COMP.SE.200-2020-2021-1 Software Testing
Test report

Jussi Kujanen 273161
Veikko Hiltunen 283475
https://github.com/Kujanenj/SoftwareTesting

Tampere University
October 10th, 2020

Contents

1 Introduction **1**

2 Tools **1**

2.1 Jest 1

2.2 TravisCI 1

2.3 Coveralls 1

3 Test cases **2**

3.1 Unit testing 2

3.1.1 Test cases 3

3.2 Integration testing 6

3.2.1 Test cases 6

1 Introduction

The purpose of this document is to describe testing for an application which is used to sell food products. Testing is done only for the most important functions of the program due to time limitations. The aim of testing is to find problems that easily break the program. In addition, the purpose of this document is to help the reader understand how the testing and the functions to be tested have been roughly performed.

2 Tools

Below this section is a description of all the tools that will be used for this project. The tools have been chosen according to our own experience and interest.

2.1 Jest

Jest[1] was chosen as the testframework, since it works well with React, and we both have had some experience with Jest beforehand. Jest is used to run the test cases described in this document. A tutorial explaining how to use jest can be found here.[2]

2.2 TravisCI

TravisCI[3] is used during integration testing, because its automatic building and code testing during changes. It is automation provides immediate feedback which is very useful even in small projects like this.
It is also useful in continuous integration, since it is better to merge small code changes frequently than a large commit at the end of the development.

Because CI detects deficiences at the early stage in development, defects are typically less complex and smaller which makes them easier to resolve. Documentation and tutorials for Travis CI usage can be found on this site. [4]

2.3 Coveralls

Coveralls[5] is used to track test coverage. Coveralls was chosen for this task, since it is required by the course, and it is open source. A tutorial on how to use Coveralls can be found here[6].

3 Test cases

This chapter will cover unit and integration testing.
With the large amount of functions present, prioritisation is a must. The functions are rated on a scale of 1 to 5, 5 having the highest priority. Based on the description of the use case for the library, we can focus mostly on functions that handle string manipulation and validation, such as 'capitalize','map' and 'isEmpty'. This is because the part of the frontend to be tested is mostly based on handling user input.

Functions that are not essential for string manipulation have lower priority, and are not tested. A complete list of module prioreties and reasoning can be found as an attachment to this document. 1

3.1 Unit testing

Unit tests are run using jest. For each function tested, a table is made, explaining the tests run on that function, their inputs, outputs and expected results.

Each test is considered 'passed' once the output matches the expected output. If bugs or errors are encountered, the corresponding section has a note of it.

The functions to be tested were selected according to their importance to the program. Almost all functions that were in no way related to the operation of the program have been excluded from testing. These tests and test cases may still change during the testing itself and not all test methods may be actually implemented.

3.1.1 Test cases

Here can be found all the test cases in a listed form, divided into sections based on the module tested. During testing new test cases can be written even if they are not listed in these tables.

Function: add.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	Positive numbers	0 + 5	5	-	No
2	Negative numbers	-5 + 6	1	-	No

Function: capitalize.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	all characters capitalized	TESTRING	Teststring	-	No

Function: compact.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	large item_id list	[0, 1, ", 5, 6]	[0,1,5,6]	-	No

Function: filter.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	filter products	[1,2,3,4,5,-5,-3]	[1,2,3,4,5]	-	No

Function: map.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	similar actions for array nodes	[product1, product2], capitalize	[Product1, Product2]	-	No

Function: slice.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	remove certain amount from product list	[1, 2, 3, 4] , 2	[3, 4]	-	No

Function: toNumber.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	modify string to number	'3.2'	3.2	-	No

Function: toString.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	multiple products into a single string	[p1, p2, p3, p4]	'p1, p2, p3, p4'	-	No

Function: endWith.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	Valid input	"abc","b"	False	-	No
2	Invalid input	abc,b	???	-	No

Function: get.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	Non existent field	{a},'a.b'	???	-	No
2	valid Input	{a:b:1},'a.b'	???	-	No

Function: isEmpty.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	Null	null	True	-	No

Function: reduce.js

Num	Test case	Test data	Expected result	Actual result	Passed
1	Sum of an array	[1,2,3],add,0	6	-	No

3.2 Integration testing

Integration tests are also run using jest, but also with TravisCI, and Coveralls. In these tests, the modules tested in unit testing phase are integrated, and tested together. Each time a new commit is made to the master branch, Travis will run all tests and sends a coverage report to Coveralls. The formatting of the tables will be the same as in unit testing.

3.2.1 Test cases

Here can be found all the integration test cases.

Test case	Test data	Expected result	Actual result	Passed
Add strings	add(toNumber(2), toNumber(5))	7	-	No
Add bad array values	reduce(compact([1,2,"]),add,0))	3	-	No
Slice of objects as string	toString(get(slice([{a:b},{b:a}],1)a.b))	"b"	-	No
Compact falsy list	Compact(Map([1,2,3,4],isEmpty))	[]	-	No
Filter ending with	Filter(["ab","ac","bb"], endWith("b"))	["ab","bb"]	-	No

References

[1] [Online]. Available: https://jestjs.io/

[2] [Online]. Available: https://www.valentinog.com/blog/jest/

[3] [Online]. Available: https://travis-ci.org/

[4] [Online]. Available: https://docs.travis-ci.com/user/tutorial/

[5] [Online]. Available: https://coveralls.io/

[6] [Online]. Available: https://github.com/dwyl/learn-coveralls.io

Attacments

Priority table 1

Table 1: Priority

Module Name	**Importance Rating**	**Reason**
add	4	Adding prices
at	1	Not needed
camelCase	1	Not needed
capitalize	3	String manipulation
castArray	1	Not needed
ceil	1	Not needed
chunk	1	Not needed
clamp	1	Not needed
compact	2	Might need to remove empty values
countBy	1	Not needed
defaultTo	1	Not needed
defaultToAny	1	Not needed
difference	1	Not needed
divide	1	Not needed
drop	1	Not needed
endsWith	3	Searching for products
eq	1	Not needed
every	1	Not needed
filter	3	Searching products
get	2	object manipulation
isArguments	1	Not needed
isArrayLike	1	Not needed
isArrayLike	1	Not needed
isBoolean	1	Not needed
isBuffer	1	Not needed
isDate	1	Not needed
isEmpty	2	Empty input fields
isLenght	1	Not needed
isObject	1	Not needed
isObjectLike	1	Not needed
isSymbol	1	Not needed
isTypedArray	1	Not needed
keys	1	Not needed
map	3	String manipulation
memoize	1	Not needed
reduce	4	String manipulation, adding prices
slice	2	Remove items from cart
toFinite	1	Not needed
toInteger	2	Input validation
toNumber	2	Input validation
toString	2	Input validation, string manipulation
upperFirst	1	Not needed
words	1	Not needed