></li>

        </ul>

    </nav>

    <!-- Main content section -->

    <main>

        <!-- Heading for the form -->

        <h3>Football Teams Entry Form</h3>

        <!-- Container div -->

        <div class="sketchc">

            <!-- Form element with action set to 'entryForm.php' and method set to POST -->

            <form id="add-team-form" action="entryForm.php" method="POST">

                <!-- Text input for team name -->

                <input type="text" name="team" placeholder="Team Name" required> <br><br>

                <!-- Text input for manager name -->

                <input type="text" name="manager" placeholder="Manager Name" required><br><br>

                <!-- Number input for matches played -->

                <input type="number" name="played" placeholder="Played" required><br><br>

                <!-- Number input for matches won -->

                <input type="number" name="won" placeholder="Won" required><br><br>

                <!-- Number input for matches drawn -->

                <input type="number" name="drawn" placeholder="Drawn" required><br><br>

                <!-- Number input for matches lost -->

                <input type="number" name="lost" placeholder="Lost" required><br><br>

                <!-- Number input for goals scored -->

                <input type="number" name="goalsfor" placeholder="For" required><br><br>

                <!-- Number input for goals conceded -->

                <input type="number" name="against" placeholder="Against" required><br><br>

                <!-- Number input for goal difference -->

                <input type="number" name="gd" placeholder="Goal Difference" required><br><br>

                <!-- Number input for points -->

                <input type="number" name="points" placeholder="Points" required><br><br>

                <!-- Number input for remaining matches -->

                <input type="number" name="remaining" placeholder="Remaining" required><br><br>

                <!-- Submit button -->

                <button type="submit">Submit</button>

            </form>

        </div>

    </main>

    <!-- Footer section -->

    <footer>&copy; CSYM019 2024</footer>

</body>

</html>

<?php

// Check if the form has been submitted using the POST method

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

    // Database connection parameters

    $servername = "localhost";

    $username = "root";

    $password = "";

    $dbname = "league";

    // Create a new MySQLi connection object

    // Referance :  MySQLi Extension Documentation. Available online at: [MySQLi Extension Documentation](https://www.php.net/manual/en/book.mysqli.php)

    $conn = new mysqli($servername, $username, $password, $dbname);

    // Check if the connection was successful

    if ($conn->connect\_error) {

        // If connection fails, terminate script execution and display error message

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

    }

    // Prepare an SQL INSERT statement with placeholders for values

    $stmt = $conn->prepare("INSERT INTO teams (team, played, won, drawn, lost, goalsfor, against, gd, points, manager, remaining) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");

    // Bind parameters to the prepared statement

    $stmt->bind\_param("siiiiiiiisi", $team, $played, $won, $drawn, $lost, $goalsfor, $against, $gd, $points, $manager, $remaining);

    // Set values for the parameters

    $team = $\_POST['team'];

    $manager = $\_POST['manager'];

    $played = $\_POST['played'];

    $won = $\_POST['won'];

    $drawn = $\_POST['drawn'];

    $lost = $\_POST['lost'];

    $goalsfor = $\_POST['goalsfor'];

    $against = $\_POST['against'];

    $gd = $\_POST['gd'];

    $points = $\_POST['points'];

    $remaining = $\_POST['remaining'];

    // Execute the prepared statement

    if ($stmt->execute()) {

        // If execution is successful, display success message

        echo "<p>New team added successfully!</p>";

    } else {

        // If execution fails, display error message with details

        echo "<p>Error: " . $stmt->error . "</p>";

    }

    // Close the prepared statement

    $stmt->close();

    // Close the database connection

    $conn->close();

    // Redirect the user to the 'report.php' page after 2 seconds

    // Referance : PHP Redirect - GeeksforGeeks. Available online at: [PHP Redirect](https://www.geeksforgeeks.org/how-to-redirect-a-page-to-another-page-in-php/)

    header("refresh:2;url=report.php");

}

?>

</body>

<!--

References:

1. HTML5 Documentation - Mozilla Developer Network (MDN). Available online at: [HTML5 Documentation](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5)

2. PHP Header Function - PHP Manual. Available online at: [PHP Header Function](https://www.php.net/manual/en/function.header.php)

3. PHP Redirect - GeeksforGeeks. Available online at: [PHP Redirect](https://www.geeksforgeeks.org/how-to-redirect-a-page-to-another-page-in-php/)

4. MySQLi Extension Documentation. Available online at: [MySQLi Extension Documentation](https://www.php.net/manual/en/book.mysqli.php)

-->

</html>

**Entryform.css**

**Entryform.css**body {

    font-family: Arial, sans-serif; /\* Specifies the font family for the entire document \*/

    background-color: #f2f2f2; /\* Sets the background color of the body \*/

    margin: 0; /\* Removes default margin \*/

    padding: 0; /\* Removes default padding \*/

}

.container {

    max-width: 600px; /\* Sets the maximum width of the container \*/

    margin: 0 auto; /\* Centers the container horizontally \*/

    padding: 20px; /\* Adds padding inside the container \*/

    background-color: #fff; /\* Sets the background color of the container \*/

    border-radius: 8px; /\* Adds rounded corners to the container \*/

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /\* Adds a box shadow to the container \*/

}

h3 {

    text-align: center; /\* Centers the text \*/

    color: #333; /\* Sets the color of the text \*/

}

.form-group {

    margin-bottom: 20px; /\* Adds space between form groups \*/

}

.form-group label {

    display: block; /\* Displays labels as block elements \*/

    font-weight: bold; /\* Sets the font weight of labels to bold \*/

}

.form-group input[type="text"],

.form-group input[type="number"] {

    width: 100%; /\* Sets the width of text and number input fields to 100% \*/

    padding: 8px; /\* Adds padding inside input fields \*/

    border: 1px solid #ccc; /\* Sets the border color and width \*/

    border-radius: 4px; /\* Adds rounded corners to input fields \*/

    box-sizing: border-box; /\* Ensures that padding and border are included in the width \*/

    font-size: 16px; /\* Sets the font size of input fields \*/

}

.form-group input[type="submit"] {

    background-color: #4caf50; /\* Sets the background color of submit button \*/

    color: #fff; /\* Sets the text color of submit button \*/

    padding: 10px 20px; /\* Adds padding to submit button \*/

    border: none; /\* Removes border from submit button \*/

    border-radius: 4px; /\* Adds rounded corners to submit button \*/

    cursor: pointer; /\* Changes cursor to pointer on hover \*/

    font-size: 16px; /\* Sets the font size of submit button \*/

    transition: background-color 0.3s; /\* Adds transition effect to background color \*/

}

.form-group input[type="submit"]:hover {

    background-color: #45a049; /\* Changes background color of submit button on hover \*/

}

**Index.html**

**Index.html**<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8"> <!-- Declares the character encoding for the document -->

    <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- Sets the viewport width and initial scale for mobile responsiveness -->

    <title>Login</title> <!-- Sets the title of the webpage -->

    <link rel="stylesheet" href="login.css"> <!-- Links an external stylesheet -->

</head>

<body>

    <div class="login-container"> <!-- Container for the login form -->

        <h2>Login</h2> <!-- Heading for the login section -->

        <form id="loginForm"> <!-- Login form with unique id -->

            <input type="text" name="username" placeholder="Username" required> <!-- Input field for username -->

            <input type="password" name="password" placeholder="Password" required> <!-- Input field for password -->

            <input type="submit" value="Login"> <!-- Submit button for login -->

        </form>

        <div id="message"></div> <!-- Div to display error message -->

        <div>

            For the DEMO Purpose shown the Login Details <br> <!-- Informational message -->

            Username: admin <br> <!-- Username for demo -->

            Password: admin</div> <!-- Password for demo -->

    </div>

    <script>

        document.getElementById("loginForm").addEventListener("submit", function(event) {

            event.preventDefault(); // Prevent default form submission

            // Get form data

            var formData = new FormData(this);

            // Send AJAX request

            var xhr = new XMLHttpRequest();

            xhr.open("POST", "login.php", true);

            xhr.onreadystatechange = function() {

                if (xhr.readyState == 4 && xhr.status == 200) {

                    var response = xhr.responseText.trim();

                    if (response === "success") {

                        // Redirect to report.php

                        window.location.href = "report.php";

                    } else {

                        document.getElementById("message").innerHTML = "<p style='color: red; text-align: center;'>Invalid username or password</p>";

                    }

                }

            };

            xhr.send(formData);

        });

    </script> <!-- Script to handle form submission and AJAX request -->

</body>

<!-- References:

Mozilla Developer Network. (n.d.). Using FormData Objects. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/FormData/Using\_FormData\_Objects

Mozilla Developer Network. (n.d.). XMLHttpRequest. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest-->

</html>

**Login.php**

**Login.php**<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") { // Checks if the request method is POST

    // Database connection

    $servername = "localhost"; // Server name or IP address

    $username = "root"; // Username for database connection

    $password = ""; // Password for database connection

    $dbname = "league"; // Database name

    $conn = new mysqli($servername, $username, $password, $dbname); // Establishes a new MySQLi connection

    // Check connection

    if ($conn->connect\_error) { // Checks if the connection failed

        die("Connection failed: " . $conn->connect\_error); // Stops script execution and outputs error message

    }

    // Get user input

    $username = $\_POST['username']; // Retrieves username from POST data

    $password = $\_POST['password']; // Retrieves password from POST data

    // Use prepared statements to prevent SQL injection

    $stmt = $conn->prepare("SELECT \* FROM users WHERE username = ? AND password = ?"); // Prepares SQL query with placeholders

    $stmt->bind\_param("ss", $username, $password); // Binds parameters to the prepared statement

    $stmt->execute(); // Executes the prepared statement

    $result = $stmt->get\_result(); // Gets the result set from the executed statement

    if ($result->num\_rows == 1) { // Checks if there is exactly one row in the result set

        // Login successful, send "success"

        echo "success"; // Outputs "success" indicating successful login

    } else {

        // Invalid credentials, send "error"

        echo "error"; // Outputs "error" indicating invalid credentials

    }

    $stmt->close(); // Closes the prepared statement

    $conn->close(); // Closes the database connection

}

// References:

//PHP Manual. (n.d.). Prepared Statements. Retrieved from https://www.php.net/manual/en/mysqli.quickstart.prepared-statements.php

//PHP Manual. (n.d.). MySQLi. Retrieved from https://www.php.net/manual/en/book.mysqli.php

?>

**Login.css**body {

    font-family: Arial, sans-serif; /\* Sets the font family for the entire document \*/

    display: flex; /\* Specifies the flexible layout model for the body \*/

    justify-content: center; /\* Aligns the content along the main axis (horizontally centered) \*/

    align-items: center; /\* Aligns the content along the cross axis (vertically centered) \*/

    height: 100vh; /\* Sets the height of the body to 100% of the viewport height \*/

    background-color: #f4f4f4; /\* Sets the background color of the body \*/

}

.login-container {

    background-color: #fff; /\* Sets the background color of the login container \*/

    padding: 20px; /\* Adds padding inside the login container \*/

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /\* Adds a box shadow to the login container \*/

    border-radius: 5px; /\* Sets the border radius of the login container \*/

}

.login-container h2 {

    margin-bottom: 20px; /\* Adds margin at the bottom of the heading \*/

    text-align: center; /\* Aligns the heading text to the center \*/

}

.login-container input[type="text"],

.login-container input[type="password"] {

    width: 100%; /\* Sets the width of the input fields to 100% \*/

    padding: 10px; /\* Adds padding inside the input fields \*/

    margin: 10px 0; /\* Sets margin at the top and bottom of the input fields \*/

    border: 1px solid #ccc; /\* Sets the border style of the input fields \*/

    border-radius: 5px; /\* Sets the border radius of the input fields \*/

}

.login-container input[type="submit"] {

    width: 100%; /\* Sets the width of the submit button to 100% \*/

    padding: 10px; /\* Adds padding inside the submit button \*/

    background-color: #5cb85c; /\* Sets the background color of the submit button \*/

    border: none; /\* Removes the border of the submit button \*/

    border-radius: 5px; /\* Sets the border radius of the submit button \*/

    color: #fff; /\* Sets the text color of the submit button \*/

    font-size: 16px; /\* Sets the font size of the submit button text \*/

    cursor: pointer; /\* Changes the cursor to a pointer on hover \*/

}

.login-container input[type="submit"]:hover {

    background-color: #4cae4c; /\* Changes the background color of the submit button on hover \*/

}

#message {

    margin-top: 20px; /\* Adds margin at the top of the message container \*/

    text-align: center; /\* Aligns the text inside the message container to the center \*/

}

**REFERENCES**

**1.** PHP Official Documentation. Available online at: [PHP Official Documentation](https://www.php.net/manual/en/)

**2.** MySQLi Extension Documentation. Available online at: [MySQLi Extension Documentation](https://www.php.net/manual/en/book.mysqli.php)

**3.** Exception Handling in PHP. Available online at: [Exception Handling in PHP](https://www.php.net/manual/en/language.exceptions.php)

**4.** PHP Header Function. Available online at: [PHP Header Function](https://www.php.net/manual/en/function.header.php)

**5.** PHP Arrays. Available online at: [PHP Arrays](https://www.php.net/manual/en/language.types.array.php)

**6.** Chart.js library

**7.** [HTML5 Documentation](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5)

**8.** PHP Redirect - GeeksforGeeks. Available online at: [PHP Redirect](https://www.geeksforgeeks.org/how-to-redirect-a-page-to-another-page-in-php/)

**TESTINGS**Testing of this application was done using manual testing using following techniques.

**Create (Add):**

Navigate to the "Add" page of the application.

Fill in the required fields in the form.

Submit the form.

Check if the data appears in the list or if there's a success message indicating the record was added.

**Read (View):**

Navigate to the "Read" or "View" page of the application.

Verify that all existing records are displayed.

Click on a specific record to view its details.

Ensure that the details displayed match the information entered previously.

**Update (Edit):**

Click to the "Edit" button of the application.

Select a record to edit.

Modify the fields want to change.

Submit the form.

Check if the changes are reflected in the database and if there's a success message.

**Delete:**

Navigate to the "Delete" button of the application.

Select a record to delete.

Confirm the deletion.

Check if the record has been removed from the database and if there's a success message.

**Validation:**

Test the validation by submitting forms with invalid or missing data.

Verify that appropriate error messages are displayed.

Ensure that the form data is not submitted if validation fails.

**Concurrency:**

Open the application in multiple browser tabs or windows.

Perform CRUD operations simultaneously in different tabs.

Verify that the application handles concurrent requests properly, avoiding data corruption or conflicts.

**Security:**

Attempt to access CRUD operations without proper authentication.

Test for SQL injection by entering malicious input into form fields.

Verify that the application handles these scenarios securely, such as by redirecting unauthorized users or sanitizing input.