File - register.php

```php
<?php
// Initialising session and connecting to database.
session_start();

$dbhost = "localhost";
$dbuser = "root";
$dbpass = "";
$dbname = "tetris";

$connection = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);


?>

<!DOCTYPE html>
<html>
    <head>
        <title>Signup</title>
        <link rel="stylesheet" href="css/style.css"></link>
    </head>
    <body>
    <div class="navbar">
        <a href="index.php">Home</a>
        <a class="right" href="leaderboard.php">Leaderboard</a>
        <a class="right" href="tetris.php">Play Tetris</a>
    </div>

    <div class="main">
        <div id="box">

            <form method="post" action="index.php">
                <div class="input-title">Signup</div>
                <input class="input" type="text" placeholder="First Name" name="firstName" required><br><br>
                <input class="input" type="text" placeholder="Last name" name="lastName" required><br
```

File - register.php

```
34 ><br>
35                 <input class="input" type="text"
   placeholder="Username" name="username" required><br><
   br>
36                 <input class="input" type="password"
   placeholder="Password" id="password" name="password"
   required><br><br>
37                 <input class="input" type="password"
   placeholder="Confirm Password" id="confirm_password"
   required><br><br>
38                 <p><b>Display Scores on leaderboard</
   b></p>
39                 <input type="radio" id="html" name="
   display" value="1" required>
40                 <label for="display">yes</label>
41                 <input type="radio" id="html" name="
   display" value="0" required>
42                 <label for="display">no</label><br><
   br>
43
44                 <input class="button" type="submit"
   value="Signup"><br><br>
45
46                 <a style="margin-left: 5px;" href="
   index.php">Click to login</a><br><br>
47             </form>
48         </div>
49     </div>
50     <script> // Checks the password and confirm
   password fields are equal.
51         let password = document.getElementById("
   password"),
52             confirm_password = document.
   getElementById("confirm_password");
53
54         function validatePassword(){
55             if(password.value != confirm_password.
   value) {
56                 confirm_password.setCustomValidity("
   Passwords Don't Match");
57             } else {
```

File - register.php

```
58                  confirm_password.setCustomValidity('' 
   );
59              }
60          }
61
62          password.onchange = validatePassword;
63          confirm_password.onkeyup = validatePassword;
64      </script>
65
66      </body>
67  </html>
68
```

File - tetris.php

```php
<?php
// Initialising session and connecting to database.
session_start();

$dbhost = "localhost";
$dbuser = "root";
$dbpass = "";
$dbname = "tetris";

$connection = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);

if(!isset($_SESSION['username'])) {
    // Redirect to index if not logged in.
    header("Location: index.php");
    die;
}

$username = $_SESSION['username'];
// Gets the score and updates the database if applicable.
if($_SERVER['REQUEST_METHOD'] == "POST") {
    // Gets score from invisible form accessed by javascript.
    $score = $_POST['score-from-form'];
    // Gets the score of the user from the database.
    $query = "SELECT * FROM scores WHERE Username = '$username' limit 1";
    $result = mysqli_query($connection, $query);
    if ($result && mysqli_num_rows($result) > 0) {
        $userData = mysqli_fetch_assoc($result);
        if ($userData['Score'] < $score) { // If its a high score update db.
            $query = "UPDATE Scores SET Score = '$score' WHERE Username = '$username' ";
            mysqli_query($connection, $query);
        }
    } else { // If no username found add them and their score.
        $query = "INSERT INTO Scores (Username, Score) VALUES ('$username', '$score')";
```

```
34          mysqli_query($connection, $query);
35      }
36  }
37  ?>
38  <!DOCTYPE html>
39  <html lang="en" dir="ltr">
40      <head>
41          <meta charset="utf-8">
42          <script src="app.js" charset="utf-8"></script
    >
43          <link rel="stylesheet" href="css/style.css"><
    link>
44          <title>ECM1417</title>
45      </head>
46      <body>
47      <div class="navbar">
48          <a href="index.php">Home</a>
49          <a class="right" href="leaderboard.php">
    Leaderboard</a>
50          <a class="right" href="tetris.php">Play
    Tetris</a>
51      </div>
52
53      <div class="main">
54          <audio loop id="tetris_theme" preload="auto">
55              <source url="tetrisTheme.mp3" type="audio
    /mpeg" >
56          </audio>
57          <form id="score-form" name="score-form"
    action="" method="POST">
58              <input id="score-input" name="score-from-
    form" value="">
59          </form>
60          <div id="score-div">Score: <span id="score">0
    </span></div>
61          <button class="button" id="start-button">
    Start the game</button>
62          <div class="tetris-bg", id="tetris-bg"></div>
63      </div>
64
65      </body>
```

```
66 </html>
```

File - leaderboard.php

```php
<?php
// Initialising session and connecting to database.
session_start();

$dbhost = "localhost";
$dbuser = "root";
$dbpass = "";
$dbname = "tetris";

$connection = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);

// Gets the top 10 scores from the database.
//$query = "SELECT Username, Score FROM Scores ORDER BY Score DESC LIMIT 10";
$query = "SELECT Scores.Username, Users.Display, Scores.Score
FROM Scores
INNER JOIN Users ON Scores.Username=Users.Username
ORDER BY Score DESC";
$result = mysqli_query($connection,$query);
?>

<!DOCTYPE html>
<html>
    <head>
        <title>Leaderboard</title>
        <link rel="stylesheet" type="text/css" href="css/style.css"/>
    </head>
    <body>
    <div class="navbar">
        <a href="index.php">Home</a>
        <a class="right" href="leaderboard.php">Leaderboard</a>
        <a class="right" href="tetris.php">Play Tetris</a>
    </div>

    <div class="main">
        <div class="leaderboard">
```

```
36              <table>
37                  <tr>
38                      <th>Username</th>
39                      <th>Score</th>
40                  </tr>
41                  <?php // Iterates through each row in
    the database.
42                      while($row = mysqli_fetch_array(
    $result)){
43                          if ($row['Display'] == 1) {
44                              // Creates a new row in
    the html table with the username and score.
45                              echo "<tr><td>" . $row['
    Username'] . "</td><td>" . $row['Score'] . "</td></tr
    >";
46                          }
47                      }
48                  ?>
49              </table>
50          </div>
51      </div>
52      </body>
53  </html>
54
```

File - index.php

```php
<?php
// Initialising session and connecting to database.
session_start();

$dbhost = "localhost";
$dbuser = "root";
$dbpass = "";
$dbname = "tetris";

$connection = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);

// Gets registration details from form because specification states to post to index.php.
if($_SERVER['REQUEST_METHOD'] == "POST" && isset($_POST['firstName'])) {
    $username = $_POST['username'];
    $firstName = $_POST['firstName'];
    $lastName = $_POST['lastName'];
    $password = $_POST['password'];
    $display = $_POST['display'];

    // Checks if name is unique.
    $select = mysqli_query($connection, "SELECT * FROM users WHERE username = '".$username."'");
    if(mysqli_num_rows($select)) {
        // Redirect to register.php if username not unique.
        header("Location: register.php");
        die;
    }

    // Checks all fields of the form were inputted.
    if(!empty($username) && !empty($firstName) && !empty($lastName) && !empty($password)) {
        $query = "INSERT INTO Users VALUES ('$username', '$firstName', '$lastName', '$password', '$display');";
        mysqli_query($connection, $query);
    }else {
        // Redirect to register.php if bad input.
```

File - index.php

```
34          header("Location: register.php");
35          die;
36      }
37  }
38
39  // If the user is not logged in we need to get their
    login details from the html form.
40  if(!isset($_SESSION['username'])) {
41      if($_SERVER['REQUEST_METHOD'] == "POST") {
42          $username = $_POST['username'];
43          $password = $_POST['password'];
44
45          // Check login details against the database.
46          if(!empty($username) && !empty($password)) {
47              $query = "SELECT * FROM Users WHERE
    Username = '$username' limit 1";
48              $result = mysqli_query($connection,
    $query);
49              // If a result is received compare the
    passwords.
50              if ($result && mysqli_num_rows($result
    ) > 0) {
51                  $userData = mysqli_fetch_assoc(
    $result);
52                  if($userData['Password'] ===
    $password){
53                      $_SESSION['username'] = $userData
    ['UserName'];
54                  }
55              }
56          }else {
57              echo "wrong username or password!";
58          }
59      }
60  }
61  ?>
62
63  <!DOCTYPE html>
64  <html>
65      <head>
66          <title>Welcome Page</title>
```

File - index.php

```
67          <link rel="stylesheet" href="css/style.css"
    ><link>
68      </head>
69      <body>
70      <div class="navbar">
71          <a href="index.php">Home</a>
72          <a class="right" href="leaderboard.php">
    Leaderboard</a>
73          <a class="right" href="tetris.php">Play
    Tetris</a>
74      </div>
75
76      <div class="main">
77          <div id="box">
78              <?php if(isset($_SESSION['username'])):
    ?>
79                  <h1 id="welcome-message">Welcome to
    Tetris</h1>
80                  <a href="tetris.php" class="button"
    id="welcome-button">Click here to play</a>
81              <?php else : ?>
82                  <form method="post">
83                      <div class="input-title">Login</
    div>
84                      <input class="input" placeholder
    ="username" type="text" name="username"><br><br>
85                      <input class="input" placeholder
    ="password" type="password" name="password"><br><br>
86
87                      <input class="button" type="
    submit" value="Login" name="login_button"><br><br>
88
89                      <p style="margin-left: 5px;">Don
    't have a user account? <a href="register.php">
    Register now</a>
90                      </p><br><br>
91
92                  </form>
93              <?php endif; ?>
94          </div>
95      </div>
```

File - index.php

```
 96
 97     </body>
 98 </html>
 99
100
```

File - app.js

```
// The Pieces, each row represents a different rotation.
const lTetromino = [
    [[1,1],[1,2],[1,3],[2,3]],
    [[1,2],[0,2],[-1,2],[-1,3]],
    [[0,2],[0,1],[0,0],[-1,0]],
    [[-1,1],[0,1],[1,1],[1,0]]
]

const zTetromino = [
    [[1,1],[2,1],[2,2],[3,2]],
    [[3,2],[3,3],[2,3],[2,4]],
    [[2,4],[1,4],[1,3],[0,3]],
    [[1,1],[1,2],[0,2],[0,3]]
]

const sTetromino = [
    [[1,2],[2,1],[2,2],[3,1]],
    [[2,2],[3,3],[2,3],[3,4]],
    [[2,3],[1,4],[1,3],[0,4]],
    [[1,3],[0,2],[1,2],[0,1]]
]

const tTetromino = [
    [[1,1],[2,1],[2,2],[3,1]],
    [[1,0],[1,1],[0,1],[1,2]],
    [[0,0],[1,0],[1,-1],[2,0]],
    [[2,-1],[2,0],[3,0],[2,1]]
]

const oRotation = [[1,1],[1,2],[2,1],[2,2]]

const oTetromino =  [
    oRotation,oRotation,oRotation,oRotation
]

const iTetromino = [
    [[1,1],[1,2],[1,3],[1,4]],
    [[-1,3],[0,3],[1,3],[2,3]],
    [[0,1],[0,2],[0,3],[0,4]],
    [[-1,1],[0,1],[1,1],[2,1]]
```

File - app.js

```
41 ]
42
43 const theTetrominoes = [lTetromino, zTetromino,
   sTetromino, tTetromino, oTetromino, iTetromino]
44 const typeOfTetrominoes = ['l', 'z', 's', 't', 'o', '
   i']
45 const colours = ['orange', 'red', 'green', 'purple',
   'yellow', 'blue']
46 const fallTime = 1000 // Default 1000 = 1 second.
47 const tileSize = 20 // Laptop preferred 20px, desktop
    preferred 40px, must change css accordingly.
48 let grid = createGrid()
49 let currentBlock
50 let scoreDisplay
51 let tetrominoID
52 let currentColumnIndex
53 let currentRowIndex
54 let rotation = 0
55 let timerId
56 let startBtn = null
57 let gameMusic = null
58 let score = 0
59
60 /**
61  * Plays the tetris theme on loop.
62  */
63 function playMusic() {
64     gameMusic = new Audio('audio/tetrisTheme.mp3');
65     gameMusic.loop = true;
66     gameMusic.volume = 0.5
67     gameMusic.load();
68     gameMusic.play()
69 }
70
71 /**
72  * Pauses the tetris theme.
73  */
74 function pauseMusic() {
75     gameMusic.pause()
76 }
77
```

File - app.js

```
 78 /**
 79  * Plays audio specified by filename.
 80  *
 81  * @param soundFileName
 82  */
 83 function playSound(soundFileName) {
 84     if(typeof soundFileName === "string") {
 85         let gameSound = new Audio('audio/'+
    soundFileName+'.mp3')
 86         gameSound.play()
 87     }
 88 }
 89
 90 /**
 91  * @returns A 20x10 2D array populated with empty
    strings.
 92  */
 93 function createGrid() {
 94     console.log("grid")
 95     let gridSquares = []
 96     for (let i = 0; i < 20; i++) {
 97         gridSquares[i] = []
 98         for (let j = 0; j < 10; j++) {
 99             gridSquares[i][j] = ''
100         }
101     }
102     return gridSquares
103 }
104
105 /**
106  * Randomly spawns a new tetromino after checking
    for game over conditions.
107  */
108 function newTetromino() {
109     console.log("newTetromino")
110     tetrominoID = Math.floor(Math.random() *
    theTetrominoes.length)
111     currentRowIndex = 0
112     currentColumnIndex = 4
113     rotation = 0
114     let currentPiece = theTetrominoes[tetrominoID]
```

File - app.js

```
115     let tilesOfPiece = []
116     for (let i = 0; i < currentPiece.length; i++) {
117         tilesOfPiece[i] = document.createElement('
    div')
118         tilesOfPiece[i].setAttribute("class","block"
    )
119         tilesOfPiece[i].setAttribute("id",
    typeOfTetrominoes[tetrominoID])
120         document.getElementById("tetris-bg").
    appendChild(tilesOfPiece[i])
121     }
122     currentBlock = tilesOfPiece
123 
124     // Checks top of grid is empty
125     let associativeArray = theTetrominoes[
    tetrominoID][rotation]
126     if (associativeArray.some(index => grid[
    currentRowIndex + index[1] - 1][currentColumnIndex
     + index[0] - 1] !== '')) {
127         gameOver()
128     } else {
129         // Update score
130         addScore()
131 
132         // Draws tile onto screen
133         currentBlock.forEach(tile => {
134             tile.style.backgroundColor = colours[
    tetrominoID]
135             tile.style.backgroundImage = 'url(images
    /' + colours[tetrominoID] + 'Tile.png)'
136         })
137         translate()
138     }
139 }
140 
141 /**
142  * Checks every row in the grid to see if it is full
    , then removes said rows and shifts the rest of the
    tiles down
143  * whilst adding new empty rows to the top of the
    grid.
```

File - app.js

```
144  */
145 function completeRow() {
146     console.log("complete row")
147     let emptyRow = ['', '', '', '', '', '', '', '',
    '', '']
148     let tileToBeRemoved
149     let tileToBeShifted
150     let removeCount = 0
151     for (let rowIndex = 0; rowIndex < grid.length;
    rowIndex++) {
152         let complete = 0
153         // Checks each column in the row to see if
    it is full.
154         for (let columnIndex = 0; columnIndex < grid
    [rowIndex].length; columnIndex++) {
155             if (grid[rowIndex][columnIndex] !== ''
    ) {
156                 complete++
157             } else {
158                 break;
159             }
160         }
161         // Remove a row.
162         if (complete === 10) {
163             removeCount++
164             complete = 0
165             // Deletes the div blocks.
166             for (let i = 0; i < grid[rowIndex].
    length; i++) {
167                 tileToBeRemoved = grid[rowIndex][i]
168                 tileToBeRemoved.remove()
169             }
170             // Shifts down all the pieces above the
    removed row.
171             for (let shiftDownRowIndex = rowIndex;
    shiftDownRowIndex >= 0; shiftDownRowIndex--) {
172                 for (let i = 0; i < grid[
    shiftDownRowIndex].length; i++) {
173                     tileToBeShifted = grid[
    shiftDownRowIndex][i]
174                     if (tileToBeShifted != '') {
```

```
175                         let yAxis = parseInt(
    tileToBeShifted.style.top.slice(0,-2)) + tileSize
176                         tileToBeShifted.style.top =
    yAxis + 'px'
177                     }
178                 }
179             }
180             // Deletes the row from the grid and
    adds a new empty row.
181             grid.splice(rowIndex,1)
182             grid.unshift(emptyRow)
183             console.log(grid)
184 
185         }
186     }
187     // Plays a special sound if 4 or more rows are
    removed at once.
188     if (removeCount >= 4) {
189         playSound("line-removal4")
190     }else if (removeCount > 1) {
191         playSound("line-remove")
192     }
193 }
194 
195 /**
196  * Moves the blocks representing the tetromino.
197  */
198 function translate() {
199     console.log("translate")
200     for (let i = 0; i < currentBlock.length; i++) {
201         let tile = currentBlock[i]
202         let pieceType = theTetrominoes[tetrominoID][
    rotation]
203         let relativeColumnPosition = pieceType[i][0
    ] - 1
204         let relativeRowPosition = pieceType[i][1] -
    1
205         let xAxis = tileSize * (currentColumnIndex
     + relativeColumnPosition)
206         let yAxis = tileSize * (currentRowIndex +
    relativeRowPosition)
```

File - app.js

```
207
208         tile.style.left = xAxis + 'px'
209         tile.style.top = yAxis + 'px'
210     }
211     currentBlock.forEach(tile => {
212         console.log(tile)
213     })
214 }
215
216 // Assign functions to keyCodes.
217 function keyboardInterrupt(keyInput) {
218     console.log("keyboard interrupt")
219     if (keyInput.keyCode === 32) {
220         hardDrop()
221     } else if (keyInput.keyCode === 37 || keyInput.
    keyCode === 65) {
222         moveLeft()
223     } else if (keyInput.keyCode === 38 || keyInput.
    keyCode === 87) {
224         rotate()
225     } else if (keyInput.keyCode === 39 || keyInput.
    keyCode === 68) {
226         moveRight()
227     } else if (keyInput.keyCode === 40 || keyInput.
    keyCode === 83) {
228         moveDown()
229     }
230 }
231
232 /**
233  * @returns Boolean of if the tetromino would
    collide if it were to go down by one tile.
234  */
235 function downCollisionDetection() {
236     console.log("down collision detection")
237     // Checks if tile below is floor or occupied.
238     let associativeArray = theTetrominoes[
    tetrominoID][rotation]
239     try {
240         return (associativeArray.some(index => (
    currentRowIndex + index[1] === 20)) ||
```

```
240 associativeArray.some(index => grid[currentRowIndex
     + index[1]][currentColumnIndex + index[0] - 1] !==
    ''))
241     } catch (error) {
242         // Out of bounds but javascript returns
    undefined, so we catch a type error instead.
243         if (error instanceof TypeError) {
244             return true
245         }else {
246             throw error
247         }
248     }
249 }
250 
251 /**
252  * @returns Boolean of if the tetromino would
    collide if it were to go left by one tile.
253  */
254 function leftCollisionDetection() {
255     console.log("left collision detection")
256     // Checks if tile below is left wall or occupied
    .
257     let associativeArray = theTetrominoes[
    tetrominoID][rotation]
258     try {
259         return (associativeArray.some(index =>
    currentColumnIndex + index[0] - 2 < 0) ||
    associativeArray.some(index => grid[currentRowIndex
     + index[1] - 1][currentColumnIndex + index[0] - 2
    ] !== ''))
260     } catch (error) {
261         // Out of bounds but javascript returns
    undefined, so we catch a type error instead.
262         if (error instanceof TypeError) {
263             return true
264         }else {
265             throw error
266         }
267     }
268 }
269 
```

File - app.js

```
/**
 * @returns Boolean of if the tetromino would collide if it were to go right by one tile.
 */
function rightCollisionDetection() {
    console.log("right collision detection")
    // Checks if tile below is right wall or occupied.
    let associativeArray = theTetrominoes[tetrominoID][rotation]
    try {
        return (associativeArray.some(index => currentColumnIndex + index[0] > 9) || associativeArray.some(index => grid[currentRowIndex + index[1] - 1][currentColumnIndex + index[0]] !== ''))
    } catch (error) {
        // Out of bounds but javascript returns undefined, so we catch a type error instead.
        if (error instanceof TypeError) {
            return true
        }else {
            throw error
        }
    }
}

/**
 * @returns Boolean of if the tetromino is colliding with a border or block.
 */
function selfCollisionDetection() {
    console.log("self collision detection")
    let associativeArray = theTetrominoes[tetrominoID][rotation]
    // Checks if the tetromino is colliding.
    try {
        return (associativeArray.some(index => grid[currentRowIndex + index[1] - 1][currentColumnIndex + index[0] - 1] !== ''))
    } catch (error) {
```

File - app.js

```
299         // Out of bounds but javascript returns
    undefined, so we catch a type error instead.
300         if (error instanceof TypeError) {
301             return true
302         }else {
303             throw error
304         }
305     }
306 }
307 
308 /**
309  * Moves the tetromino left by one tile.
310  */
311 function moveLeft() {
312     console.log("left")
313     if (!leftCollisionDetection()) {
314         currentColumnIndex--
315         translate()
316     }
317 }
318 
319 /**
320  * Moves the tetromino right by one tile.
321  */
322 function moveRight() {
323     console.log("right")
324     if (!rightCollisionDetection()) {
325         currentColumnIndex++
326         translate()
327     }
328 }
329 
330 /**
331  * Moves the tetromino down by one tile.
332  */
333 function moveDown() {
334     console.log("down")
335     if (!downCollisionDetection()) {
336         currentRowIndex++
337         playSound("whoosh")
338         translate()
```

```
339     }
340 }
341 
342 /**
343  * Rotates the tetromino 90 degrees clockwise.
344  */
345 function rotate() {
346     console.log("rotate")
347     rotation++
348     rotation %= 4
349     if (selfCollisionDetection()) {
350         rotation--
351         rotation %= 4
352     }else {
353         playSound("block-rotate")
354     }
355     translate()
356 }
357 
358 /**
359  * Moves the tetromino down until it collides.
360  */
361 function hardDrop() {
362     console.log("hard drop")
363     while (!downCollisionDetection()) {
364         currentRowIndex++
365     }
366     playSound("line-drop")
367     translate()
368     freeze()
369 }
370 
371 /**
372  * Stops the tetromino in place and calls for a
    complete row check and a new tetromino.
373  */
374 function freeze() {
375     console.log(grid)
376     if (downCollisionDetection()) {
377         let associativeArray = theTetrominoes[
    tetrominoID][rotation]
```

File - app.js

```
378         for (let i = 0; i < associativeArray.length
    ; i++) {
379             let index = associativeArray[i]
380             let tile = currentBlock[i]
381             try {
382                 grid[currentRowIndex + index[1] - 1
    ][currentColumnIndex + index[0] - 1] = tile
383             } catch (error) {
384                 // Out of bounds but javascript
    returns undefined, so we catch a type error instead.
385                 if (!(error instanceof TypeError)) {
386                     throw error
387                 }
388             }
389 
390         }
391         completeRow()
392         newTetromino()
393     } else {
394         moveDown()
395     }
396 
397 }
398 
399 /**
400  * Adds one to the users score and displays it on
    the webpage.
401  */
402 function addScore() {
403     score++
404     scoreDisplay.innerHTML = score
405 }
406 
407 /**
408  * Resets the game and posts the users score to the
    server.
409  */
410 function gameOver() {
411     console.log("Game Over")
412     startBtn.style.visibility = "visible";
413     pauseMusic()
```

File - app.js

```
414     playSound('gameOver')
415     document.removeEventListener('keyup',
    keyboardInterrupt)
416     clearInterval(timerId)
417     grid = createGrid()
418     let scoreForm = document.getElementById('score-
    form')
419     let scoreInput = document.getElementById('score-
    input')
420     scoreInput.setAttribute('value',score)
421     scoreForm.submit()
422 }
423
424 // Start of the code.
425 document.addEventListener('DOMContentLoaded'
    , () => {
426     // Page Setup
427     startBtn = document.querySelector('#start-button
    ')
428     scoreDisplay = document.querySelector('#score')
429
430     startBtn.addEventListener('click', () => {
431         console.log("startbtn")
432         document.getElementById("tetris-bg").
    innerHTML = ''
433         startBtn.style.visibility = "hidden";
434         playMusic()
435         score = 0
436         document.addEventListener('keyup',
    keyboardInterrupt)
437         timerId = setInterval(freeze, fallTime)
438
439         newTetromino()
440     })
441 })
442
443
```

File - style.css

```css
html, body {
    margin: 0;
    height: 100%;
}

table, th, td {
    border-spacing: 2px;
    border: 2px solid black;
    font-size: 25px;
}

th {
    background-color: blue;
}

#start-button {
    position: absolute;
    left: 30%;
    top: 30%;
    background-color: blue;
}

#box{

    background-color: #c7c7c7;
    box-shadow: 5px 5px;
    position: center;
    width: 400px;
    padding: 20px;
    margin: auto;
    margin-top: 20px;
}

#welcome-button{
    margin-left: 35%;
}

#welcome-message{
    text-align: center;
}

```

File - style.css

```
#score-div {
    color: white;
    background-color: blue;
    margin: auto;
    width: 130px;
    height: 50px;
    font-size: xx-large;
    border: black solid 3px;
}

#score-input, #score-form {
    visibility: hidden;
}

.main{
    background-image: url(../images/tetris.png);
    background-repeat: no-repeat;
    background-size: cover;
    margin: auto;
    width: 95%;
    height: 93%;
}

.input-title{
    font-size: 25px;
    margin: 10px;
    padding-top: 10px;
    text-align: center;
    color: Black;
    font-weight: bold;
}

.button {
    background-color: #0096FF;
    border: none;
    color: white;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
```

```
 83      cursor: pointer;
 84  }
 85
 86  .input{
 87
 88      height: 25px;
 89      border-radius: 5px;
 90      padding: 4px;
 91      border: solid thin #aaa;
 92      width: 92%;
 93      margin-left: 5px;
 94  }
 95
 96  .container {
 97      background-color: #c7c7c7;
 98  }
 99
100  .tetris-bg {
101      width: 200px;
102      height: 400px;
103      background-image: url(../images/tetris-grid-bg.
     png);
104      background-repeat: no-repeat;
105      background-size: cover;
106      background-color: #c7c7c7;
107      position: absolute;
108      box-shadow: 5px 5px;
109      top: 40%;
110      left: 50%;
111      margin: -100px 0 0 -100px;
112  }
113
114  .tetris-bg div {
115      height: 20px;
116      width: 20px;
117      background-repeat: no-repeat;
118      background-size: cover;
119  }
120
121  .block {
122      position: absolute;
```

```
    top: 0px;
    right: 0px;
    bottom: 0px;
    left: 0px;
}

.leaderboard {
    background-color: #c7c7c7;
    display: table;
    box-shadow: 5px 5px;
    margin: auto;
}

.navbar {
    background-color: blue;
    overflow: hidden;
    margin: 0px;
    height: 7%;
}

.navbar a {
    float: left;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    font-family: Arial;
    font-size: 12px;
    font-weight: bold;
    text-decoration: none;
}

.navbar a:hover {
    background-color: #c7c7c7;
    color: black;
}

.navbar a.right{
    float: right;
}
```