­­­

|  |
| --- |
| Arden University |
| Computing |
| Web Application Development |
| Web Application Development |
|  |
|  |
| 2936 |

Table of Contents

[GAME OVERVIEW 3](#_Toc190843481)

[System Design and Architecture 5](#_Toc190843482)

[DATABASE(SQL) 5](#_Toc190843483)

[FRONTEND 6](#_Toc190843484)

[BACKEND 7](#_Toc190843485)

[Game Logic and API Interaction 8](#_Toc190843486)

[CONCLUTION 9](#_Toc190843487)

[REFERENCE 9](#_Toc190843488)

[APPENDIX 10](#_Toc190843489)

[DATABASE 10](#_Toc190843490)

[HTML 10](#_Toc190843491)

[CSS 14](#_Toc190843492)

[JAVASCRIPT 18](#_Toc190843493)

[PHP 22](#_Toc190843494)

# GAME OVERVIEW

Movie Guesser is a web-based game that challenges players to guess movie titles using hints. The game starts when the player clicks the "Play the Game" button, which offers two options: "Enter the Game" and "Delete User." Players can enter their name to participate and have the option to remove their entry later if needed.

The game revolves around selecting a random movie card and guessing the correct title with the help of hints like release year, runtime, cast, genre, and director. The scoring system is designed to reward accuracy—players earn 6 points for guessing correctly on the first try, while additional hints gradually lower the score, making the game both challenging and engaging.

From a technical perspective, Movie Guesser is developed using HTML, CSS, JavaScript, and PHP, with MySQL handling user data. Players enter their names through a web form, and their data is stored in a database for future sessions. If a user wishes to leave the game, their entry can be deleted via a DELETE request, ensuring smooth database management.

The gameplay logic is powered by JavaScript, which manages random movie selection, hint displays, and score calculations. Meanwhile, PHP handles backend tasks, including storing user details, retrieving movie information, and processing scores. Instead of a separate score page, scores are instantly displayed whenever a player submits the correct answer.

This combination of frontend interactivity and backend functionality ensures a seamless, fun, and competitive gaming experience, keeping players entertained while testing their movie knowledge.

Website URL: <https://stu146134.webhosting.arden.ac.uk/rose/>

1.1 SREENSHOT OF THE APPLICATION

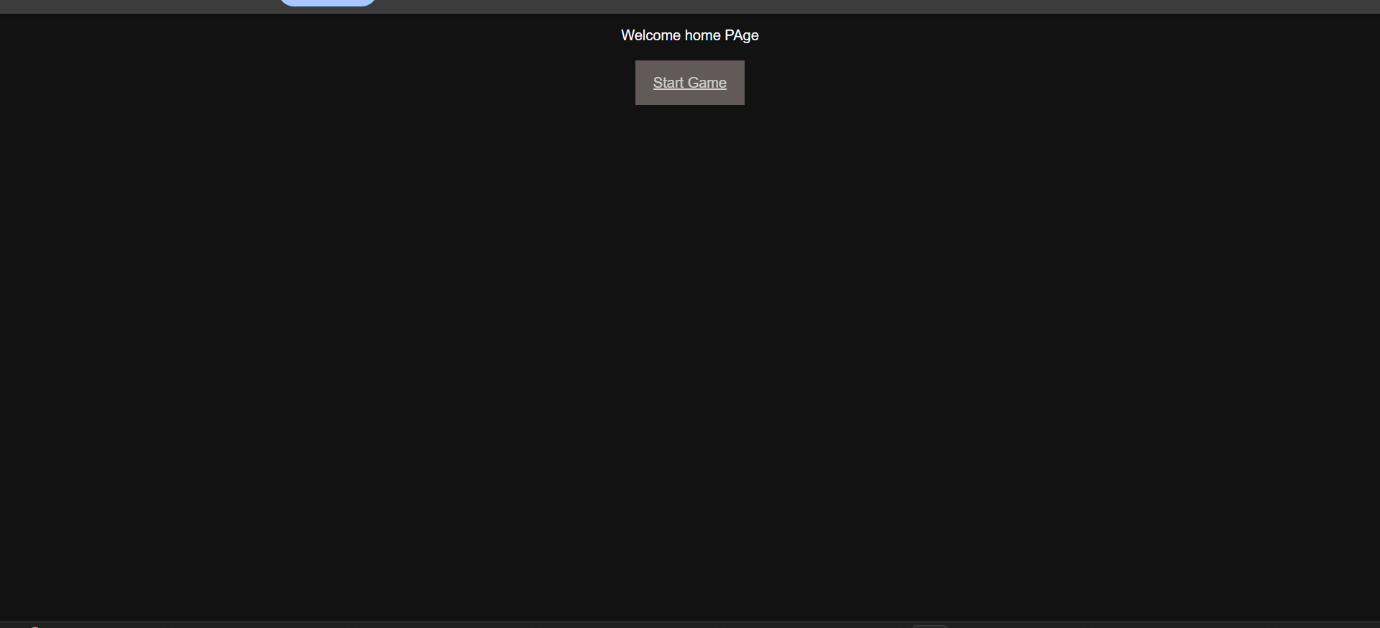


Fig1.1

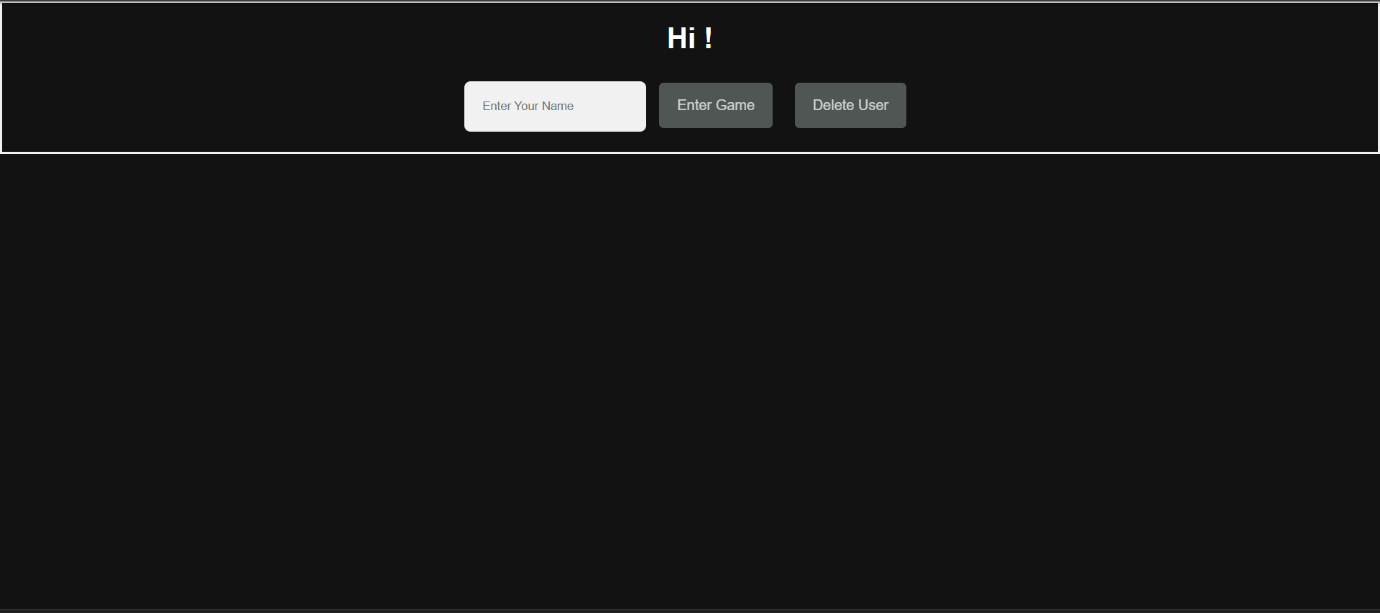


Fig 1.2

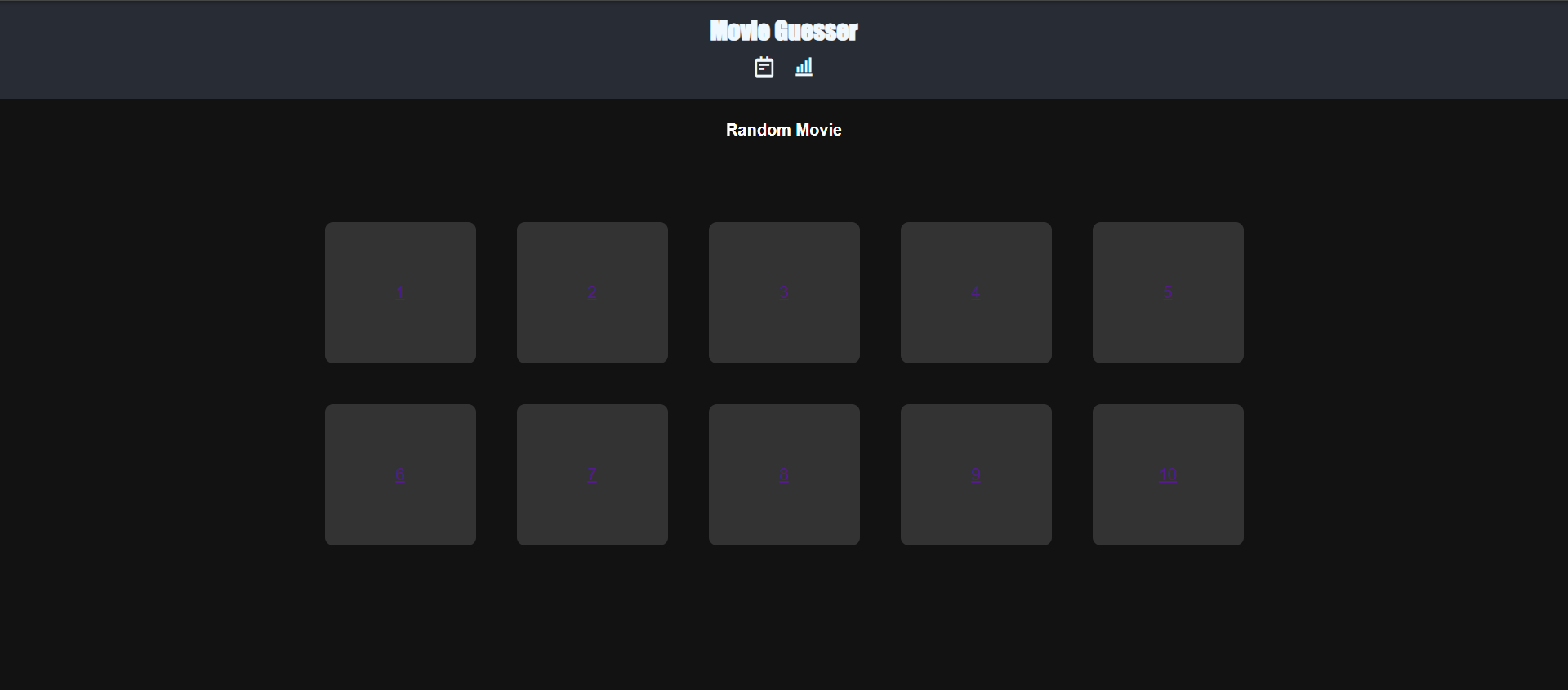


Fig 1.3

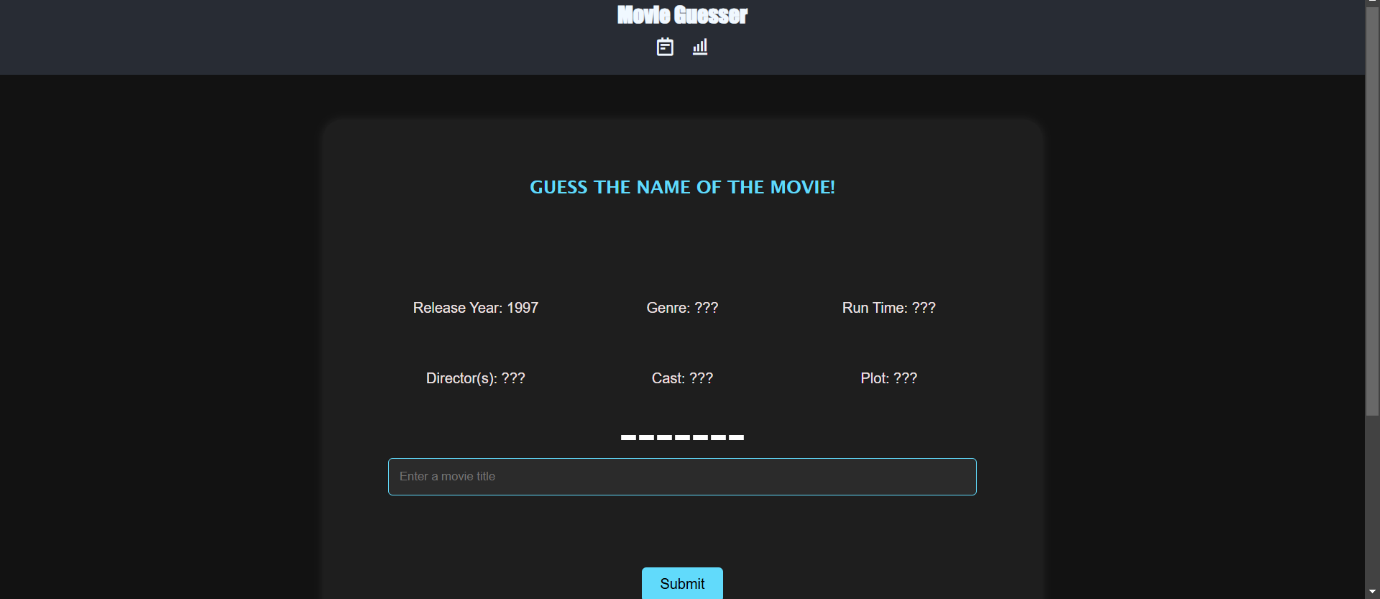


Fig 1.4

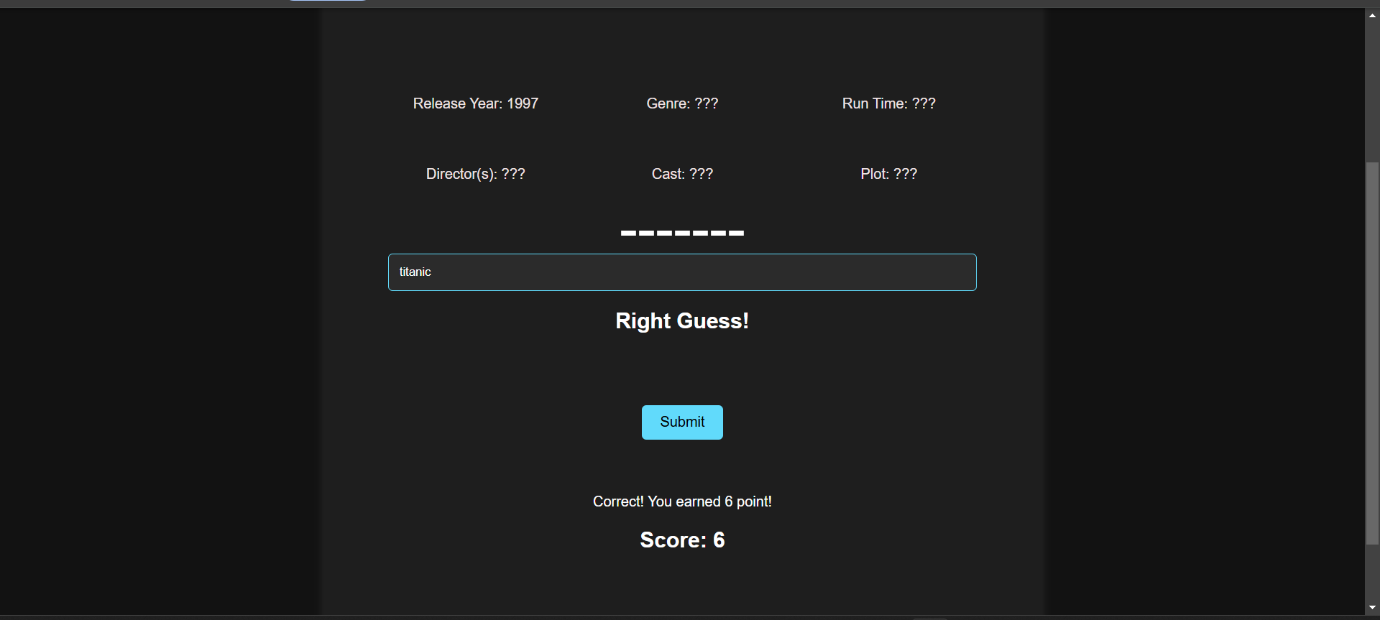
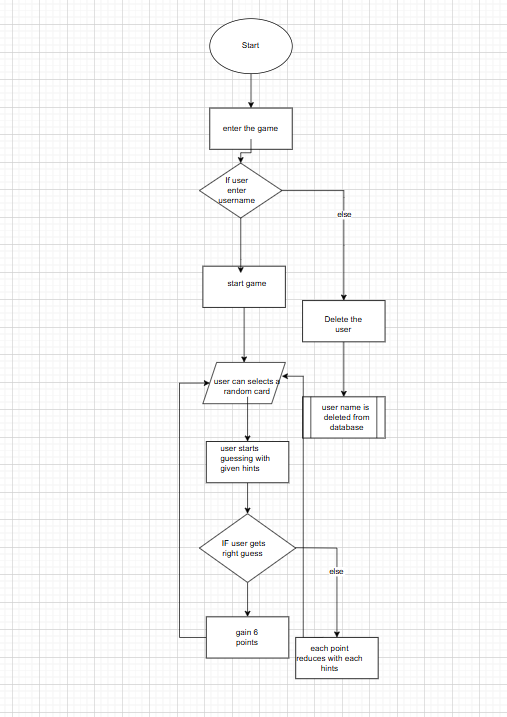


Fig 1.5

# System Design and Architecture

## DATABASE(SQL)

The WAD database has been set up, and a table called "movies" has been created. The columns for the table have been defined, and movie data has been added successfully. All usernames are stored in the WAD database. Below is a flowchart that shows the workflow of the application.



## FRONTEND

Languages Used:

- HTML – Used for structuring the web pages.

- CSS – Used for styling and designing the application.

- JavaScript – Used for interactivity and handling user inputs.

Application Overview:

This is a movie guessing game where users interact with various pages to enter their usernames, select random movies, and make guesses based on provided hints. The design is minimalistic, ensuring an easy-to-use experience.

Files and Their Roles:

- HTML Files:

- index.html – The main homepage of the application.

- userform.html – A page where users enter their username.

- dltuser.html – A page for users to delete their username.

- getMovie.html – Displays a random movie selection with clickable movie boxes.

- guessMovie.html – The page where users make their guesses based on hints.

- CSS File:

- style.css – Contains all design and style elements for the application.

- JavaScript Files:

- userform.js – Manages user interactions when entering their name.

- dltuser.js – Handles the deletion of a username from the system.

- script.js – Manages the functionality for selecting random movie boxes.

- guessMovie.js – Controls the guessing logic when users try to guess the movie.

## BACKEND

Technology Used:

- PHP– The backend language used to manage server-side logic and interact with the database.

Backend Overview:

The backend is responsible for the core functionality of the application. It manages database operations and handles user requests. It interacts with the WAD database to store, retrieve, and delete data as needed by the frontend.

HTTP Methods Used:

- GET – Used to fetch data from the database (e.g., retrieving movie information).

- POST – Used to insert new data into the database (e.g., saving a username).

- DELETE – Used to remove data from the database (e.g., deleting a user).

PHP Files and Their Roles:

- userForm.php – Handles saving user data into the database using the POST method.

- dltuser.php – Deletes a username from the database using the DELETE method.

- getMovies.php – Retrieves movie data from the database using the GET method.

# Game Logic and API Interaction

The movie guessing game follows a clear flow: users enter their username, choose a random movie, and then try to guess it using the hints provided. The backend PHP files handle database interactions, while the Fetch API in JavaScript ensures smooth communication with the server.

1. Entering the Username (POST Method)

When a user enters their username, a POST request is sent to userForm.php.

The PHP script collects the username from the request and stores it in the database.

This allows the game to recognize the user throughout the session.

2. Deleting a Username (DELETE Method)

If the user wants to delete their stored username, a DELETE request is sent to dltuser.php.

The PHP script processes the request and removes the username from the database, keeping the data clean and up-to-date.

3. Selecting a Random Movie (GET Method)

After the username is saved, the user can select a random movie.

A GET request is sent to getMovies.php, which retrieves all available movies.

A random movie is displayed to the user, and they are asked to guess it based on the provided hints.

How the Fetch API Works

The Fetch API is a modern JavaScript feature that allows the app to communicate with the server without reloading the page. It enables the game to send, retrieve, and delete data asynchronously, offering a smoother user experience.

# CONCLUTION

The Guess the Movie application is an exciting and interactive project that blends frontend and backend technologies to create an engaging movie-guessing experience. Built using HTML, CSS, JavaScript, PHP, and SQL, the game allows users to guess movie titles based on limited hints, with more details revealed as they progress.

A MySQL database efficiently stores key movie attributes, including movie name, release year, genre, runtime, cast, plot, and director, ensuring smooth data management and retrieval. PHP scripts and SQL queries dynamically fetch movie data, which is then displayed on the frontend for user interaction.

The user interface is designed to be modern and intuitive, offering a clean and engaging layout. JavaScript event listeners handle user interactions efficiently, managing form submissions and real-time updates. To ensure a smooth gameplay experience, challenges like efficient database queries, dynamic content handling, and user input validation were tackled using structured programming and optimized SQL queries.

This project showcases a well-balanced full-stack development approach, ensuring smooth functionality and an engaging experience. This application provides users with a fun and interactive platform to test their movie knowledge while demonstrating the power of web development technologies.

# REFERENCE

<https://www.w3schools.com/Php/php_superglobals_post.asp>

<https://freefrontend.com/css-buttons/>

<https://freefrontend.com/css-buttons/>

# APPENDIX

## DATABASE

CREATE TABLE movies (

id INT AUTO\_INCREMENT PRIMARY KEY,

movie\_name VARCHAR(255) NOT NULL,

release\_year INT NOT NULL,

genre VARCHAR(100) NOT NULL,

runtime VARCHAR(50) NOT NULL,

cast TEXT NOT NULL,

plot TEXT NOT NULL,

director VARCHAR(255) NOT NULL

);

## HTML

1. Index.html
2. <!DOCTYPE html>
3. <html>
4. <head>
5. <link rel="stylesheet" href="style.css">
6. </head>
7. <body>
8. <p>Welcome home Page</p><br>
9. <a class="start-game-link" href="userForm.html">Start Game<i></i></a>
10. </body>
11. </html>

2. userForm.html

<!DOCTYPE html>

<html lang="en">

  <head>

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

  </head>

  <body>

    <div class="user-form-popup" id="myForm">

    <form id="userForm" class="user-form-container">

        <h1>Hi !</h1>

        <input  class="user-text-box" type="text"  pattern="[a-zA-Z]+"placeholder="Enter Your Name" name="name"required>

        <button type="submit" class="user-form-btn">Enter Game</button>

        <button type="button" class="user-form-btn" id="deleteBtn">Delete User</button>

      </form>

    </div>

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

  </body>

</html>

3 .dltuser.html

<!DOCTYPE html>

<html lang="en">

  <head>

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

  </head>

  <body>

    <div class="user-form-popup" id="myForm">

    <form id="userForm" class="user-form-container">

        <h1>Hi !</h1>

        <input  class="user-text-box" type="text"  pattern="[a-zA-Z]+"placeholder="Enter Your Name" name="name"required>

        <button type="submit" class="user-form-btn">Enter Game</button>

        <button type="button" class="user-form-btn" id="deleteBtn">Delete User</button>

      </form>

    </div>

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

    </body>

  </html>

4.getMovies.html

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">
6. <title>Guess the Movie Game</title>
7. <link rel="stylesheet" href="style.css">
8. <script src="script.js"></script>
9. <link rel="stylesheet" href="https://unpkg.com/boxicons@latest/css/boxicons.min.css">
10. </head>
11. <body>
12. <header class="header">
13. <div class="app-name">Movie Guesser</div>
14. <div class="nav-options">
15. <a href="guessMovie.html"><i class="bx bx-notepad"></i></a>
16. <a href="score.html"><i class="bx bx-bar-chart-alt"></i></a>
17. </div>
18. </header>
19. <h4>Random Movie</h4>
20. <div class="movie-grid"id="random-movie"></div>
21. </body>
22. </html>

5. guessMovie.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Guess the Movie Game</title>

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

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

    <link rel="stylesheet" href="https://unpkg.com/boxicons@latest/css/boxicons.min.css">

</head>

<body>

    <header class="header">

        <div class="app-name">Movie Guesser</div>

        <div class="nav-options">

            <a href="getMovies.html"><i class="bx bx-notepad"></i></a>

            <a href="score.html" id="score-link"><i class="bx bx-bar-chart-alt"></i></a>

        </div>

    </header>

    <div id="game-container">

      <h1>Guess the Name of the Movie!</h1><br>

      <div class="guess-grid"id="guess">

        <div id="Release\_Year"></div>

        <div id="Genre"></div>

        <div id="Run\_Time"></div>

        <div id="Director"></div>

        <div id="Cast"></div>

        <div id="Plot"></div>

    </div>

        <div id = "dashed-movie-name"></div>

        <input type="text" id="guess-input" placeholder="Enter a movie title">

        <div id="guess-count"><h2 id="guess-remaining"></h2></div>

        <button class="btnSubmit" id="guess-movie-submit">Submit</button>

        <div id="result"></div>

        <h2>Score: <span id="score"></span></h2>

    </div>

</body>

</html>

## CSS

1. body {
2. font-family: Arial, sans-serif;
3. background-color: #121212;
4. color: white;
5. text-align: center;
6. margin: 0;
7. padding: 0;
8. }
10. .header {
11. display: flex;
12. flex-direction: column;
13. align-items: center;
14. padding: 15px;
15. background-color: #282c34;
16. color: white;
17. text-align: center;
18. }
20. .app-name {
21. font-size: 24px;
22. font-weight: bold;
23. color:aliceblue;
24. margin-bottom: 10px;
25. font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
26. }
28. .nav-options {
29. font-size: 1.5rem;
30. display: flex;   /\* Aligns icons in a row \*/
31. justify-content: center;  /\* Centers them horizontally \*/
32. align-items: center;  /\* Centers them vertically \*/
33. gap: 15px;  /\* Adds spacing between icons \*/
34. color: white;
35. }
36. .nav-options i{
37. color: aliceblue;
38. }
40. #game-container {
41. background-color: #1e1e1e;
42. padding: 50px;
43. margin: 50px auto;
44. width: 700px;
45. border-radius: 20px;
46. text-align: center;
47. box-shadow: 0px 0px 10px rgba(255, 255, 255, 0.1);
48. }
50. #game-container h1 {
51. font-size: 20px;
52. color: #61dafb;
53. font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
54. text-transform: uppercase;
55. margin-bottom: 60px;
56. }
57. #game-container h2{
58. margin-bottom: 70px;
59. }
61. #game-container p {
62. font-size: 16px;
63. font-family:system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
64. margin-bottom: 30px;
65. }
67. #guess-input {
68. width: 90%;
69. padding: 12px;
70. margin-top: 10px;
71. border-radius: 5px;
72. border: 1px solid #61dafb;
73. background: #2b2b2b;
74. color: white;
75. }
77. button {
78. padding: 10px 20px;
79. margin: 10px;
80. border: none;
81. border-radius: 5px;
82. font-size: 16px;
83. cursor: pointer;
84. transition: 0.3s;
85. }
87. .btnSubmit {
88. background-color: #61dafb;
89. color: black;
90. margin-bottom: 60px;
91. }
93. button:hover {
94. opacity: 0.8;
95. }
96. /\* Movie Grid \*/
97. .movie-grid {
98. display: grid;
99. grid-template-columns: repeat(auto-fit, minmax(120px, 1fr)); /\* Adjust columns \*/
100. gap: 40px;
101. padding: 60px;
102. max-width: 900px;
103. margin: auto;
104. color: rgb(11, 1, 1);
105. }

108. .movie-box {
109. background: #333;
110. padding: 60px;
111. border-radius: 8px;
112. cursor: pointer;
113. transition: 0.3s;
114. text-align: center;
115. }
117. .movie-box:hover {
118. background: #555;
119. }
121. .score-container {
122. background-color: #2b2b2b;
123. padding: 20px;
124. margin-top: 20px;
125. border-radius: 10px;
126. box-shadow: 0px 0px 10px rgba(255, 255, 255, 0.1);
127. color: white;
128. }
130. .score-item {
131. background: #333;
132. padding: 10px;
133. margin: 10px 0;
134. border-radius: 8px;
135. text-align: left;
136. }
138. .score-item p {
139. margin: 0;
140. font-size: 16px;
141. }
142. #dashed-movie-name{
143. font-size: 30px;
144. transform: scale(2);
145. }
146. .guess-grid {
147. display: grid;
148. grid-template-columns: repeat(auto-fit, minmax(120px, 1fr)); /\* Adjust columns \*/
149. gap: 60px;
150. padding: 35px;
151. max-width: 900px;
152. margin: auto;
153. color: rgb(238, 227, 227);
154. }
155. .start-game-link{
156. background-color: #756a6a;
157. color: white;
158. padding: 16px 20px;
159. border: none;
160. cursor: pointer;
161. opacity: 0.8;
162. bottom: 23px;
163. right: 28px;
164. width: 280px;
165. }
166. .user-form-popup {
167. bottom: 0;
168. right: 15px;
169. border: 3px solid #f1f1f1;
170. }
171. .user-form-container .user-form-btn {
172. background-color: #5e6763;
173. color: white;
174. padding: 16px 20px;
175. border: none;
176. cursor: pointer;
177. margin-bottom:10px;
178. opacity: 0.8;
179. }
180. .user-form-container .user-form-btn:hover{
181. opacity: 1;
182. }
183. .user-text-box{
184. padding: 20px;
185. margin: 5px 0 22px 0;
186. border: none;
187. background: #f1f1f1;
188. border-radius: 7px;
189. }
190. .basic-box {
191. width: 200px;
192. height: 100px;
193. border: 2px solid #000;
194. background-color: #f0f0f0;
195. padding: 10px;
196. color: rgb(86, 90, 86);
197. display: inline-block;
198. }

## JAVASCRIPT

1. userform.js
2. document.addEventListener("DOMContentLoaded", function() {
3. const form = document.getElementById("userForm");
4. if (form) {
5. form.addEventListener("submit", function(event) {
6. event.preventDefault();
7. const formData = new FormData(this);
8. fetch('userForm.php', {
9. method: 'POST',
10. body: formData
11. })
12. .then(response => response.json())
13. .then(data => {
14. console.log(data);
15. window.location.href = data.redirect;
16. })
17. .catch(error => console.error("Error:", error));
18. });
19. } else {
20. console.error("Form with ID 'userForm' not found.");
21. }
22. });

2. dltuser.js

document.addEventListener("DOMContentLoaded", function () {

    const deleteBtn = document.getElementById("deleteBtn");

    if (deleteBtn) {

        deleteBtn.addEventListener("click", function () {

            console.log("Delete button clicked!");

            const name = document.querySelector("input[name='name']").value.trim(); // Trim whitespace

            if (name) {

                console.log("Deleting user:", name);

                // Send the name in the request body

                fetch('dltuser.php', {

                    method: 'DELETE',

                    headers: {

                        'Content-Type': 'application/json',

                    },

                    body: JSON.stringify({ name: name }), // Send name in the body

                })

                    .then(response => response.json())

                    .then(data => {

                        console.log(data);

                        alert(data.message);

                        if (data.redirect) {

                            window.location.href = data.redirect; // Redirect if needed

                        }

                    })

                    .catch(error => {

                        console.error("Error deleting user:", error);

                        alert("An error occurred while deleting the user.");

                    });

            } else {

                alert("Please enter a name to delete.");

            }

        });

    } else {

        console.error("Delete button not found.");

    }

});

3. script.js

fetch('getMovies.php')

    .then(response => response.json())

    .then(movies => {

        const randomMovie = document.getElementById('random-movie');

        movies.forEach(movie => {

            const div = document.createElement('div');

            div.className = "movie-box";

            const a = document.createElement('a');

            a.textContent = movie.Id;

            a.href = "guessMovie.html";

            a.addEventListener('click', () => {

                sessionStorage.setItem('selectedMovie', JSON.stringify(movie));

            });

            div.appendChild(a);

            randomMovie.appendChild(div);

        });

    })

    .catch(error => console.error('Error fetching tasks:', error));

4. guessMovie.js

let selectedMovieName = "";

let selectedMovieId ="";

let guessCount = 5;

let score = 6;

let played = 0;

function updateGameStats() {

    sessionStorage.setItem("gameStats", JSON.stringify({ score: score, played: played }));

    console.log(score,played);

}

document.addEventListener("DOMContentLoaded", () => {

    const movieData = JSON.parse(sessionStorage.getItem('selectedMovie'));

    selectedMovieName = `${movieData.Movie\_Name}`;

    document.getElementById('Release\_Year').textContent = `Release Year: ${movieData.Release\_Year}`;

    document.getElementById('Genre').textContent = "Genre: ???";

    document.getElementById('Run\_Time').textContent = "Run Time: ???";

    document.getElementById('Director').textContent = "Director(s): ???";

    document.getElementById('Cast').textContent = "Cast: ???";

    document.getElementById('Plot').textContent = "Plot: ???";

                    let hiddenTitle = movieData.Movie\_Name

                    .split(' ')

                    .map(word => '-'.repeat(word.length))

                    .join(' ');

                    document.getElementById('dashed-movie-name').textContent = `${hiddenTitle}`;

document.getElementById('guess-movie-submit').addEventListener('click',function(){

    const movieName = document.getElementById('guess-input').value.trim();

    if(movieName.toLowerCase() == selectedMovieName.toLowerCase()){

        const correctGuess = document.getElementById('guess-remaining');

        correctGuess.innerText = "Right Guess!"

        document.getElementById("result").textContent = `Correct! You earned ${score} point!`;

        document.getElementById('score').textContent = score;

        this.disabled = true;

        played++;

        console.log(played);

        updateGameStats();

    }

    else{

        const count = document.getElementById('guess-remaining');

        switch (guessCount) {

            case 5:

                document.getElementById('Genre').textContent = `Genre: ${movieData.Genre}`;

                guessCount--;

                score--;

                count.innerText =  `${guessCount} Guesses Remaining`;

                updateGameStats();

                break;

            case 4:

                document.getElementById('Run\_Time').textContent = `Run Time: ${movieData.Run\_Time}`;

                guessCount--;

                score--;

                count.innerText =  `${guessCount} Guesses Remaining`;

                updateGameStats();

                break;

            case 3:

                document.getElementById('Director').textContent = `Director(s): ${movieData.Director}`;

                guessCount--;

                score--;

                count.innerText =  `${guessCount} Guesses Remaining`;

                updateGameStats();

                break;

            case 2:

                document.getElementById('Cast').textContent = `Cast: ${movieData.Cast}`;

                guessCount--;

                score--;

                count.innerText =  `${guessCount} Guesses Remaining`;

                updateGameStats();

                break;

            case 1:

                document.getElementById('Plot').textContent = `Plot: ${movieData.Plot}`;

                guessCount--;

                score--;

                count.innerText = "Maximum guesses reached!"

                document.getElementById("result").textContent = `Wrong! The correct answer was "${movieData.Movie\_Name}".`;

                this.disabled = true;

                updateGameStats();

                break;

            case 0:

                break;

        }

    }

})

});

## PHP

1. userForm.php
2. <?php
3. // Enable error reporting for debugging
4. error\_reporting(E\_ALL);
5. ini\_set('display\_errors', 1);
6. header('Content-Type: application/json');
7. // Database credentials
8. $servername = "localhost";
9. $username = "y2jfwcz\_WAD";
10. $password = "Thrisha22";
11. $dbname = "y2jfwcz\_WAD";
12. // Connect to database
13. $conn = mysqli\_connect($servername, $username, $password, $dbname);
14. // Check connection
15. if (!$conn) {
16. die(json\_encode(["error" => "Database connection failed: " . mysqli\_connect\_error()]));
17. }
18. // Handle form submission
19. if ($\_SERVER["REQUEST\_METHOD"] == "POST") {
20. $name = $\_POST['name'] ?? '';
21. // Validate input
22. if (empty($username)) {
23. die(json\_encode(["error" => "Username is required"]));
24. }
25. // Prevent SQL Injection
26. $username = mysqli\_real\_escape\_string($conn, $name);
27. // Insert username into database
28. $sql = "INSERT INTO users (name) VALUES ('$name')";
29. if (mysqli\_query($conn, $sql)) {
30. echo json\_encode(["success" => true, "redirect" => "getMovies.html"]);
31. } else {
32. echo json\_encode(["error" => "Error: " . mysqli\_error($conn)]);
33. }
34. }
35. // Close connection
36. mysqli\_close($conn);
37. ?>

3. dltuser.php

<?php

// Enable error reporting for debugging

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

header('Content-Type: application/json');

// Database credentials

$servername = "localhost";

$username = "y2jfwcz\_WAD";

$password = "Thrisha22";

$dbname = "y2jfwcz\_WAD";

// Connect to database

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

if ($\_SERVER["REQUEST\_METHOD"] == "DELETE") {

    // Get the JSON input

    $input = json\_decode(file\_get\_contents("php://input"), true);

    $name = $input['name'] ?? '';

    // Validate input

    if (empty($name)) {

        die(json\_encode(["error" => "Username is required"]));

    }

    // Use prepared statements to prevent SQL injection

    $stmt = $conn->prepare("DELETE FROM users WHERE name = ?");

    $stmt->bind\_param("s", $name);

    if ($stmt->execute()) {

        echo json\_encode(["success" => true, "message" => "User '$name' deleted successfully", "redirect" => "index.html"]);

    } else {

        echo json\_encode(["error" => "Error deleting user: " . $stmt->error]);

    }

    $stmt->close();

} else {

    echo json\_encode(["error" => "Invalid request method"]);

}

// Close connection

$conn->close();

?>

4. getMovies.php

<?php

$servername = "localhost";

$username = "y2jfwcz\_WAD";

$password = "Thrisha22";

$dbname = "y2jfwcz\_WAD";

// Create connection

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

   // Check connection

   if ($conn->connect\_error) {

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

   }

   // Fetch tasks from the database

   $result = $conn->query("SELECT \* FROM movies");

   $tasks = [];

   // Store tasks in an array

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

       $tasks[] = $row;

   }

   // Return tasks as JSON

   echo json\_encode($tasks);

   $conn->close();

   ?>