CS5103 Practical 2

2400300341

Word Count: 666

Overview

The goal of this assignment was to create an interactive guessing game using the Vue framework. The player is provided with a string of text which is a random mathematical fact, with its letters replaced by an underscore character ("_"). The player's goal is to reveal the fact by pressing the provided letter buttons in as few presses as possible. If the player presses a letter that is not present in the fact text, then the player loses score. After sufficient guesswork, the player must enter their guesses for the fact and the number that is described by that fact in the provided fields. If the player guesses correctly, they receive bonus points.

Requirements

Done

- Fetche a fact about a random number from Numbers API (or the cache if the API fetch fails), and display it to the user with all letters replaced with underscores.
- Allows the user to guess a letter. If the letter appears in the phrase, the underscores should be replaced with the letter. This should be case insensitive
- Allows the user to submit their answer for the complete phrase and the number it describes, and tells them what they got correct (the phrase, the number, both or neither). This should end the game.
- Allows the user to start a new game without refreshing or reloading the page in any way.
- Implement the scoring system
- Session Scoreboard: Keep track of players scores until they refresh or reload the page.
- Sortable Scoreboard: Make the scoreboard sortable by score, number and phrase.
- Components based structure

Not done

- Accessibility: Make the page accessible to visually impaired users. Describe and justify your accessibility features in your report.
- Quiz feature implementation
- Accurately calculate current score

Code Design

Component Structure
The code follows a component-based structure:
- MainGame: main logic of the game
- Scoreboard: the scoreboard implementation
Components are helpful in decomposing the code into smaller modular pieces that are easy to work with. Each Vue component can be worked on in isolation from the rest of the web page, and implements its own content and logic. This separation allows for better maintainability and potential reusability of components.

Readability:
Vue has some very useful shorthands for directives, for example `@click` replaces `v-on:click`, and ';' replaces 'v-bind'. However, in order to facilitate easier readability and ensure clarity, I have intentionally chosen to avoid using the short hand.

Using the semicolon ':' in Javascript code after each line is not required, however, it helps make the code more readable and avoid errors.

Accessibility:
The scope of accessibility in this code is limited to UI design elements such as color contrast, font selection, text sizing, and clear language. However, the more important aspects of accessibility like using appropriate HTML attributes, ARIA roles, etc. have not been implemented in this code.

Vue vs Vanilla JavaScript
While most of the logic implementation would be the same if this game were written in vanilla Javascript, the DOM manipulation would have been significantly more cumbersome to implement. Vue handles reactivity rather seamlessly. Vue's many directives for DOM manipulation such as v-bind, v-on and v-model make DOM manipulation very easy and quick. Moreover, Vue's component system allows code reusability. If Quizzical decides to build another game, they can modify and use the Scoreboard component such that it can be reused for another completely different game.

Testing

Test	Expected Behaviour	Pass/Fail
Guess all letters (a-z)	Score goes to 0	Pass
Guess only wrong letters then get the phrase and number correct	Score goes to 4	Pass
Guess only correct letters then get the phrase and number wrong	Score goes to 0	Pass
Exponent numbers in current score for cases where numberOfWrongLetters has values of 5, 6, 11	Score goes to 0	Fail

Console.log statements used for testing:

```
console.log("Current Score:  " + currentScorePretty.value)console.log("current fact:   " +
currentFact.value.number + " " + currentFact.value.text)
console.log("# wrong letter: " + numberOfWrongLetters)
console.log("deduct by: " + 26 / numberOfWrongLetters)
console.log("current score: " + currentScore.value)
console.log("Wrong letters: " + Array.from(numberOfWrongLetters))
```