Name: Lone Nilsson 2019-03-07
Mail: ln222tu@student.lnu.se
Github repo: https://github.com/onlylonely1986/ln222tu_1dv600

ASSIGNMENT 3, SOFTWARE TECHNOLOGY 7, 5 hp

Test plan

Objectives

To learn more about testing software and get knowledge about estimates and planning my testing, also use suitable tools for that.

Testing generally is about get rid of faults, save time from searching manually after bugs and it also builds confidence in the software.

The primary objectives is to test the code that was implemented during assignment 2 and get hints about what to solve in next iteration.

What to test

Dynamically tests:

- Manually test cases (2)
- Automated Unit tests (Create two automated unit-tests for each method in a total of four test methods)
- Automated Unit tests (implement a new method and test it before implementation is finished)

Statically tests:

- None

Time plan:

Task	Estimated	Actual
Manual TC 1	1 h	30 min
Manual TC 2	30 min	30 min
Unit test 1.1 and 1.2	2 h	1 h
Unit test 2.1 and 2.2	1,5 h	30 min
Running manual tests	20 min	10 min
Unit test for new method 3.1, 3.2	2 h	5 min
Implement code	1 h	1 h
Run unit test	20 min	15 min
Test report	1,5 h	1 h

Reflection

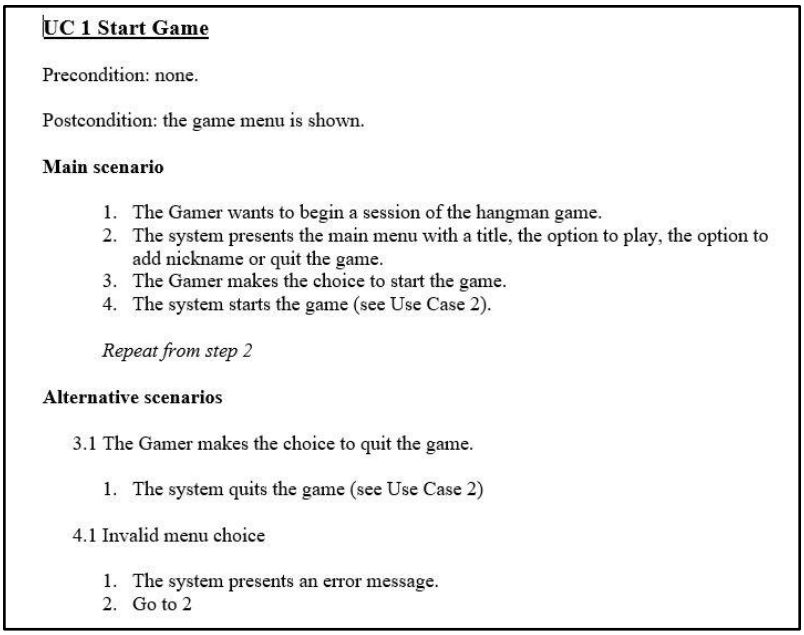
It was hard to do estimates in this task because it was so much new to understand and get knowledge about. But I can see that I was need less time than I first thought and that is a

better way, to estimate a bit too much instead of having very good thoughts about myself and then take much longer time than I first estimated.

Manual test cases

TC1.1 Start Game successful

Use Case: *UC1 Start Game*



UC 1 Start Game

Precondition: none.

Postcondition: the game menu is shown.

Main scenario

1. The Gamer wants to begin a session of the hangman game.
2. The system presents the main menu with a title, the option to play, the option to add nickname or quit the game.
3. The Gamer makes the choice to start the game.
4. The system starts the game (see Use Case 2).

Repeat from step 2

Alternative scenarios

3.1 The Gamer makes the choice to quit the game.

1. The system quits the game (see Use Case 2)

4.1 Invalid menu choice

1. The system presents an error message.
2. Go to 2

Scenario: Start game successful

The main scenario of UC1 is tested where a user starts a new game.

Precondition: Run npm start in terminal

Test steps

- System shows a menu with four alternatives
- Enter P for play

Expected

- System should show a new word to guess and stand ready for input characters

TC1.2 Start Game unsuccessful

Use Case: UC1 Start Game

Scenario: Start game unsuccessful

The main scenario of UC1 is tested where a user starts a new game.

Precondition: Run npm start in terminal

Test steps

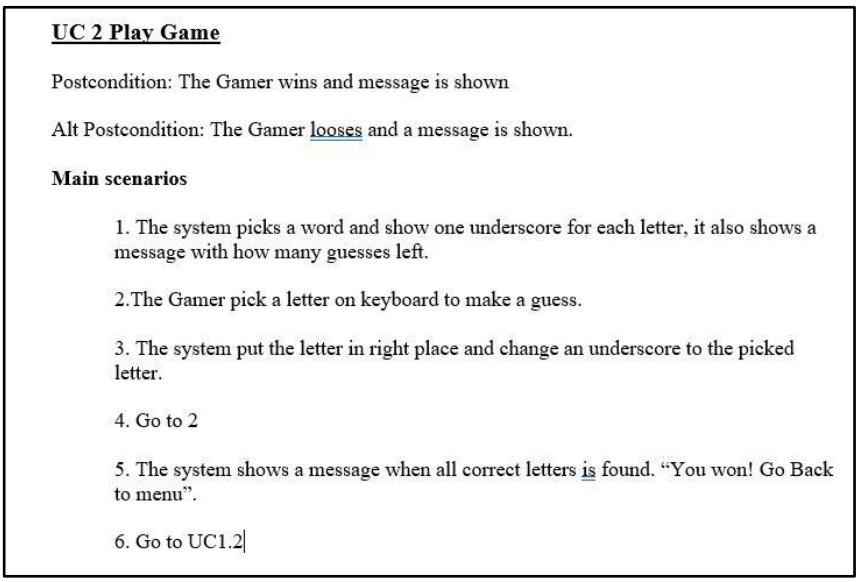
- System shows a menu with four alternatives
- Enter O for mistake

Expected

- System should show a message about wrong input and still show menu and wait for new input

TC2.1 Play Game successful

Use Case: UC2 Play Game



UC 2 Play Game

Postcondition: The Gamer wins and message is shown

Alt Postcondition: The Gamer looses and a message is shown.

Main scenarios

1. The system picks a word and show one underscore for each letter, it also shows a message with how many guesses left.

2.The Gamer pick a letter on keyboard to make a guess.

3. The system put the letter in right place and change an underscore to the picked letter.

4. Go to 2

5. The system shows a message when all correct letters is found. "You won! Go Back to menu".

6. Go to UC1.2

Scenario: Play game successful

The main scenario of UC2 is tested where a user plays a game.

Precondition: Choose test-version of word-bank, so you can be sure about what word it is. Run npm start in terminal, pick P for play.

Test steps

- System shows five underscores for the word HORSE, and also 6 tries left
- Enter H
- Enter O
- Enter R
- Enter S
- Enter E

Expected

- System should show a message about winning game and show the start menu again

Name: Lone Nilsson
Mail: ln222tu@student.lnu.se
2019-03-07
Github repo: https://github.com/onlylonely1986/ln222tu_1dv600

```
Guess a letter:
H O R S _   lives: 6.
 [ 'H', 'O', 'R', 'S' ]
e
You picked: e

Guess a letter:
H O R S E  lives: 6.
 [ 'H', 'O', 'R', 'S', 'E' ]

YOU WON, GOOD JOB!

Welcome to your favorite HANGMAN GAME!

MENU:
PLAY (P)              HELP (H)
NICK (N)              QUIT (Q)

Do your choice: >
```

TC2.1 is OK.

Test Report

Test	UC1	UC2
TC1.1	1/OK	0
TC1.2	0	0
TC2.1	0	1/OK
COVERAGE & SUCCESS	1/OK	1/OK

Comment

New implementation is needed, because TC1.2 did not pass. The user is out from application if the user makes a mistake and input the wrong button, that is not a good thing. For a higher quality and coverage there should be more tests in every Use Case.

Automate unit tests

Unit test 1, on method “help”.

1.1 Should return a string.

1.2 Should return right value.

Name: Lone Nilsson
Mail: ln222tu@student.lnu.se
2019-03-07
Github repo: https://github.com/onlylonely1986/ln222tu_1dv600

Method “help”:

help.js

```
'use strict'
// const menu = require('./menu')

const helpMe = () => {
  return 'Follow the instructions to play this game in terminal.\nYou are supposed to gues
}
helpMe()
// menu.runMenu()

module.exports = { helpMe }
```

Unit test 2, on method “quit”.

2.1 Should return a string.

2.2 Should return right value.

Method “quit”:

quit.js

```
function sayGoodBye () {
  return 'Bye'
}

module.exports = { sayGoodBye }
```

Unit test 1 and 2

1.1, 1.2 (“help”) and 2.1, 2.2 (“quit”):

gameTest.js

```
'use strict'
const mocha = require('mocha')
const assert = require('chai').assert
const help = require('../lib/help')
const quit = require('../lib/quit')

mocha.describe('Help', function () {
  mocha.it('Help should return a string', function () {
    assert.isString(help.helpMe(), 'string')
  })
  mocha.it('Help should return a message to guide the player', function () {
    assert.equal(help.helpMe(), 'Follow the instructions to play this game in terminal.\nYou are supposed to guess th
  })
})

mocha.describe('Quit', function () {
  mocha.it('Quit should return a string', function () {
    assert.isString(quit.sayGoodBye(), 'string')
  })
  mocha.it('Quit should return a message with Bye', function () {
    assert.equal(quit.sayGoodBye(), 'Bye')
  })
})
```

Name: Lone Nilsson
Mail: ln222tu@student.lnu.se
Github repo: https://github.com/onlylonely1986/ln222tu_1dv600

Test result for unit test 1 and 2:

```
C:\Users\onlyl\ln222tu_1dv600\hangman-game>npm test

> ln222tu_1dv600@1.0.0 test C:\Users\onlyl\ln222tu_1dv600\hangman-game
> mocha



  Help
    √ Help should return a string
    √ Help should return a message to guide the player

  Quit
    √ Quit should return a string
    √ Quit should return a message with Bye


  4 passing (16ms)


C:\Users\onlyl\ln222tu_1dv600\hangman-game>
```

Unit test 3, on method “nick”.

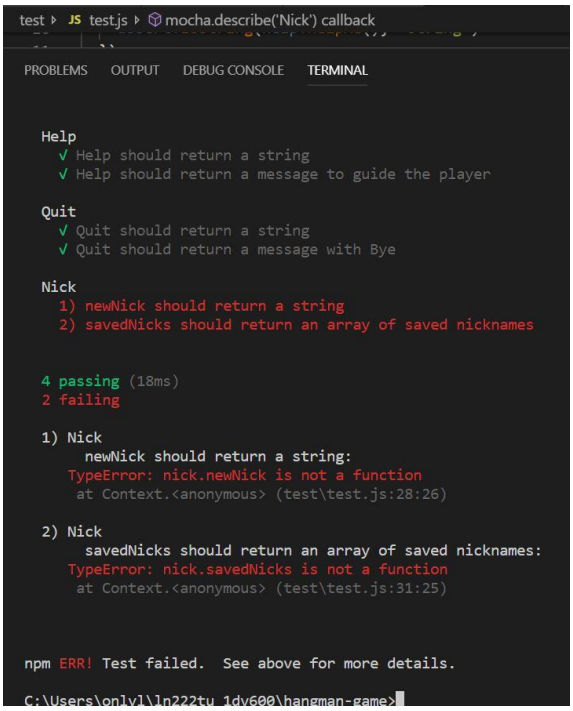
3.1 Should return a string.

3.2 Should return an array.

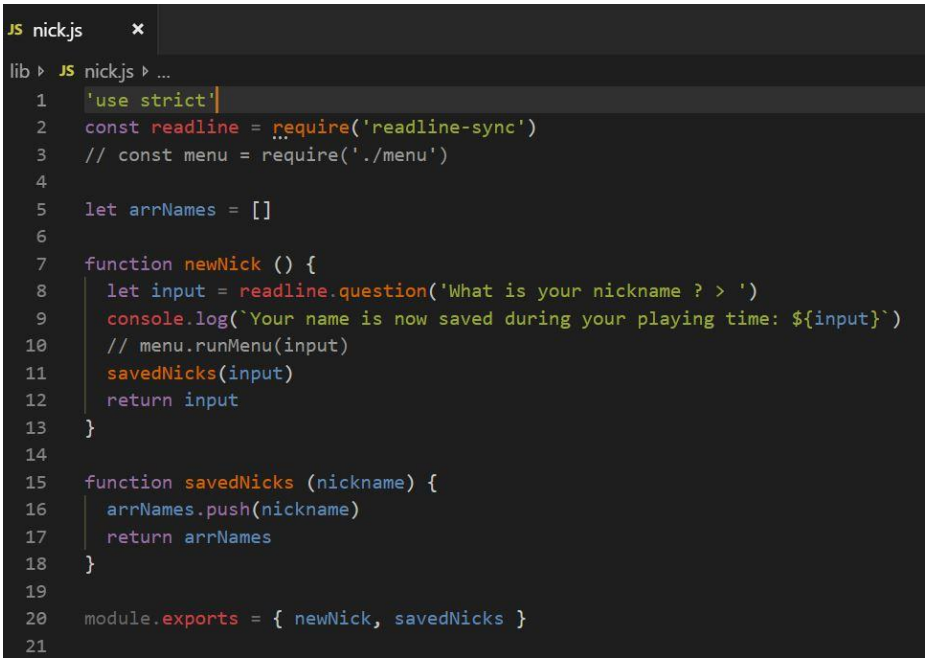
```
25
26  mocha.describe('Nick', function () {
27    mocha.it('newNick should return a string', function () {
28      assert.isString(nick.newNick(), 'string')
29    })
30    mocha.it('savedNicks should return an array of saved nicknames', function () {
31      assert.isArray(nick.savedNicks(), 'Array')
32    })
33  })
34
```

Name: Lone Nilsson
Mail: ln222tu@student.lnu.se
Github repo: https://github.com/onlylonely1986/ln222tu_1dv600

Test result on unit test 3 before implemented the code:



Implemented code for new method:



Name: Lone Nilsson
Mail: ln222tu@student.lnu.se
Github repo: https://github.com/onlylonely1986/ln222tu_1dv600

Test result for new method after implemented code:

```
C:\Users\onlyl\ln222tu_1dv600\hangman-game>npm test

> ln222tu_1dv600@1.0.0 test C:\Users\onlyl\ln222tu_1dv600\hangman-game
> mocha



  Help
    √ Help should return a string
    √ Help should return a message to guide the player

  Quit
    √ Quit should return a string
    √ Quit should return a message with Bye

  Nick
What is your nickname ? > ooj
Your name is now saved during your playing time: ooj
    √ newNick should return a string (982ms)
    √ savedNicks should return an array of saved nicknames


  6 passing (1s)


C:\Users\onlyl\ln222tu_1dv600\hangman-game>
```

Reflections

Even if I did this task with preferably easy and simple test. I can see that I have learned a lot about it, and I have gained new knowledges. In the next project I am supposed to develop I am going to plan more and estimate times before my work, I am also going to use testing more. And now, I know a bit how I can use tests, so I am going to think about how to code methods and classes to easier test them afterwards. In this iteration and the work with the tests I have realised a couple of problems to solve in next iteration. Some parts of my planned functionality in my program is not working as it should, so it is a bit more work to do.