CMPS 115
Gabriela Medvetska (Product Owner)
Fawaz Chaudhry (Scrum Master)
Juan Lee
Kyungmo Kim
Jonathan Huey

Working Prototype Known Problems Report
English with Sammy

Expected Release: 22 July 2019

List of functions not working correctly:

1. **Website Overall**
    a. Known problems
        i. No support for all of the browsers
        ii. No common scaling for all screen sizes
        iii. Misaligned game cards
    b. input/action that causes failure
        i. When user trying to use Internet Explorer browser
        ii. When user trying to use a small screen, like a smartphone
        iii. When loading /games route for the first time
    c. Location of fault
        i. We used ES6 grammar such as using ‘let’ when defining a variable
        ii. We did not implement responsive design at this stage
        iii. Currently images are loaded from the database which is slower than building a grid in JavaScript
    d. Possible action for removal of fault
        i. Use the module name ‘Babel’ to solve compatibility problem in Internet Explorer browser
        ii. Use the bootstrap module to serve flexible design for various screen sizes.
        iii. Statically generate cards, use placeholder images while the database is loading, or use JS cache and reload the page again

2. **Crossword**
    a. Known problems
        i. In a very rare case, crossword boxes overlap with the navigation bar
        ii. Crossword is not scaling properly for small screens
        iii. Text mode is not working now (descriptions are not ready yet)
        iv. Crossword may not exist theoretically, so it won't display the board in the case
        v. Database latencies
    b. input/action that causes failure
        i. When the user trying to use a small screen, like a smartphone
        ii. When map generation algorithm fails
        iii. When data loading from the database is too slow
    c. Location of fault
        i. The minimum requirement width of crossword board
        ii. The board will not be displayed
        iii. The display or reaction of web (will be slow due to DB latency)
    d. Possible action for removal of fault
        i. We can use a small board with short words
        ii. Prepare a mathematically possible set of words to generate crossword board or regenerate if the algorithm fails
        iii. Use a database cache or buy a faster database

3. **Matching Letters**
    a. Known problems
        i. Database latencies
    b. input/action that causes failure
        i. When data loading from the database is too slow
    c. Location of fault
        i. The display or reaction of the web (will be slow due to DB latency)
    d. Possible action for removal of fault
        i. Use a database cache, refactor the code using Async-Await JavaScript functionality, or upgrade to a faster database

4. **Matching pictures**
    a. Known problems
        i. Database latencies

        ii. Pictures are draggable but do not do anything
    b. input/action that causes failure
        i. Whenever the picture is changed in the game
        ii. Dragging the Picture in the game
    c. Location of fault
        i. Where the database retrieves values
        ii. Within the code for allowing elements to be dragged and dropped
    d. Possible action for removal of fault
        i. Retrieve the values from the array and add a buffer
        ii. Redesign the drag and drop such that the picture can also be dragged towards the word or redesign the drag mechanic

5. **Typing racer**
    a. Known problems
        i. If the user typed all words but the user lost all lives already, the game does not finish immediately.
    b. input/action that causes failure
        i. When the player lost all lives while the last word is moving from left
    c. Location of fault
        i. fillC1 and fillC2 function checks the number of life when each word loaded on the screen. Therefore, if the last word is loaded on the screen, the game cannot check whether the user lost all lives.
    d. Possible action for removal of fault
        i. Add additional handler function that checks whether the number of the user’s life after the last word is loaded on the screen.