



PHP Hangman Game


Project Presentation

Team Name:

PHP GAME CRAFTERS

Team members:

Walid Abdullahi(Leader),
Stuart Idehen

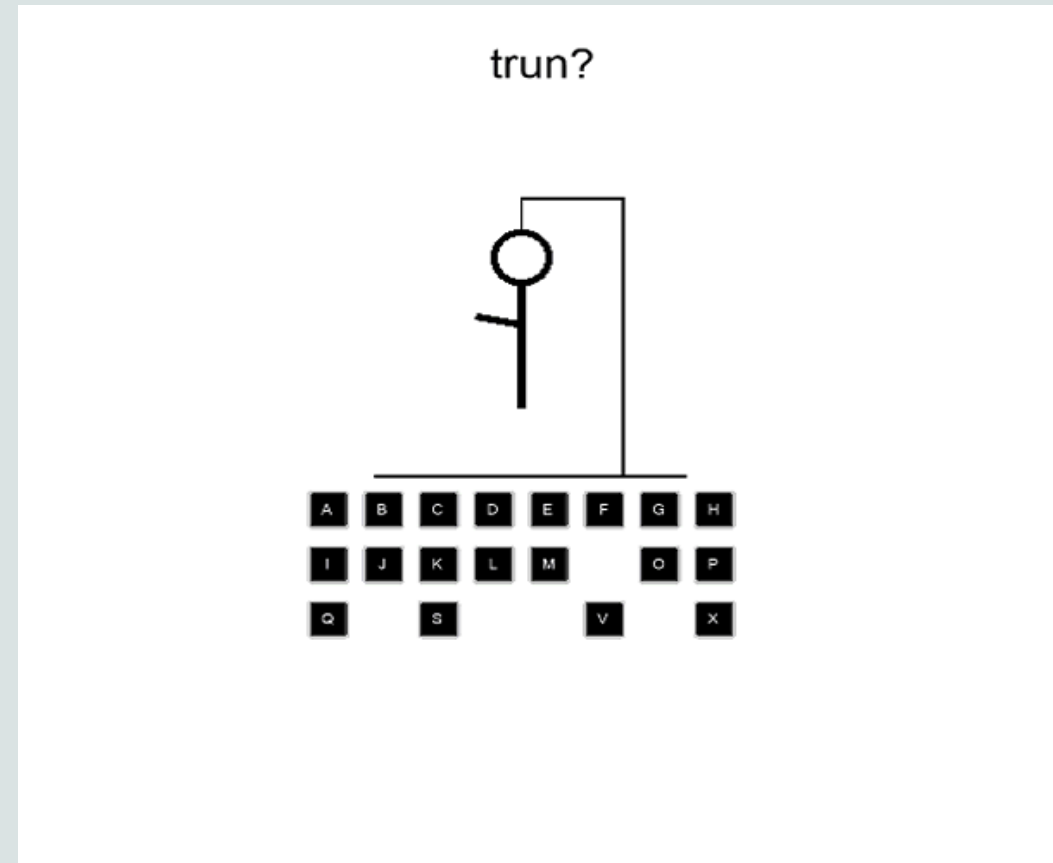


Team Information

- **Team Members and Roles:**
 - Walid Abdullahi: **Frontend Design & CSS** - Designed the game interface (user input areas, scoreboard, etc.).Crafted the UI layout and ensured mobile responsiveness.
 - Walid Abdullahi: **PHP Game Logic & Session Management** - Developed core game mechanics.
 - Stuart Idehen: **User Authentication System** - Implemented secure user login, registration, and session handling.
 - Stuart Idehen: **Leaderboard & Score System** - Built the leaderboard and score-tracking functions.

Problem Statement & Objectives

- **Problem Statement:** While classic games like Hangman are widely loved, modern adaptations often lack enhanced functionality, such as user management and score tracking, which makes them less engaging.
- **Objectives:** To create a multi-difficulty Hangman game with secure user authentication, session management, and real-time leaderboard ranking for competitive engagement.



Project Overview

- Classic word-guessing game with modern features
- Multiple difficulty levels
- User authentication system with secure registration and login
- Real-time score tracking
- Persistent leaderboard
- Mobile-responsive design

Key Features

- User Management
 - Secure registration system
 - Password hashing
 - Session management
 - Remember me functionality
- Game Mechanics
 - Three difficulty levels:
 - Easy (3-4 letter words)
 - Medium (5-7 letter words)
 - Hard (8+ letter words)
 - 6 lives per game
 - Score based on remaining lives
- Leaderboard System
 - Top 10 scores tracking
 - Real-time updates

Architecture and Design

- **Application Architecture:**
 - login.php: User login, session management, "remember me"
 - register.php: User registration, password hashing
 - game.php: Game logic, word selection, guessing mechanics
 - leaderboard.php: Leaderboard display and score updates
 - menu.php: Main navigation menu
 - logout.php: User logout and session termination
 - rules.php: Information and rules for gameplay
 - style.css: Custom CSS styles and UI enhancements

Kanban Method in Project Management

- **What is Kanban?**
 - A visual workflow management method that helps track project progress. Focuses on continuous delivery, maximizing efficiency, and managing work in progress (WIP).
- **Benefits of Kanban:**
 - **Clear Task Visibility:** Enabled us to track the entire project and see individual progress.
 - **Improved Collaboration:** Allowed the team to stay aligned and avoid task overlap.
 - **Efficiency:** Helped manage workload effectively by keeping focus on key tasks.
- **How We Applied Kanban:**
 - **Task Breakdown:**
 - Divided the project into specific tasks (e.g., coding, design, testing).
 - Each team member was responsible for their own tasks and communicated progress regularly.
- **Kanban Process:**
 - **Stages:**
 - **To Do:** Initial interface design, authentication setup.
 - **In Progress:** Game logic, leaderboard functionality.
 - **Completed:** UI design, user registration, and session management.

Code Demo and Walkthrough

- **Gameplay Demo:**
 - This will be a demonstration of the game screen, difficulty selection, and leaderboard update.
- **Highlight Key Features:**
 - User login and registration, gameplay, and scoring system.

Technical Implementation: User Authentication

- **Walkthrough the user registration process:**

- Handle user input (username, password)
- Hash password using 'password_hash()' function
- Store user data in the 'users.txt' file

Code (from register.php):

```
// Hash the password before saving
$password_hash = password_hash($password, PASSWORD_DEFAULT);

// Store username and hashed password in users.txt
$file = 'users.txt';

// Check if the username already exists
$users = file($file, FILE_IGNORE_NEW_LINES);
foreach ($users as $user) {
    list($storedUsername, $storedPassword) = explode(' ', $user);
    if ($storedUsername === $username) {
        $error = "Username already exists.";
        break;
    }
}

// If no existing username, save the new user
if (empty($error)) {
    $userData = $username . ' ' . $password_hash . PHP_EOL;

    // Attempt to write the data to the file
    if (file_put_contents($file, $userData, FILE_APPEND)) {
        $_SESSION['user_name'] = htmlspecialchars($username);
        $_SESSION['logged_in'] = true;

        // Redirect to the main menu after successful registration
        header("Location: menu.php");
        exit();
    } else {
        $error = "Failed to write data to the file.";
    }
}
```

Explanation: The registration process involves checking if the username already exists, hashing the password using `password_hash()`, and storing the user data in a `users.txt` file. If the registration is successful, the user's session is started, and they are redirected to the main menu.

- **Demonstrate the login process:**

- Verify username and password
- Manage user sessions with session_start()
- Implement "remember me" functionality with cookies

Code (from login.php):

```
// Check if either username or password is empty
if (empty($username) || empty($password)) {
    $error = "Username and Password are required."; // Set error message
} else {
    $file = 'users.txt'; // Define the file where user data is stored
    $users = file($file, FILE_IGNORE_NEW_LINES); // Read users from file and ignore new lines

    $validUser = false; // Flag to track if user credentials are valid

    // Loop through the users to find a matching username and password
    foreach ($users as $user) {
        // Split the user data into username and password
        list($storedUsername, $storedPassword) = explode(' ', $user);

        // Check if the username matches and if the password is correct using password_verify
        if ($storedUsername === $username && password_verify($password, $storedPassword)) {
            $validUser = true; // Set validUser to true if a match is found
            break; // Exit the loop once a valid user is found
        }
    }

    // If valid user, create session and optionally remember the user
    if ($validUser) {
        $_SESSION['user_name'] = htmlspecialchars($username); // Store sanitized username in session
        $_SESSION['logged_in'] = true; // Mark user as logged in

        // Check if the "remember me" option was selected
        if (isset($_POST['remember'])) {
            // Set a cookie to remember the user for 30 days
            setcookie('user_name', $_SESSION['user_name'], time() + (86400 * 30), "/");
        }

        header("Location: menu.php"); // Redirect to the menu page after successful login
        exit(); // Exit script to prevent further code execution
    } else {
        $error = "Invalid username or password."; // Set error message if login fails
    }
}
```

Explanation: The login process involves reading the user data from the `users.txt` file, verifying the username and password using `password_verify()`, and starting the user's session. The "Remember me" functionality is implemented using a cookie.

Technical Implementation: Game Logic and Mechanics

Explain the game logic flow:

- Word selection based on difficulty level using the 'getWordForLevel()' function
- Handling user guesses with 'handleGuess()' function and updating game state.
- Calculating score based on remaining lives
- Detecting game over and win conditions

Code (from game.php):

```
// Function to select word based on difficulty
function getWordForLevel($difficulty) {
    $wordBanks = [
        'easy' => [
            'cat', 'dog', 'hat', 'run', 'jump', 'play', 'bird', 'fish', 'ball', 'book',
            'hand', 'lamp', 'desk', 'star', 'moon', 'sun'
        ],
        'medium' => [
            'monkey', 'dragon', 'puzzle', 'guitar', 'planet', 'rainbow', 'picture',
            'elephant', 'dolphin', 'library', 'bicycle', 'journey', 'mystery'
        ],
        'hard' => [
            'challenge', 'adventure', 'discovery', 'knowledge', 'butterfly',
            'universe', 'symphony', 'beautiful', 'wonderful', 'fantastic'
        ]
    ];

    // Filter words based on level requirements
    $validWords = array_filter($wordBanks[$difficulty], function($word) use ($difficulty) {
        $slen = strlen($word);
        switch($difficulty) {
            case 'easy':
                return $slen >= 3 && $slen <= 4;
            case 'medium':
                return $slen >= 5 && $slen <= 7;
            case 'hard':
                return $slen >= 8;
        }
    });

    return $validWords[array_rand($validWords)];
}
```

Explanation: The `getWordForLevel()` function selects a random word from a predefined word bank based on the chosen difficulty level. The function uses `array_filter()` to filter the words based on length requirements for each difficulty.

Code (from game.php):

```
// Function to handle guesses
function handleGuess($letter) {
    $game = &$_SESSION['game_state'];
    $word = $game['word'];
    $correct = false;

    // Make the guess case-insensitive
    $letter = strtolower($letter);

    if (!in_array($letter, $game['guessed'])) {
        $game['guessed'][] = $letter;

        for ($i = 0; $i < strlen($word); $i++) {
            if (strtolower($word[$i]) == $letter) {
                $game['revealed'][$i] = true;
                $correct = true;
            }
        }

        // Deduct lives if the guess is incorrect
        if (!$correct) {
            $game['lives']--;
            if ($game['lives'] <= 0) {
                $game['game_over'] = true;
            }
        }

        // Check if the game is won (all letters are revealed)
        if (!in_array(false, $game['revealed'])) {
            $game['won'] = true;
            $game['score'] += 100 * $game['lives']; // Score based on remaining lives
        }
    }
}
```

Explanation: The `handleGuess()` function processes user guesses, updating the game state accordingly. It checks if the guessed letter is correct, deducts lives for incorrect guesses, and updates the game state to track the revealed letters and the player's score.

Technical Implementation: Leaderboard and Score Tracking

- **Demonstrate the leaderboard functionality:**

- Storing and retrieving the top 10 scores in the session
- Updating the leaderboard when a player wins a game
- Sorting leaderboard by score in descending order

Code (from leaderboard.php):

```
// Check if the game state indicates a win
if ($_SESSION['game_state']['won']) {
    $updated = false; // Flag to check if the user's score is updated

    // Loop through the leaderboard to find the player and update their score
    foreach ($leaderboard as $entry) {
        if ($entry['player'] == $_SESSION['user_name']) {
            $entry['score'] += $_SESSION['game_state']['score']; // Add the score to the existing score
            $updated = true; // Set updated to true since the player's score was found and updated
            break; // Exit the loop once the player's score is updated
        }
    }

    // If the player's score was not found, add a new entry for the player
    if (!$updated) {
        $leaderboard[] = [
            'player' => $_SESSION['user_name'], // Add the player's name
            'score' => $_SESSION['game_state']['score'] // Add the score from the game state
        ];
    }

    // Sort the leaderboard in descending order based on the score
    usort($leaderboard, function($a, $b) {
        return $b['score'] - $a['score']; // Sort by score, highest first
    });

    // Save the top 10 players' scores to the session
    $_SESSION['leaderboard'] = array_slice($leaderboard, 0, 10); // Keep only the top 10 players
}
```

Explanation: When a player wins a game, their score is added to the leaderboard array. The leaderboard is then sorted in descending order by score, and the top 10 scores are stored in the session for display.

Code (from leaderboard.php):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Leaderboard</title>
    <link rel="stylesheet" href="style.css"> <!-- Link to external CSS file -->
</head>
<body>
    <div class="leaderboard-container">
        <h2>Leaderboard</h2>
        <ul>
            <!-- Loop through the leaderboard and display each player's name and score -->
            <?php foreach ($leaderboard as $entry): ?>
                <li><?php echo $entry['player'] . ' : ' . $entry['score']; ?></li>
            <?php endforeach; ?>
        </ul>

        <!-- Link to navigate back to the main menu -->
        <a href="menu.php">Back to Menu</a>
        <br>
        <!-- Logout button that submits a request to logout.php -->
        <form method="post" action="logout.php">
            <button type="submit">Logout</button>
        </form>
    </div>
</body>
</html>
```

Explanation: The leaderboard is displayed using an HTML table, with the top 10 scores fetched from the session and rendered in the table rows.

Technical Implementation: User Interface and CSS Enhancements

- **Highlight the CSS enhancements and animations:**

- Responsive and visually appealing layout
- Animations and transitions for game elements
- Color scheme and typography choices
- Hangman drawing animation

Code Snippet (from style.css):

```
.letter.revealed {
  background: rgba(255, 255, 255, 0.2); /* Darken background when revealed */
  animation: revealLetter 0.6s cubic-bezier(0.4, 0, 0.2, 1);
  box-shadow: 0 0 10px rgba(0, 255, 0, 0.5); /* Glow effect on revealed letters */
}

@keyframes revealLetter {
  0% {
    transform: rotateX(0);
    background: transparent;
  }
  50% {
    background: rgba(255, 255, 255, 0.1);
  }
  100% {
    transform: rotateX(360deg);
    background: rgba(255, 255, 255, 0.2);
  }
}
```

Explanation: The CSS defines a `revealLetter` animation that is applied to the `.letter.revealed` class. This creates a smooth, 3D-like rotation effect when a letter is revealed, along with a subtle glow effect.

Code Snippet (from style.css):

```
/* Hangman Drawing Animation */
.hangman-drawing {
  position: relative;
  height: 200px;
  margin: 2rem auto;
}

.hangman-part {
  position: absolute;
  background: var(--text-color);
  animation: drawPart 0.5s ease-out forwards;
  opacity: 0;
}

@keyframes drawPart {
  from {
    opacity: 0;
    transform: scale(0);
  }
  to {
    opacity: 1;
    transform: scale(1);
  }
}
```

Explanation: The CSS defines a `drawPart` animation that is applied to the hangman drawing elements. This creates a progressive, scaled-up appearance of the hangman parts as the game progresses.

Future Enhancements

- **Potential Additions:**
 - Add categories like animals, sports, and movies.
 - Expand game difficulty levels (e.g., expert, champion).
 - Enable multiplayer mode for head-to-head gameplay.
 - Enhance the visual and audio elements of the game.
 - Transition from users.txt to a database for scalable user data management.

Conclusion

- **Summary:**
 - Implemented a classic Hangman game with modern features
 - Provided a secure and engaging user experience
 - Leveraged PHP, HTML, and CSS to create a visually appealing and functional application
- **Key Takeaways:**
 - *Reinforced PHP skills, session handling, and UI design.*



**Thanks
for Your
Attention!**

Thank you!

