

Assignment 3

This file document the Wordle game updtes with php and Ajax.

Overview

We used php to build a top 5 best score board, which will use `$_SESSION` array to store the best 5 attempts to solve the Wordle game. (We choose 5 since Wordle is different from the game Yatzy). And a global variable `attempts` to keep track the total number of attempts of user, when a user win the wordle, the attempts will increase by one, and decrease by 1 if the player loses all the chances.

Ajax

We use the axis to call perform the Ajax functions, the details are in the comment of the code.

```
import axios from 'axios'

// Create a object to send the ajax request to my php server
const apiClient = axios.create({
  baseURL: 'http://localhost:8080',
  // Include the cookies,
  withCredentials: true,
  // Format is json
  headers: {
```

```
baseURL: 'http://localhost:8080',
// Include the cookies,
withCredentials: true,
// Format is json
headers: {
  Accept: 'application/json',
  'Content-Type': 'application/json'
}
})

export default {
  // Get the $_SESSION array from the player.php, to get all the user attempts
  getMessage() {
    return apiClient.get('/player.php')
  },
  // Put new attempts of a player into the $_SESSION array
  sendMessage(player) {
    return apiClient.post('/player.php', player)
  },
  // Close and destroy the $_SESSION array, to refresh the score board
  destroySession() {
    return apiClient.get('/quit.php')
  },
  // To update the global variable attempts into the $_SESSION array
  postAttempts(attempts) {
    apiClient.post('/attempts.php', JSON.stringify({ attempts }))
  },
  // Get the global variable attempts from $_SESSION array
  getAttempts() {
    return apiClient.get('/attempts.php')
  }
}
```

Global Variable: Attempts

This variable is aim to track the total number of attempts the player since the game starts. Since it's stored in the `$_SESSION` array, when u refresh the page, it will not be reset to 0.

We update the variable when the user either win or lose a game:

```
const resetBoard = () => {
  ...
  // Pass the variable to the attempts.php
  api.postAttempts(php_attempts.value++)
  ...
}
```

attempts.php

```
if ($_SERVER["REQUEST_METHOD"] === "POST") {
  $json = file_get_contents("php://input");
  // Get the updated variable's value
  $data = json_decode($json, true);

  // Initialize Attempt_Number if not already set
  if (!isset($_SESSION["Attempt_Number"])) {
    $_SESSION["Attempt_Number"] = 1;
  }

  if (json_last_error() !== JSON_ERROR_NONE) {
    http_response_code(400);
    die(json_encode(["error" => "Invalid JSON: " . json_last_error_msg()]));
  }
}
```

```
if (isset($data["attempts"])) {
    // Update the variable
    $_SESSION["Attempt_Number"] += $data["attempts"];
}

} elseif ($_SERVER["REQUEST_METHOD"] === "GET") {
    echo json_encode($_SESSION);
}
```

Scoreboard

In the `player.php` we store a object:

```
let player = {
    a_number: php_attempts.value,
    a_attempts: 7 - life.value
}
```

into the session. In the method `handleKeyDown`, we pass this object to the `player.php`:

```
const handleKeyDown = (event: KeyboardEvent) => {
    ...
    if (inputWord.value[currentRow.value].join('') === answer.value.toUpperCase()) {
        ...
        // Construct a player object
        let player = {
            a_number: php_attempts.value,
            a_attempts: 7 - life.value
        }
        // Send the object
        api.sendMessage(player)
```

In the player.php:

```
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $json = file_get_contents("php://input");
    // Get the player object
    $data = json_decode($json, true);

    if (json_last_error() !== JSON_ERROR_NONE) {
        http_response_code(400);
        die(json_encode(["error" => "Invalid JSON: " . json_last_error_msg()]));
    }

    if (isset($data['a_number']) && isset($data['a_attempts'])) {
        // Store the number of attempts and the sucess attempts into new variables
        $attempt_number = $data['a_number'];
        $attempts = $data['a_attempts'];
        // Put into the $_SESSION array
        $_SESSION["Attempt_{$attempt_number}"] = $attempts;
        // Return this array (optional)
        $response = ["Attempt " . $attempt_number => $_SESSION["Attempt_{$attempt_number}"]];
        echo json_encode($response);
    }
} elseif ($_SERVER["REQUEST_METHOD"] === "GET") {
    // Get the $_SESSION array
    echo json_encode($_SESSION);
}
```

Lastly, we will retrieve the both the global variable and the player attempts from the session:

Lastly, we will retrieve the both the global variable and the player attempts from the session:

```
async function fetchMessage() {
  try {
    const session = await api.getMessage()
    // Get session's data
    const session_object = session.data
    const tbody = document.getElementsByTagName('tbody')[0]

    // Filter out 'Attempt_Number' and then sort the attempts in ascending order
    const session_array = Object.entries(session_object)
      .filter((player) => player[0] !== 'Attempt_Number')
      .sort((a, b) => a[1] - b[1])

    // Only keep the top 5 attempts
    const top_5 = session_array.slice(0, 5)

    // Put each attempts into the table, except the global variable
    top_5.forEach((player) => {
      // Ignore
      if (player[0] === 'Attempt_Number') {
        return
      }
      // Create the table elements
      const tr = document.createElement('tr')
      const td_key = document.createElement('td')
      const td_value = document.createElement('td')

      td_key.innerText = player[0].substring(8)
      td_value.innerText = player[1]
    })
  }
}
```

```
        tr.appendChild(td_key)
        tr.appendChild(td_value)

    tbody.appendChild(tr)
  })
} catch (error) {
  console.error('Error fetching message:', error)
}
}
```

Clean Scoreboard

We also include a button to clean the session, so the player can refresh their score board!

```
<?php
// Start
session_start();
...

if ($_SERVER["REQUEST_METHOD"] === "GET") {
  // Unset first
  session_unset();
  // Destroy the session
  session_destroy();
}
echo json_encode(["status" => "All session variables destroyed"]);
?>
```

Button

```
function destroy() {
```

Button

```
function destroy() {  
  // Use the api object to call the method  
  api.destroySession()  
}
```

Api

```
import axios from 'axios'  
  
const apiClient = axios.create({  
  baseURL: 'http://localhost:8080',  
  withCredentials: true,  
  headers: {  
    Accept: 'application/json',  
    'Content-Type': 'application/json'  
  }  
})  
  
export default {  
  ...  
  // Method to destroy the session  
  destroySession() {  
    return apiClient.get('/quit.php')  
  }  
  ...  
}
```

State of the game

1 Initial State

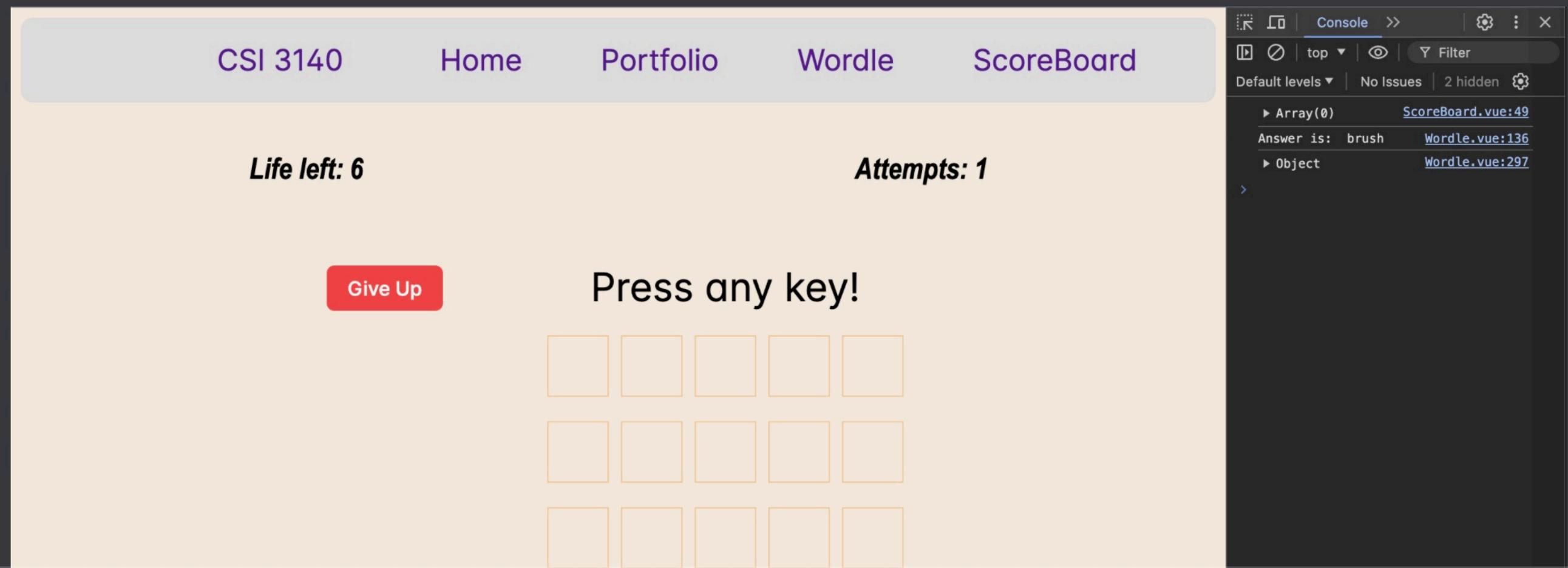
Run the following command to start the game

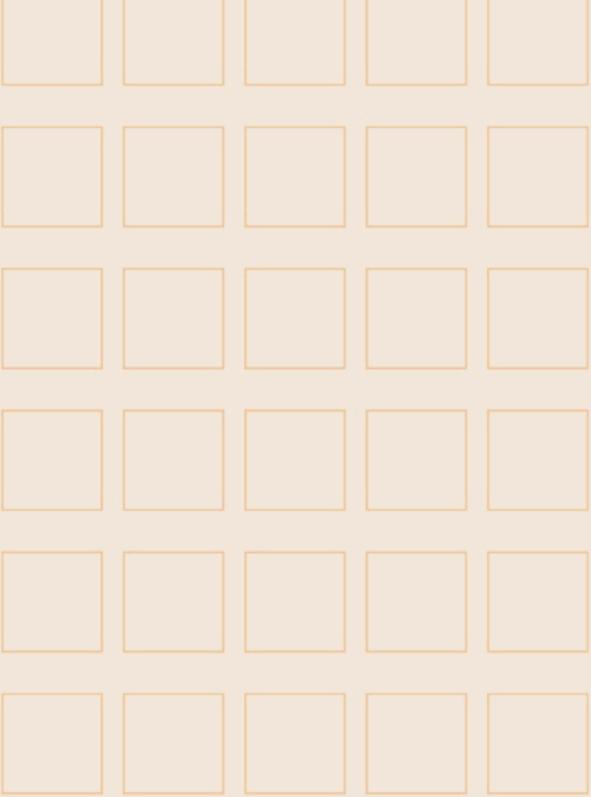
```
Wordle git:(main) ✘ npm run dev
```

Run this to start the php server

```
→ Wordle git:(main) ✘ php -S localhost:8080 -t src/api
[Wed Jul 10 15:49:04 2024] PHP 8.3.8 Development Server (http://localhost:8080) started
□
```

The game should look like





CSI 3140

Home

Portfolio

Wordle

ScoreBoard

Top 3 Scoreboard

Table

Attempt Number	Attempts

Clean Board

Console >> |   top |  Filter |  Default levels | No Issues | 2 hidden | 

```
▶ Array(0) ScoreBoard.vue:49
Answer is: brush Wordle.vue:136
▶ Object Wordle.vue:297
▶ [] ScoreBoard.vue:49
>
```

2 Playing

Now start our first trial

You can see on the terminal, the correct answer is: facet, and we have used 3 life, if we enter the correct answer in the 4th line, then our attempt number is 1(1st round), and the attempts is 4.

The screenshot shows a web application interface for a Wordle game. At the top, there is a navigation bar with links for 'CSI 3140', 'Home', 'Portfolio', 'Wordle', and 'ScoreBoard'. The 'Wordle' link is highlighted. On the left, there is a message 'Life left: 3' and a red 'Give Up' button. In the center, there is a message 'Press any key!' above a 4x5 grid of letters. The letters are colored as follows: Row 1: D (grey), J (grey), A (yellow), J (grey), W (grey). Row 2: D (grey), E (yellow), A (yellow), T (yellow), H (grey). Row 3: F (cyan), A (cyan), T (yellow), T (yellow), Y (grey). Row 4: Five empty orange-outlined boxes. To the right of the grid is a developer tools 'Console' tab showing the following log:

```
Array(0) ScoreBoard.vue:49
Answer is: brush Wordle.vue:136
▶ Object Wordle.vue:297
▶ [] ScoreBoard.vue:49
Answer is: facet Wordle.vue:136
Wordle.vue:297
{data: Array(0), status: 200, statu
▶ sText: 'OK', headers: AxiosHeaders,
config: {...}, ...}
```

Lets check out scoreboard, you can see the table has been updated, the Attempt number indicates n-th round of player's result, and the right column is the number of attempts player used.

Top 3 Scoreboard

Table

Attempt Number	Attempts
1	4

Clean Board

```
Console >> | Settings | X
Default levels | No Issues | 2 hidden | Filter
▶ Array(0) ScoreBoard.vue:49
Answer is: brush Wordle.vue:136
▶ Object Wordle.vue:297
▶ [] ScoreBoard.vue:49
Answer is: facet Wordle.vue:136
Wordle.vue:297
{data: Array(0), status: 200, statu
▶ sText: 'OK', headers: AxiosHeaders,
config: {...}, ...}
Answer is: eagle Wordle.vue:136
ScoreBoard.vue:49
▼ {Attempt_1: 4, Attempt_Number: 2}
  i
    Attempt_1: 4
    Attempt_Number: 2
  ▶ [[Prototype]]: Object
>
```


Now on the scoreboard, the table will display the 2nd try first since it has a lower(better) attempts.

CSI 3140 Home Portfolio Wordle ScoreBoard

Top 3 Scoreboard

Table

Attempt Number	Attempts
2	1
1	4

Clean Board

Console >>

```
Array(0) ScoreBoard.vue:49
Answer is: brush Wordle.vue:136
▶ Object Wordle.vue:297
▶ [] ScoreBoard.vue:49
Answer is: facet Wordle.vue:136
Wordle.vue:297
{data: Array(0), status: 200, statu
▶ sText: 'OK', headers: AxiosHeaders,
config: {...}, ...}
Answer is: eagle Wordle.vue:136
ScoreBoard.vue:49
▼ {Attempt_1: 4, Attempt_Number: 2}
  ▶ Attempt_1: 4
  ▶ Attempt_Number: 2
  ▶ [[Prototype]]: Object
Answer is: floor Wordle.vue:136
ScoreBoard.vue:49
▼ {Attempt_1: 4, Attempt_Number: 2, A
  ▶ ttempt_2: 1} ▶ Attempt_1: 4
  ▶ Attempt_2: 1
  ▶ Attempt_Number: 2
  ▶ [[Prototype]]: Object
>
```

The table should write as: Top 5 Scoreboard. I didn't realize it by far...

3 Keep playing

Lets keep try a few time, and see the results.

The screenshot shows a web application interface. At the top, there is a navigation bar with links: 'CSI 3140', 'Home', 'Portfolio', 'Wordle', and 'ScoreBoard'. The 'ScoreBoard' link is highlighted with a red box. Below the navigation bar, the text 'Top 5 Scoreboard' is displayed in red. Underneath this, the word 'Table' is centered. A table is shown with two columns: 'Attempt Number' and 'Attempts'. The data in the table is:

Attempt Number	Attempts
2	1
11	1
3	2
1	4
9	5

Below the table is a red button labeled 'Clean Board'.

On the right side of the screen, the browser's developer tools are open, specifically the 'Console' tab. The console output shows a list of words and their attempt counts, which correspond to the data in the table. The words listed are: exact, aware, dandy, delay, debit, class, feast, climb, brush, and dirty. Each word is followed by its attempt count and the line number where it was found (e.g., 'exact Wordle.vue:135').

As the table shows, no matter how many attempts we try, it will only store the top 5 scores!

4 Clean Board

If you want to challenge yourself again, you can press the `clean board` button

The screenshot shows a web application interface with a terminal window on the right. The application has a header with tabs: 'CSI 3140', 'Info', 'Portfolio', 'File', and 'ScoreBoard'. A green toast notification in the 'ScoreBoard' tab area says 'Score Board has been reset'. The main content is a red 'Top 5 Scoreboard' section with a table and a 'Clean Board' button. The table has two columns: 'Attempt Number' and 'Attempts'. The data is as follows:

Attempt Number	Attempts
2	1
11	1
3	2
1	4
9	5

Below the table is a red 'Clean Board' button. To the right, a terminal window shows the following log entries:

```
ScoreBoard.vue
{Attempt_1: 4, Attempt_Number:
▶ Attempt_2: 1, Attempt_3: 2, Attempt_6: 6}

Answer is: debit Wordle.vue:
ScoreBoard.vue
{Attempt_1: 4, Attempt_Number:
▶ Attempt_2: 1, Attempt_3: 2, Attempt_6: 6}

Answer is: class Wordle.vue:
Answer is: feast Wordle.vue:
Answer is: climb Wordle.vue:
ScoreBoard.vue
{Attempt_1: 4, Attempt_Number:
▶ Attempt_2: 1, Attempt_3: 2, Attempt_6: 6, ...}

Answer is: brush Wordle.vue:
ScoreBoard.vue
{Attempt_1: 4, Attempt_Number:
▶ Attempt_2: 1, Attempt_3: 2, Attempt_6: 6, ...}

Answer is: dirty Wordle.vue:
Answer is: brush Wordle.vue:
ScoreBoard.vue
{Attempt_1: 4, Attempt_Number:
▶ Attempt_2: 1, Attempt_3: 2, Attempt_6: 6, ...}

Answer is: delay Wordle.vue:
ScoreBoard.vue
{Attempt_1: 4, Attempt_Number:
▶ Attempt_2: 1, Attempt_3: 2, Attempt_6: 6, ...}

Answer is: exact Wordle.vue:
ScoreBoard.vue
{Attempt_1: 4, Attempt_Number:
▶ Attempt_2: 1, Attempt_3: 2, Attempt_6: 6, ...}
```

After a few seconds:

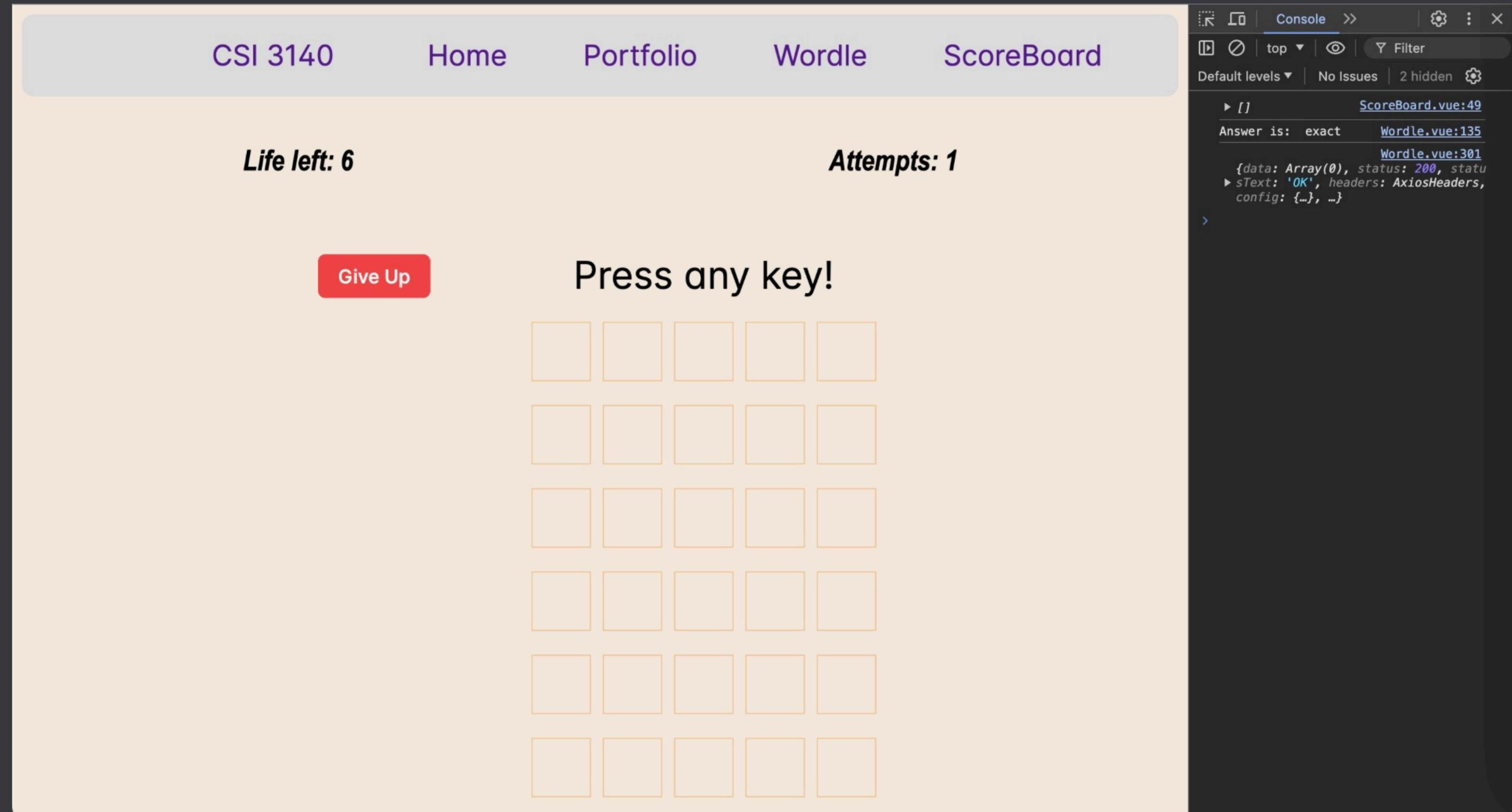
The screenshot shows a web application interface. At the top, there is a navigation bar with the following items: 'CSI 3140', 'Home', 'Portfolio', 'Wordle', and 'ScoreBoard'. The 'ScoreBoard' item is highlighted with a blue background. Below the navigation bar, the text 'Top 5 Scoreboard' is displayed in a large, bold, red font. Underneath this, the word 'Table' is centered in a bold, black font. A table structure is shown with two columns: 'Attempt Number' and 'Attempts'. A red button labeled 'Clean Board' is positioned below the table. To the right of the main content, there is a vertical panel for the 'ScoreBoard.vue:49' component. This panel includes a 'Console' tab, a 'Filter' section, and a message indicating 'No Issues'.

Attempt Number	Attempts
----------------	----------

Clean Board

Console >> | | Default levels | No Issues | 2 hidden | ScoreBoard.vue:49

The board is clean! Lets go back to the game:



CSI 3140 Home Portfolio Wordle ScoreBoard

Life left: 6 Attempts: 1

Give Up

Press any key!

Console

Default levels ▾ No Issues 2 hidden

ScoreBoard.vue:49

Answer is: exact Wordle.vue:135

Wordle.vue:301

{data: Array(0), status: 200, statu

► sText: 'OK', headers: AxiosHeaders, config: {}, ...}

The Attempt has been reset to 1, that means the button works!