# Software Requirements Specifications

# 

# 

# 

# 

# Word Statistics

# 

# 

# 

# 

## Prepared By: Sergio J Falcon

### CS5153 Software Engineering

## Formal Specifications

### wordFrequency:

Concept Definition:

COUNT == N1

WORD == [char]

FREQUENCY == SET (WORD, COUNT)

WORDLIST == seq1 (char\blank)

INCREASE == COUNT’ + 1

wordFrequency: word list => SET (WORD, COUNT)

\forall word: wordlist

If word in frequency then frequency = frequency ‘ + (word, increase)

Else

Frequency’ = (word, count = 1)

*/\*\**

*method title: wordFrequency*

*@params: array wordList*

*Description: Takes in an array of words and inputs*

*unique words into a map, if a word that is in the map*

*is inputted the count is increased*

*Returns: an associated array with the keys being the 'words'*

*and the occurrence of said 'word' count as its value*

*\*\*/*

export const wordFrequency = wordList => {

let frequency = new Map();

*//run through list of words in textfile*

const listOfWords = cleanList(wordList);

if(!wordList){

return [0];

}

else {

for(let i = 0; i < listOfWords.length; i++){

*//check if current word has already been entered into the hashmap*

*//if no, add into hashmap with a count of 1*

*//if yes, search hashmap and increase count by 1*

if(frequency.has(listOfWords[i])) {

let increase = frequency.get(listOfWords[i]) + 1;

frequency.set(listOfWords[i], increase);

}

else {

frequency.set(listOfWords[i], 1);

}

}

let newArray = Array.from(frequency, ([word, counter]) => ({word, counter}));

return newArray;

}

};

### wordCount:

Concept Definition:

TEXT == seq1 (char/space)

wordCount: TEXT -> N

\forall file: TEXT

wordCount file = # (words file)

*/\*\**

*method title: wordCount*

*@params: string text*

*Description: Breaks apart a large string using space, comma,*

*and a vertical slash as a delimiter to split apart words*

*in the string*

*Returns: integer of total words in the string*

*\*\*/*

export const wordCount = text => {

if (!text) {

return null;

}

else {

const words = text.split(/[\s,|]+/);

return words.length;

}

};

### lineCount:

Concept Definition:

TEXT == seq1 (char)

LINECOUNTER == N1

lineCount = TEXT -> N

\forall file: TEXT

lineCount file = #(nl file)

*/\*\**

*method title: lineCount*

*@params: string textFile*

*Description: takes in a large string and counts the number of new lines '\n'*

*Returns: integer of the counted newlines*

*\*\*/*

export const lineCount = textfile => {

*//count the number of '/n'*

if(!textfile){

return null;

}

else {

let lineCounter = 1;

for (let i = 0; i < textfile.length; i++) {

if (textfile[i] === '\n'){

lineCounter += 1;

}

}

return lineCounter;

}

};

### charCount:

Concept Definition:

CHAR = [CHAR]

TEXTFILE == seq1 (CHAR)

CHARCOUNTER == N

charCount = TEXT -> N

\forall file: TEXT

charCount file = # (chars file)

*/\*\**

*method title: charCount*

*@params: string textFile*

*Description: takes in a large string and counts the number*

*of characters in the string*

*Returns: integer of the counted character*

*\*\*/*

export const charCount = textfile => {

*//count the number of characters total in the file*

if(!textfile){

return null;

}

else {

let charCounter = 0;

for (let i = 0; i < textfile.length; i++) {

*//check if character is a letter or a number*

if(/^[0-9a-zA-Z]+$/.test(textfile[i])) {

charCounter += 1;

}

}

return charCounter;

}

};

### uniqueCharFrequency:

Concept Definition:

CHAR = [CHAR]

TEXTFILE == seq1 (char)

COUNT == N1

CHARFREQUENCY == SET (CHAR, COUNT)

INCREASE == COUNT + 1

uniqueCharFrequency = TEXTFILE -> SET(CHAR, COUNT)

\forall c: TEXT && c == char

If c in charFrequency then charFrequency’ = charFrequency + (c, increase)

Else

Frequency’ = (c, count = 1)

*/\*\**

*method title: uniqueCharFrequency*

*@params: string textFile*

*Description: counts the occurrence of each character in the*

*string using a map to hold the character as its key and*

*its frequency as its value*

*Returns: associated array of the characters as its keys and its frequency as its value*

*\*\*/*

export const uniqueCharFrequency = textfile => {

*//counter for unique characters in the file*

let charFrequency = new Map();

if(!charFrequency){

return null;

}

else {

for (let i = 0; i < textfile.length; i++) {

*//check if character is a letter or a number*

if(/^[0-9a-zA-Z]+$/.test(textfile[i])) {

*//check if character is unique, if it is increase its count in the map, if not, add it into the map*

if(charFrequency.has(textfile[i])) {

let increase = charFrequency.get(textfile[i]) + 1;

charFrequency.set(textfile[i], increase);

}

else {

charFrequency.set(textfile[i], 1);

}

}

}

let newArray = Array.from(charFrequency, ([character, counter]) => ({character, counter}));

return newArray;

}

};

### replaceWord

Concept Definition:

CHAR = [CHAR]

TEXTFILE == seq1 (char)

WORDTOBEREPLACED == seq1 (char)

REPLACINGWORD == seq1(char)

OTHERCHARACTERS :== . | , | \* | + | ? | ^ | $ | { | } | ( | ) | | | [ | ] | \ |

replaceWord = (TEXTFILE, WORDTOBEREPLACED, REPLACINGWORD) -> seq1 (char)

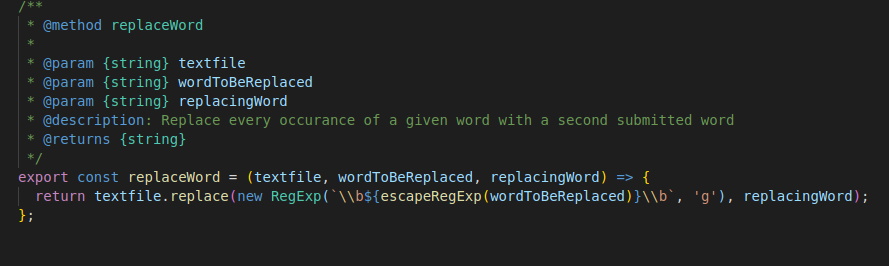
ONLYLETTERSANDNUMBERS == TEXTFILE / OTHERCHARACTERS

If WORDTOBEREPLACED IS IN ONLYLETTERS

Foreach WORDTOBEREPLACED

Split TEXTFILE at index of WORDTOBEREPLACED

TEXTFILE = TEXTFILE[0] + REPLACINGWORD + TEXTFILE[1]



## Unit Tests

### App.test.js

### 

### 

### 

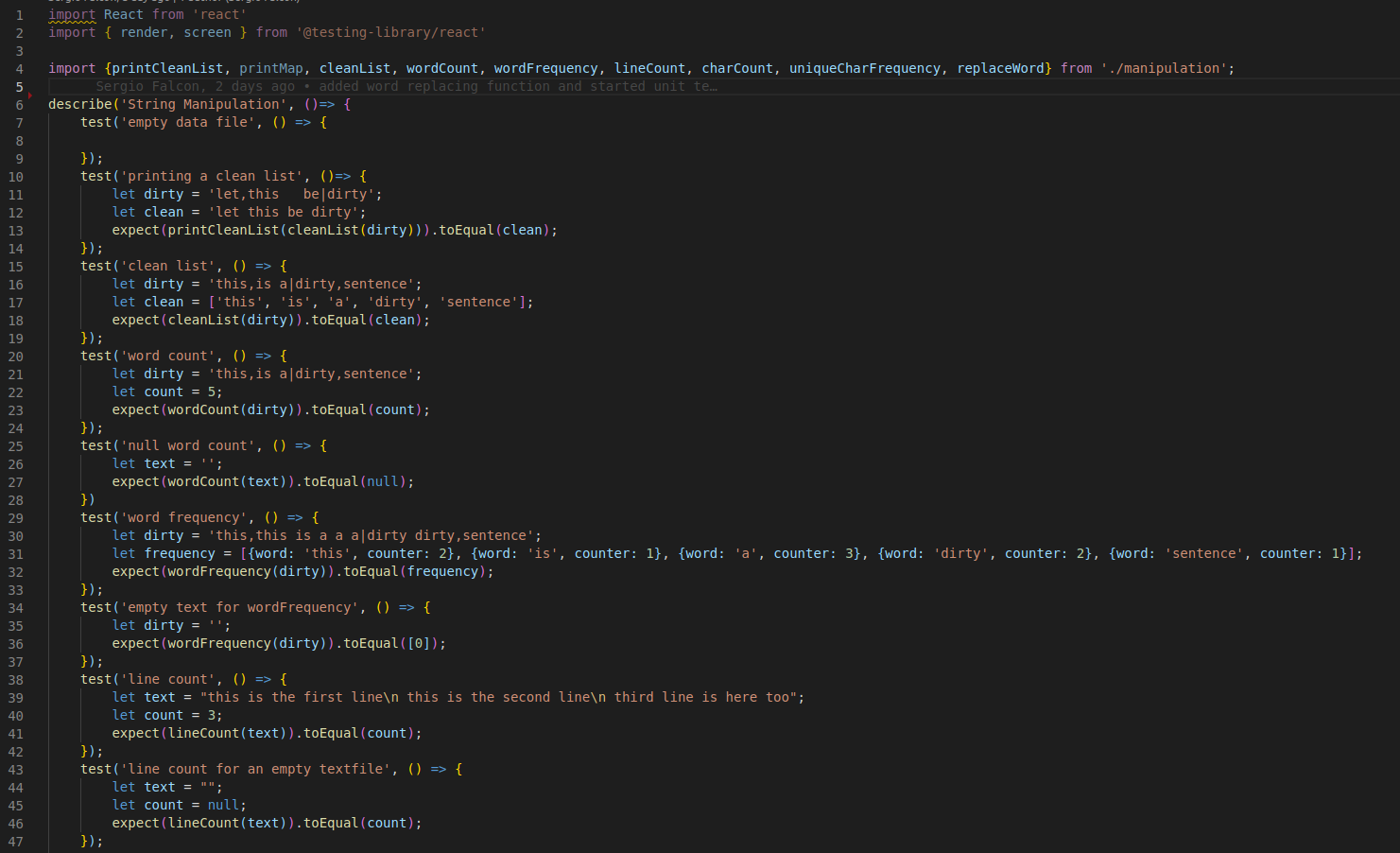
### 

### 

### 

### 

### Manipulation.test.js



## Design Change Report

### Sprint 2

Code remained the same.

### Sprint 3

Added the functionality of replaceWord to manipulation.js

Removed InputFile.js as I updated uploading a file

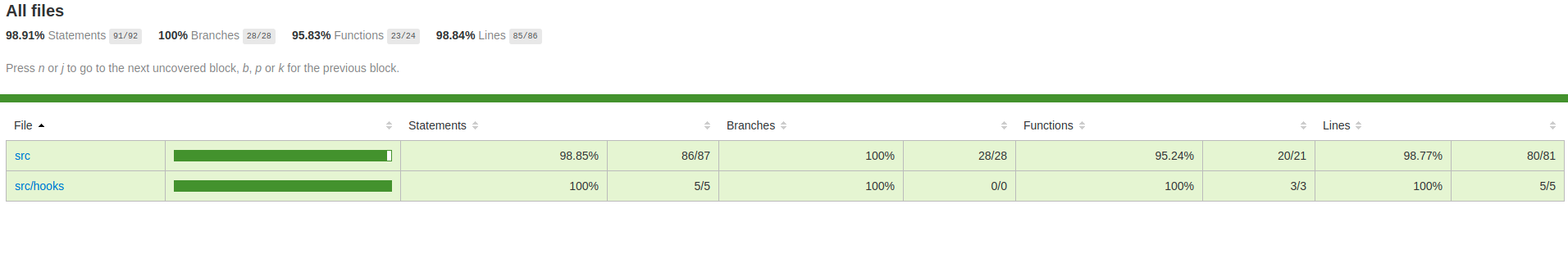
Installed Jest/React Testing Library to project or unit testing and test coverage

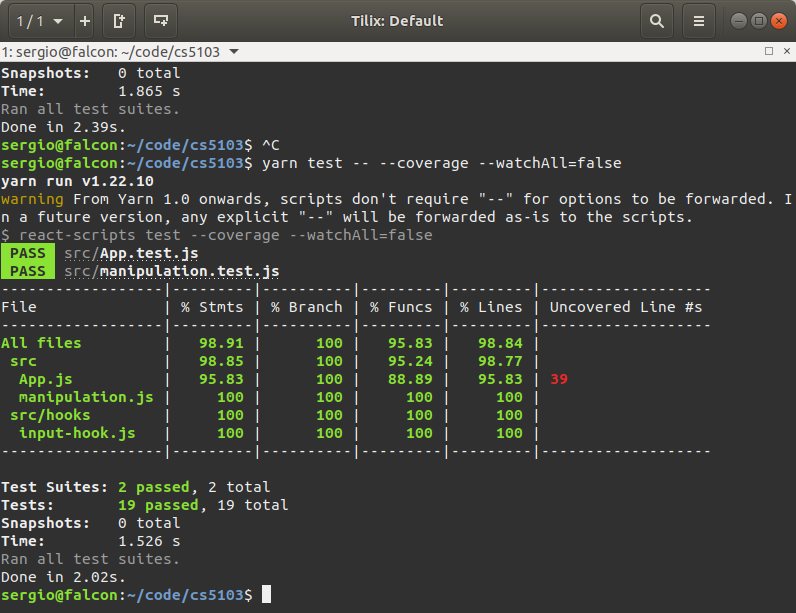
## Tool Application Report

### Test Coverage Provided by React Test Library/Jest

Running the Test Coverage script, updates the Coverage folder and provides a detailed HTML file that shows exact lines that have been covered or are missing.

File location for Test Coverages: cs5103/coverage/lcov-report/index.html





### SonarQube

I couldn’t quite get SonarQube to work on my computer. I installed PostgeSQL, set up access to the database by creating a user, table, giving the user permission, and running the server, and had it running on port 5432. I started running into problems when I installed SonarQube through the zip file, unpacked it and set up the variables as instructed through the documentation provided. However no matter what I did, it wouldn’t run, kept giving me an error of sonarqube.service start request repeated too quickly, ‘service-start-limit-hit’.