OHIO UNIVERSITY

Fibble

Final Presentation

Kavanaugh Frank / Matt Proboski / Nick Beoglos / Alec Martel / Amos Agyeman

Mac OS Mac OS Mac OS Windows

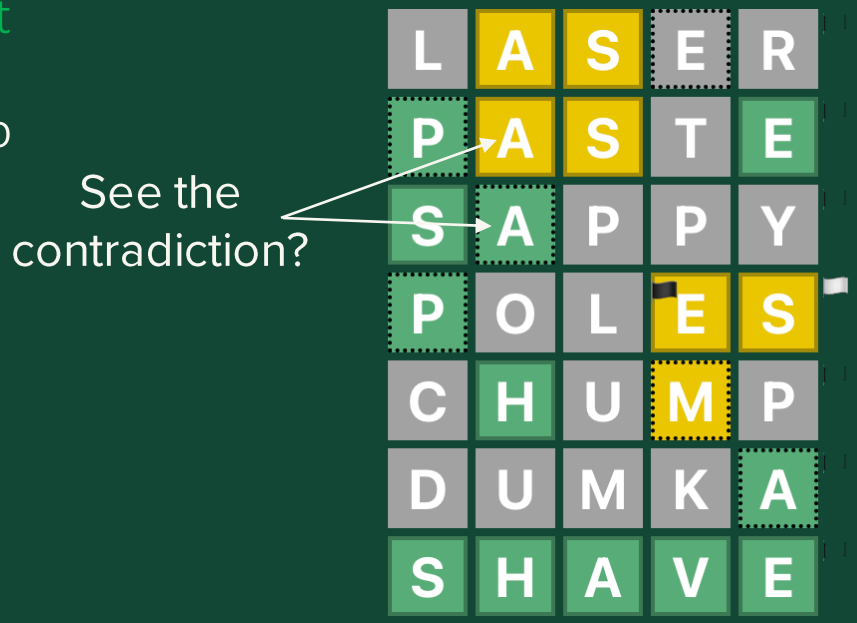
Meeting times: Sundays @ 4:00 // Mondays @ 3:30

FOREVER OHIO

What's Fibble?

- Well, what is Wordle?
  - Wordle is a game in which you must guess a random 5-letter word. When you make a guess, you get information on if the letters you used are in the right spot, in the wrong spot but right letter, or incorrect letter. You have 5 guesses to get the word right

- So then what's Fibble?
  - Fibble is a variant of Wordle where every time you guess, *one* piece of information is a lie.



Basic Project Information

- What are we making?
  - We're making a bot to play through a game of Fibble *efficiently,* not quickly. This means the bot will attempt to guess the correct word in as little guesses as possible, not as fast as possible.

- What'll it look like?
  - Our first Figma mockup

  - Options for Play type to change from Manual to Bot play

Technology Stack // Tool Chain

- Technology Stack
  - HTML - Standard Markup Language for developing websites
  - Java Script– Programming language primarily used for web application development
  - CSS – Use to style and change layout of webpages
  - NPM – Node Package Manager for Java Script
  - MySQL – Relational Database using SQL

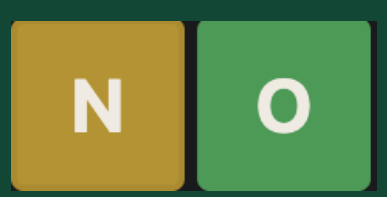
- Tool-Chain
  - Axios – Javascript Library for making HTTP Requests
  - Body-Parser – NPM Library used to process data sent through HTTP request
  - ExpressJS – Web app framework for Node.JS, used to build web application.
  - Nodemon - Utility that monitors and changes made in Node.JS applications
  - Node JS – Java Script runtime environment to execute code outside of a browser
  - ESLint – static code analysis for the Javasccript Language

Team Credentials

- Kavanaugh Frank
  - Worked primarily on the findBestWord function for the bot
- Nick Beoglos
  - Front end bug fixes, displaying word after guesses
- Matt Proboski
  - Fixed two bot functions
  - Updated read me
  - Bug fixes
  - optimization
- Alec Martell
  - Fixed logic errors in isValidBoard function
- Amos Agyeman Jr.
  - Worked on the isValid board function
  - Displayed the correct word once game is over

Meeting Attendance Sheet:

- Meeting 1 – 11/20/23 - All members attended
- Meeting 2 – 11/26/23 - All members attended
- Meeting 3 – 11/27/23 - All Members attended



- 1.) Bots first Guess will be a random word given to it in Fibble
- 2.) A bot will decide through algorithm, or hard-coded design what to guess next. This guess will run through Fibble algorithm to produce a lie and sent to server as an array.
  - ```
    let sendToServer = new Array(2);
    sendToServer[0] = userInput;
    sendToServer[1] = wordInfo;
    ```

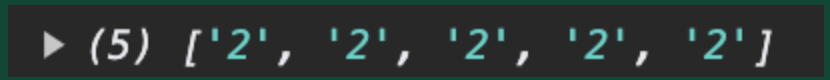
3.) For instance, if the word is slice,

and the first guess is sligh like this:

4.) Then return to server looks like this:

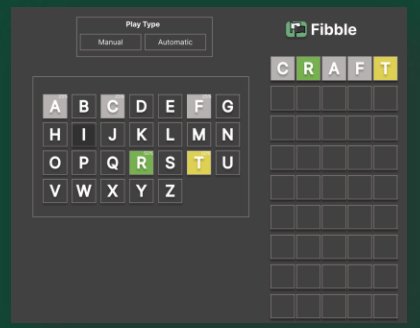
```
sligh   <- - - Return word
▶ (5) ['2', '2', '2', '1', '0']
```

^ String bot uses to make next guess
0 – Absent | 1 – Present | 2 - Correct
```

- Bot uses this sendToServer array and process it through our (soon to be developed) guessing algorithm to determine the perceived next guess. Of course, this guess can't be a word already used by the bot.
- Repeat until sendToServer[1] returns 

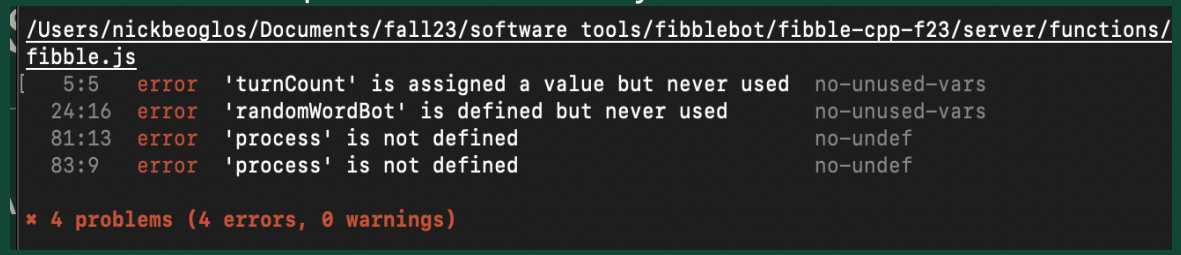
Toolkit

- We're using git with different branches (originally, each member had their own branch, but we've strayed away from that) and have branches made for specific purposes.
  - Backend testing, Frontend testing, cosmetic testing, bot testing, etc.)
- We're using ChromeDev tools to test outputs to the chrome terminal, aswell as VS terminal.
- Static Analysis Tool of Choice: ESLint, a static analyzer for Javascript
- Figma mockup as seen previously
- Other libraries/Frameworks included on our Fourth Slide (nodeJS, Axios, etc.)

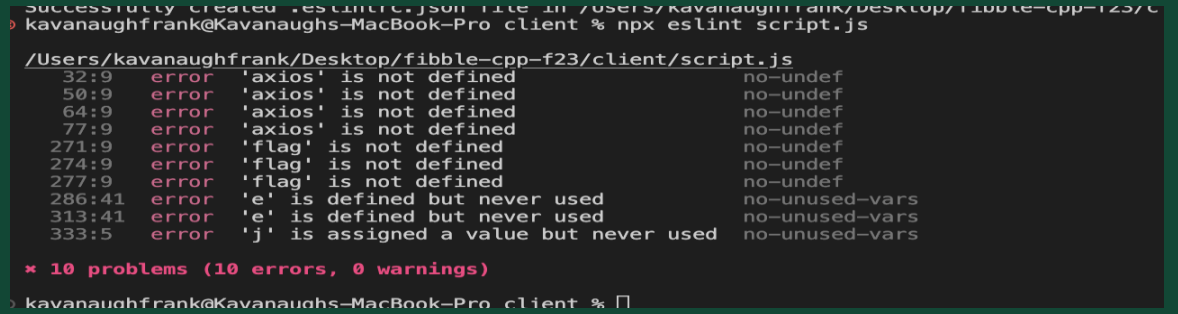


Static Analysis P1

- Nick Beoglos
  - Used ESLint Static Analyzer for Javascript, output from this Analysis:



- Kavanaugh Frank
  - Found and used ESLint Static Analyzer on my reformatted scipt.js file.



A lot of these issues were just unsued unused variables, and other non-critical errors. It helped to clean up unusedvariables, but the last two errors "process is not defined" was vital to the code working, so I ignored that analysis. Overall, a useful tool! Used the same tool as other team members, because its easily accesible and a "plug and play" type analyzer, very easy to get configured and go

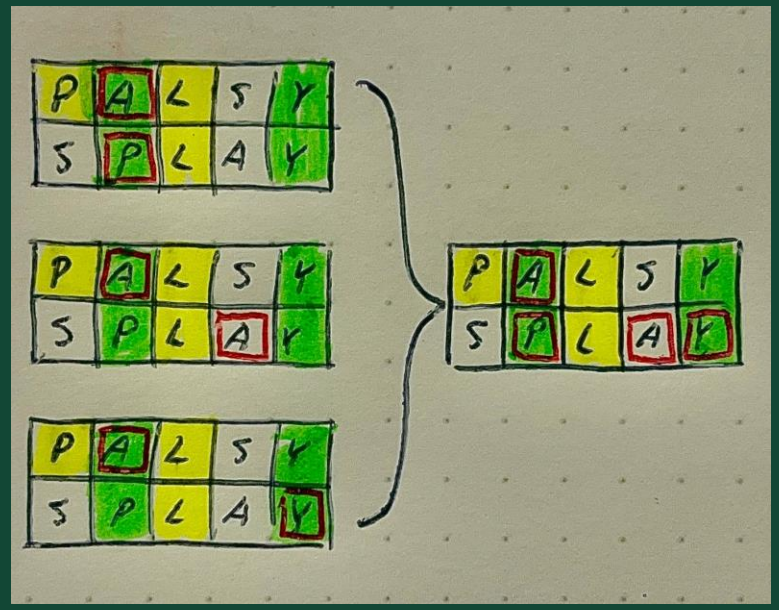
Fibble Bot Algorithm (high level overview)

Three Steps:

1. Find valid boards
2. Combine valid boards
3. Find valid word

What is a 'valid' board?

- A board where the combination of lies and truths do not contradict one another

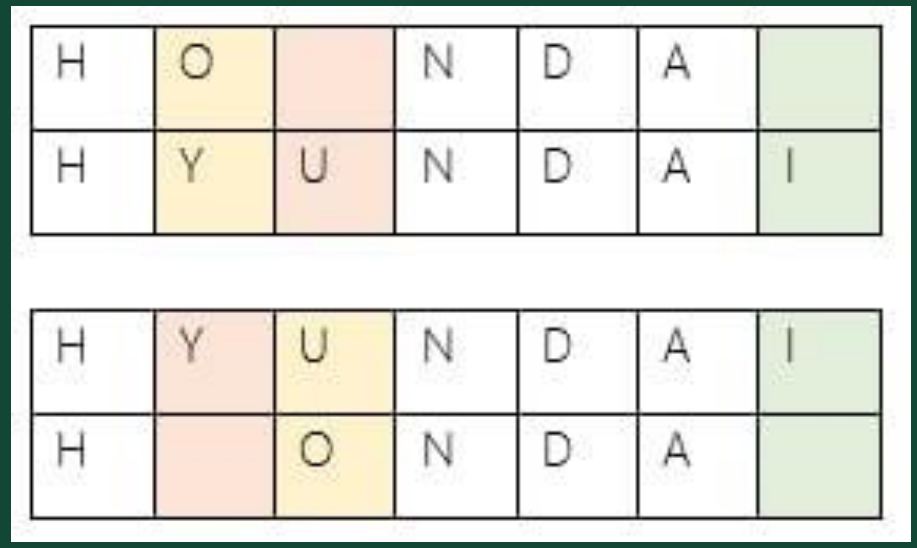


Constraints for the next word:

- Must include: P,L,Y,A
- Must exclude: S

Levenshtein Distance

- Using the Levenshtein distance equation to calculate the most similar word
- Our function can find words:
  - with a letter at any index
  - with a letter(s) at a specific index
  - without a letter at any index



Summary of Contributions

- Kavanaugh Frank
  - Database and server setup and connecting them with the frontend
  - Worked primarily on the findMostSimilar (Levenshtein Dist.) function for the bot
- Nick Beoglos
  - Front End design, JS of Fibble game, misc. Bug front-end bug fixing, theory
- Matt Proboski
  - Built the 1st iteration of the fibble game
  - Developed the algorithm that the bot is based off of
  - Implemented a lot of the algorithm into the bot
- Alec Martell
  - Theory implementation
  - Misc. back-end bug fixes
- Amos Agyeman Jr.
  - Misc. Frontend work
  - Documentation of code

Tool-Chain Reflection

- The Tool chain we selected provided a nice introduction into using those tools collaboratively for the first time. We have one team member focus mostly on the backend development, while the others focused on front-end and bot development. In this way, it was nice to collaborate with different parts of the project (front end and back end) in a way that felt natural. Most of us had never used these tools, and for some even the HTML, JSS, CSS, MySQL Technology Stack was new. Overall, it allowed us to get familiar with using them in a way that wasn't too foreboding or stressful. They worked well together, and we never really ran into any issues compatibility wise.

Team Work Reflection

- Members communicated well through Snapchat & Discord
- Members accomplished what was asked of them in a timely manner
- Consistent and extremely productive team meetings
- Overall good group relations and dynamics

Unit Testing – Jest - Easy to use framework built

specifically for JavaScript

- Test Case 1 -

```
test('racecar is an anagram', () => {
expect(areAnagrams("racecar", "racecar")).toBe(true);
});
```

- Need to make sure we can get anagrams.
- Input a known anagram, should return true

- Test Case 2 -

```
test('guess is not an anagram', () => {
expect(areAnagrams("guess", "guess")).toBe(true);
});
```

- Need to make sure we can get tell if something isnt an anagram
- Input a known non-anagram. With how function works, expect to be true.

Unit Testing – Jest - Easy to use framework built

specifically for JavaScript

- Test Case 3 -

```
test("the same two letters but at diff indexes", () =>{
expect(hasTwoLettersAtDifferentIndexes([
{index:0, letter:"a"},
{index:4, letter:"a"},
])).toBe(true)
})
```

- Need to know if a word has two of the same letters in it at dif. Indexes
- Put in two indexes, with the same letters. Should return true.

- Test Case 4 -

```
test("two different letters at diff indexes to insure that it does not return true", () =>{
expect(hasTwoLettersAtDifferentIndexes([
{index:0, letter:"b"},
{index:4, letter:"a"},
])).toBe(false)
})
```

- Need to know if a word doesn't have two of the same letters at different indexes.
- Put in two indexes with dif. Letters, should return false.

Unit Testing – Jest - Easy to use framework built

specifically for JavaScript

- Test Case 5 -

```
test('string with repeating characters', () => {
expect(hasRepeats("hello")).toBe(true);
});
```

- Simpler function to return if a word has repeating letters
- input word with known repeating letters. Should return true

- Test case 6 -

```
test('string without repeating characters', () => {
expect(hasRepeats("train")).toBe(false);
});
```

- Again, need to know if a word doesn't repeat letters
- Input a word with no repeated letters, should return false.

Unit Testing – Jest - Easy to use framework built

specifically for JavaScript

- Test Case 7 -

```
test('string with special characters and repeats', () => {
expect(hasRepeats("!@#$%^&*()hello1234567890hello")).toBe(true);
});
```

- Should return false even though there are sepcial characters. Might be an edge case if something goes wrong in the program.
- Input a string with special characets & repeated word. Should return true still

- Test Case 8 -

```
test('empty string', () => {
expect(hasRepeats("")).toBe(false);
});
```

- Empty word should return false for repeates.
- Input an empty string

Unit Testing – Jest - Easy to use framework built

specifically for JavaScript

- Test Case 9 -

```
test('letter at valid index matches', () => {
expect(hasLetterAtIndex("hello", 1, "e")).toBe(true);
});
```

- Need to know letters at indexes.
- Input a word, an index and the letter at the index. Should return true if function works

- Test Case 10 -

```
test('letter at valid index does not match', () => {
expect(hasLetterAtIndex("world", 2, "x")).toBe(false);
});
```

- Need to know if letters at indexes.
- Input a word, index and letter that isnt at that index. Should return false.

Unit Testing – Jest - Easy to use framework built

specifically for JavaScript

- Test Case 11 -

```
test('index is negative', () => {
expect(hasLetterAtIndex("testy", -1, "t")).toBe(false);
});
```

- Should still return false If even if a negative is somehow input
- Input a word, negative index, and whatever letter. Should always return false b/c index doesn't exist

- Test Case 12 -

```
test('index is larger than word length', () => {
expect(hasLetterAtIndex("train", 10, "e")).toBe(false);
});
```

- Should still return false if an input is larger than the word.
- Input a 5 letter word, index bigger than 4, and random letter. Should always return false

Unit Testing – Jest - Matt's unit tests

- Test Case 13 -

```
test('empty string', () => {
expect(hasLetterAtIndex("", 0, "a")).toBe(false);
});
```

- If an empty string is input, should always return false
- Empty string, random index, random letter, need to return false

- Test Case 14 -

```
test('letter is at index 0', () => {
expect(hasLetterAtIndex("start", 0, "s")).toBe(true);
});
```

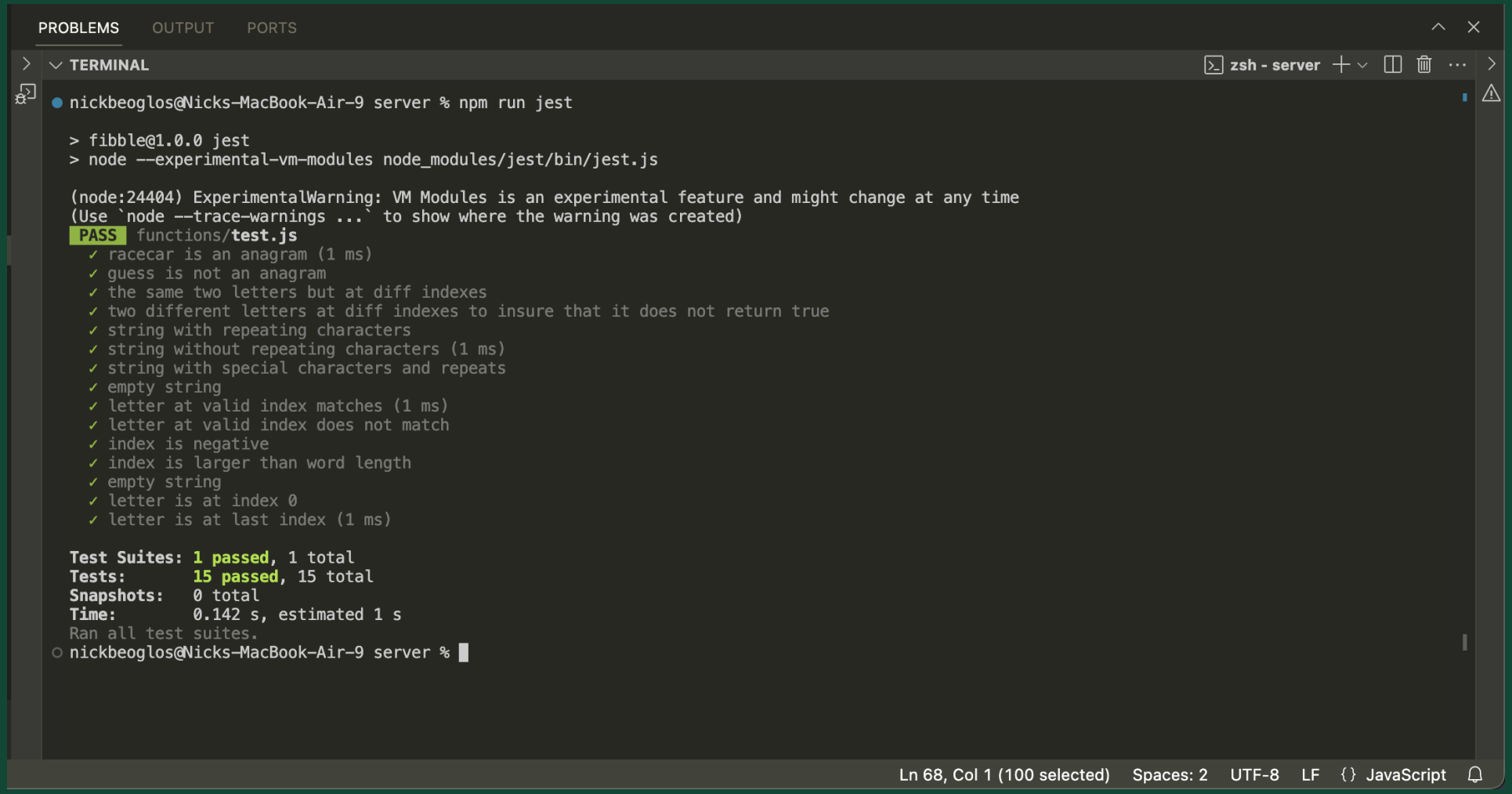
- If the check letter is first index, should still work.
- Input word, index 0, and the first letter in word. Should return true

Unit Testing – Jest - Matt's unit tests (cont'd)

- Test Case 15 -

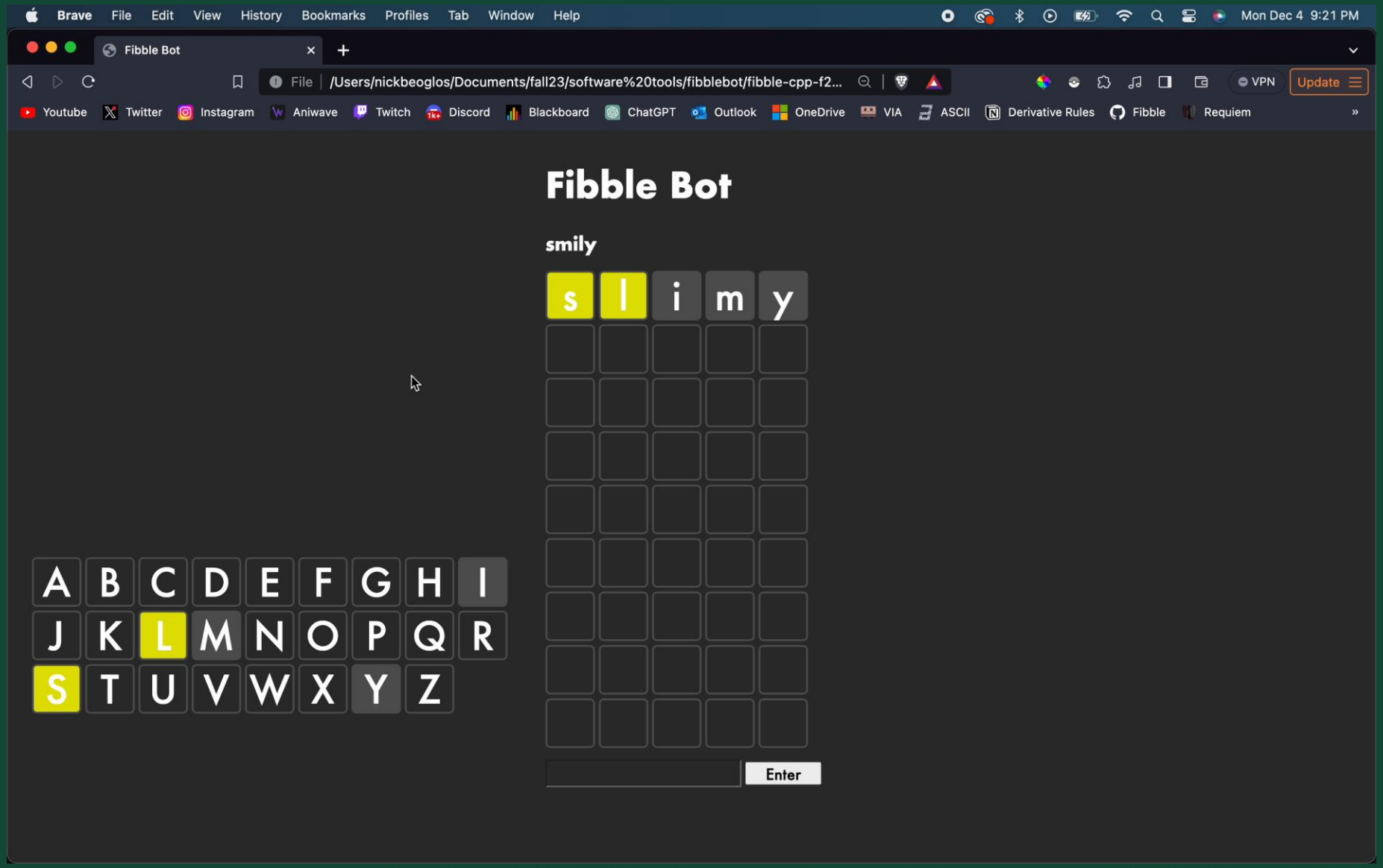
```
test('letter is at last index', () => {
expect(hasLetterAtIndex("poles", 4, "s")).toBe(true);
});
```

- Similar to last test case. If the index is the last, it should still work. Edge case
- Input letter, 4, and the last letter in the word. Should return true.


PROBLEMS
OUTPUT
PORTS
TERMINAL
zsh - server
nickbeoglos@Nicks-MacBook-Air-9 server % npm run jest
> fibble@1.0.0 jest
> node --experimental-vm-modules node_modules/jest/bin/jest.js
(node:24404) ExperimentalWarning: VM Modules is an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
PASS functions/test.js
✓ racecar is an anagram (1 ms)
✓ guess is not an anagram
✓ the same two letters but at diff indexes
✓ two different letters at diff indexes to insure that it does not return true
✓ string with repeating characters
✓ string without repeating characters (1 ms)
✓ string with special characters and repeats
✓ empty string
✓ letter at valid index matches (1 ms)
✓ letter at valid index does not match
✓ index is negative
✓ index is larger than word length
✓ empty string
✓ letter is at index 0
✓ letter is at last index (1 ms)
Test Suites: 1 passed, 1 total
Tests: 15 passed, 15 total
Snapshots: 0 total
Time: 0.142 s, estimated 1 s
Ran all test suites.
nickbeoglos@Nicks-MacBook-Air-9 server %
Ln 68, Col 1 (100 selected)
Spaces: 2
UTF-8
LF
{} JavaScript

OHIO UNIVERSITY

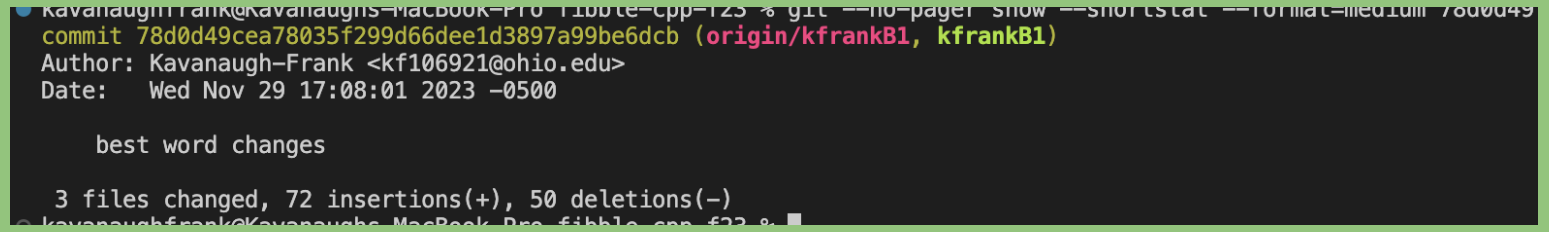
Demo -



FOREVER OHIO

Kavanaugh Frank - P101032426-Kavanaugh-Frank

- https://github.com/OU-CS3560/fibble-cpp-f23/commit/78d0d49cea78035f299d66dee1d3897a99be6dcb
- Made tweaks to the findBestWord function to be more accurate



Matthew Proboski - P101053601-matthewproboski

Most significant commit: fixed the two functions that prevented the bot from working properly

Link: https://github.com/OU-CS3560/fibble-cpp-f23/commit/f5e496e3aaca62a70a8cc70c70bca7bb62144b27

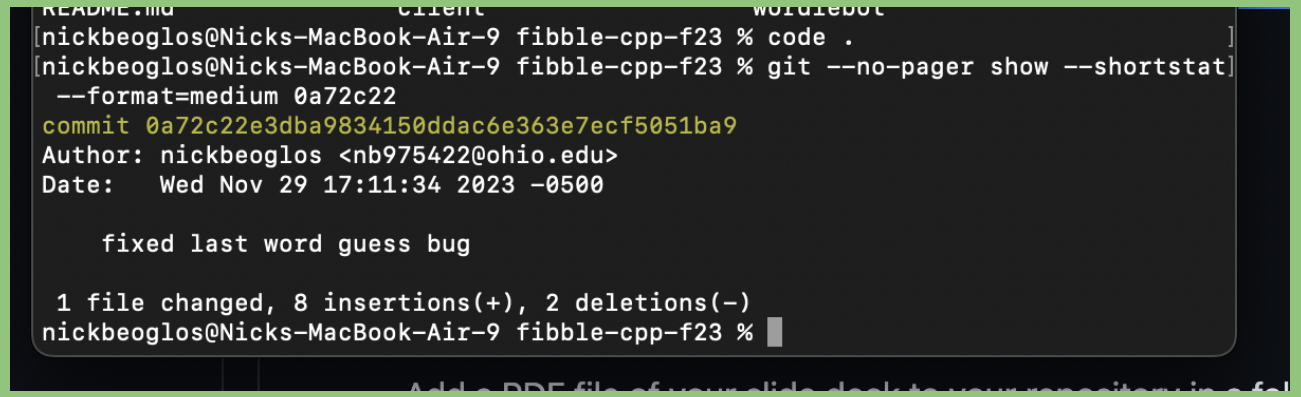
```
└─ git --no-pager show --shortstat --format=medium f5e496e3aaca62a70a8cc70c70bca
7bb62144b27
commit f5e496e3aaca62a70a8cc70c70bca7bb62144b27
Author: Matthew Proboski <mp636421@ohio.edu>
Date:   Mon Nov 27 15:14:27 2023 -0500

    fixed two functions so bot works properly

 2 files changed, 125 insertions(+), 186 deletions(-)
```

Nick Beoglos - nb975422 - nickbeoglos

Front end bug fixing, bot theory, bugs, file cleanups.



Alec Martell- Am754720 - Amartell22

- https://github.com/OU-CS3560/fibble-cpp-f23/commit/d38ebb4f721064047344fe1d57324ef671afa01b

```
PS C:\Users\123am\OneDrive\Documents\23 Fall Semester\Tools\fibble-cpp-f23> git --no-pager show --shortstat --format=medium d38ebb4
commit d38ebb4f721064047344fe1d57324ef671afa01b (HEAD -> git-bug, origin/git-bug)
Author: Alec Martell <am754720@ohio.edu>
Date:   Wed Nov 29 17:10:22 2023 -0500

    Yellow-Yellow bug fix

 1 file changed, 13 insertions(+), 13 deletions(-)
```

Amos Agyeman Jr. - aa852712 - AmosJr

- https://github.com/OU-CS3560/fibble-cpp-f23/commits/Famous

```
amosagyeman@Amoss-MacBook-Pro fibble-cpp-f23 % git --no-pager show --short
stat --format=medium d21b32a
commit d21b32a2fc95bdff8cfe148ff4b9dccade4dc5fe
Author: AmosJr <aa852712@ohio.edu>
Date:   Wed Nov 29 17:13:30 2023 -0500

    fetch update

 27 files changed, 272 insertions(+)
amosagyeman@Amoss-MacBook-Pro fibble-cpp-f23 %
```