HANGMAN DEVELOPMENT PROJECT

Fredrik Langå

Linnaeus University

2019-02-08

CONTENTS

1. REVISION HISTORY

Date	Version	Description	Author
190207	**1.0**	**First edition**	**fl222pw**
190222	**1.1**	**Finished second iteration, adding to this document**	**fl222pw**
190405	**1.2**	**Finished third iteration, adding to this document**	**Fl222pw**
190418	**1.3**	**Finished fourth and final iteration** • **Adding UML modelling** • **Adding Tests** • **Updated Use Case diagram** • **Updated State Machine diagram** • **Updated Class diagram** • **Adding time logs**	**fl222pw**

2. GENERAL INFORMATION

Project Summary	
Project Name	**Project ID**
Hangman	fl222pw_1dv600
Project Manager	**Main Client**
Fredrik Langå	Linnaeus University
Key Stakeholders	
Project Manager Developer End-User	
Executive Summary	
Creation of the game Hangman, playable in the terminal with ascii-style graphics. This is done for the purpose of the developer learning about planning, designing, implementing and testing of software.	

3. VISION

A game is to be made. A game of guessing, wit, with intriguing suspense where any missteps have consequences, making each successive guess all the more difficult to make.

We are to create the game *Hangman,* working in the terminal window and with ascii-style graphics. It's not a ground-breaking idea for a game, but it is an old and proven way of entertainment, and if you can get the basics right everyone can enjoy it. Hangman is a game of guesses, where guessing the correct word without giving a wrong guess too many times saves a digital person's life. Of course, the stakes aren't really that high, but it lends credence to wanting to give the correct answer while also keeping guesses to a minimum, keeping a competitive edge!

Reflection

Writing a vision was a bit harder than expected. To "inspire" other people reading this document and vision I thought a bit of humour and exaggeration to start with might help. I also hoped it would inspire me to help me write the vision itself. It is also quite difficult to describe a project like Hangman in terms to help with inspiring people to want do make it, but at the same time a lot of what we are going to do in our professional lives are probably not going to be a super-exciting project after another.

4. PROJECT PLAN

4.1 Introduction

The purpose of this project is to create the game *Hangman* working in the terminal window/command prompt and with ascii-graphics, with user-input from the terminal. In Hangman, the game takes a random word from a pool of words, where the user is supposed to guess which letters the word contains. The amount of guesses the user gets is made up of how many parts to the hanging pole and the character.

4.2 Justification

This application has been requested by the teachers of the course 1dv600 as a means to learn the ins and outs of planning, designing, developing and testing a bigger type of project.

4.3 Stakeholders

Manager; which is me, planning the project and makes sure deadlines are met.

Developer; which is me, designs and develops the features of the game with maintainability in mind.

User; which is me, has expectations on the delivered program.

4.4 Resources

Development will be done with Visual Studio Code as IDE, using JavaScript as the programming language and running server-side using Node.js.

People-resources are extremely limited; project planning, designing, developing and testing is done by one person with limited time due to other factors involving the course, for example lectures and reading course literature.

Little to no money available, and a small timeframe to completion.

4.5 Hard- and Software requirements

Computer, access to terminal/command prompt, Node.js (LTS) installed, ability to install node modules, familiarity with command-line use.

4.6 Overall Project Schedule

Week	Theme	Deadline
4		
5	**Process**	
6	**Model**	**Assignment 1**
7	**UML**	
8	**Design**	**Assignment 2**
9	**Test Plan**	
10	**Test**	**Assignment 3**
11	**Project**	**Present git-repo**
12	**Project**	**Hand in project**

4.7 Scope, Constraints and Assumptions

4.7.1 Scope

- Menu with user-input from keyboard.
- Menu containing “Play game” and “Quit”.
- Ability to choose difficulty
- The user has nine guesses, with each guess representing, in order, the vertical pole, horizontal pole, noose, head, body, left arm, right arm, left leg and right leg.
- The word to be guessed displayed as “_” for each letter.
- Each wrong guess to write the corresponding graphic to the terminal.
- Each correct guess to replace “_” with the letter in the correct position of the word to be guessed.
- On correct word display the full word and give the user options to “Play again” and “quit”.

4.7.2 Constraints

- Small timeframe to complete necessary documentation and development.
- Only one person acting as project manager, developer and end-user.
- Implementing necessary functions to be re-playable.
- Local or online word database?

4.7.3 Assumptions

- The user to have some degree of computer knowledge.
- The user to have the ability to install necessary applications and packages.
- That the time left is enough to complete the assignment.

Reflection

The project plan I thought was quite difficult to write. I haven’t thought about planning in this way and amount of detail before, so it was quite a challenge. Still, I think it gives a good idea how it is when working professionally, albeit in a much smaller scope. I still feel I’m missing some ideas and will probably add things I didn’t think about to the project plan as we get deeper in the course and iterations. It feels like a good learning experience to get more structured in the way I approach new projects.

5. ITERATIONS

5.1 Iteration 1

Objective	Est. Hrs
Reading of course literature chapter 2, 3, 22 and 23	10
Viewing of online lectures	4
Course lectures on campus	6
Creation of repository on GitHub	1
Writing Vision document	3
Writing Project Plan	6
Writing Iterations	4
Writing Risk Analysis	2
Filling in Time Log	2
Writing reflections	2
Total time Estimate	40

5.2 Iteration 2

Objective	Est. Hrs
Reading of course literature chapter 4, 5, 6, 7, 15 and 20	20
Viewing of online lectures	4
Course lectures on campus	8
Design Features to Hangman	2
Implement features to Hangman	10
Developing project plan	4
Total time Estimate	48

5.3 Iteration 3

Objective	Est. Hrs
Reading of course literature chapter 8	2
Viewing of online lectures	6
Course lectures on campus	8
Add additional features	10
Create test	4
Plan, perform and document tests	15
Total time Estimate	45

5.4 Iteration 4

Objective	Est. Hrs
Course lectures on campus	4
Combine all previous documents	2
Implement new feature	2
Create new use cases	1
Implement new manual tests	2
Finish implementation of code	4
Update diagrams	2
Finish the project plan	2
Total time Estimate	19

6. RISK ANALYSIS

6.1 List of risks

Risk	Probability	Effect
Illness	High	serious
Computer breakdown	Low (jinx)	catastrophic
Requirements change	high	tolerable
Underestimate size of project	high	serious

6.2 Strategies

Illness

Due to a medical condition leading to a lowered immune system, the probability of illness is unfortunately high. By being careful of hygiene in public places, washing hands and using rubbing alcohol the chances are at least lowered, but not unavoidable.

Computer breakdown

Always keep backups, push changes to repo.

Requirements change

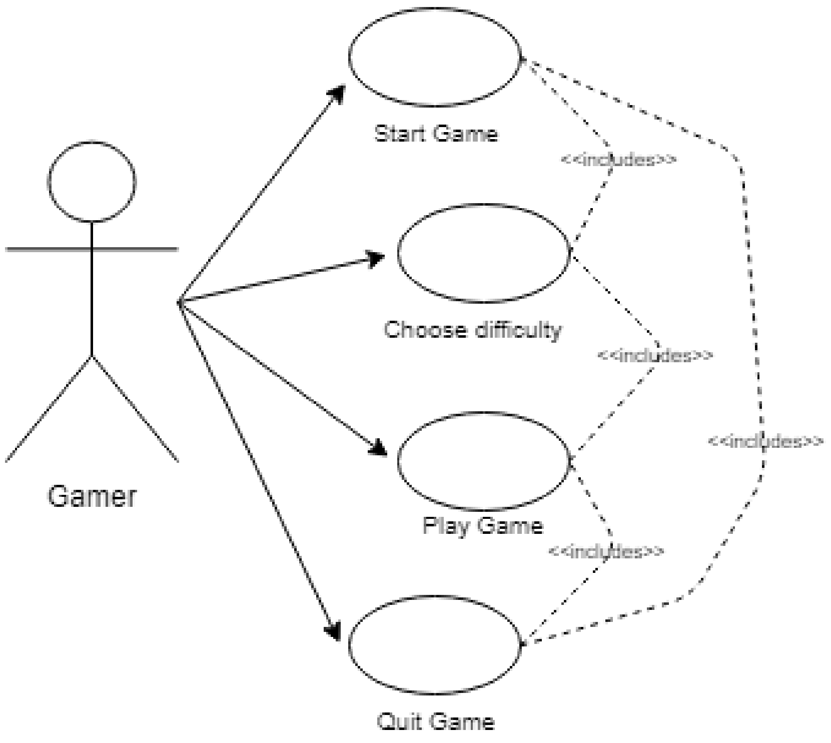
Due to the nature of this project, requirements will be added or removed in future iterations. Should not pose a problem to the deliverance of the project.

Underestimate size of project

Stay on schedule, do what I can to get the required information and stay on targeted delivery date. Remove lesser functions if necessary.

7. MODELING

7.1 Use cases



7.1.1 UC 1 Start Game

Precondition: none.

Postcondition: the game menu is shown.

Main scenario

1. Starts when the user wants to begin a session of the hangman game.
2. The system presents the main menu with a title, the option to play and quit the game.
3. The user makes the choice to start the game.
4. The system asks player to choose difficulty (see Use Case 2)

Alternative scenarios

3.1 The Gamer makes the choice to quit the game.

The system quits the game (see Use Case 4)

7.1.2 UC 2 Choose difficulty

Precondition: The user has chosen to start playing

Postcondition: The game has started

Main scenario:

1. Starts when the player has chosen to start the game
2. The system displays difficulty levels
3. The user selects a difficulty
4. The game starts with the selected difficulty (see Use Case 3)

7.1.3 UC 3 Play game

Precondition: The user has selected a difficulty

Postcondition: the game is finished, asks to play again.

Main scenario

1. Starts when the user has chosen a difficulty level
2. The system has chosen a word and prompts the user for input
3. The user inputs a letter
4. The system checks if the word contains the letter
5. The system displays the result and number of guesses left
6. The user is prompted for input

Repeat from step 3

Alternative scenarios

3.1 The user enters a capital Q to quit the game (see Use Case 4)

6.1 The word is complete and displayed, and the user is prompted to play again or quit.

6.2 All guesses have been made and the user is prompted to play again or quit

7.1.4 UC 4 Quit Game

Precondition: The game is running.

Postcondition: The game is terminated.

Main scenario

1. Starts when the user wants to quit the game.
2. The system prompts for confirmation.
3. The user confirms.
4. The system terminates.

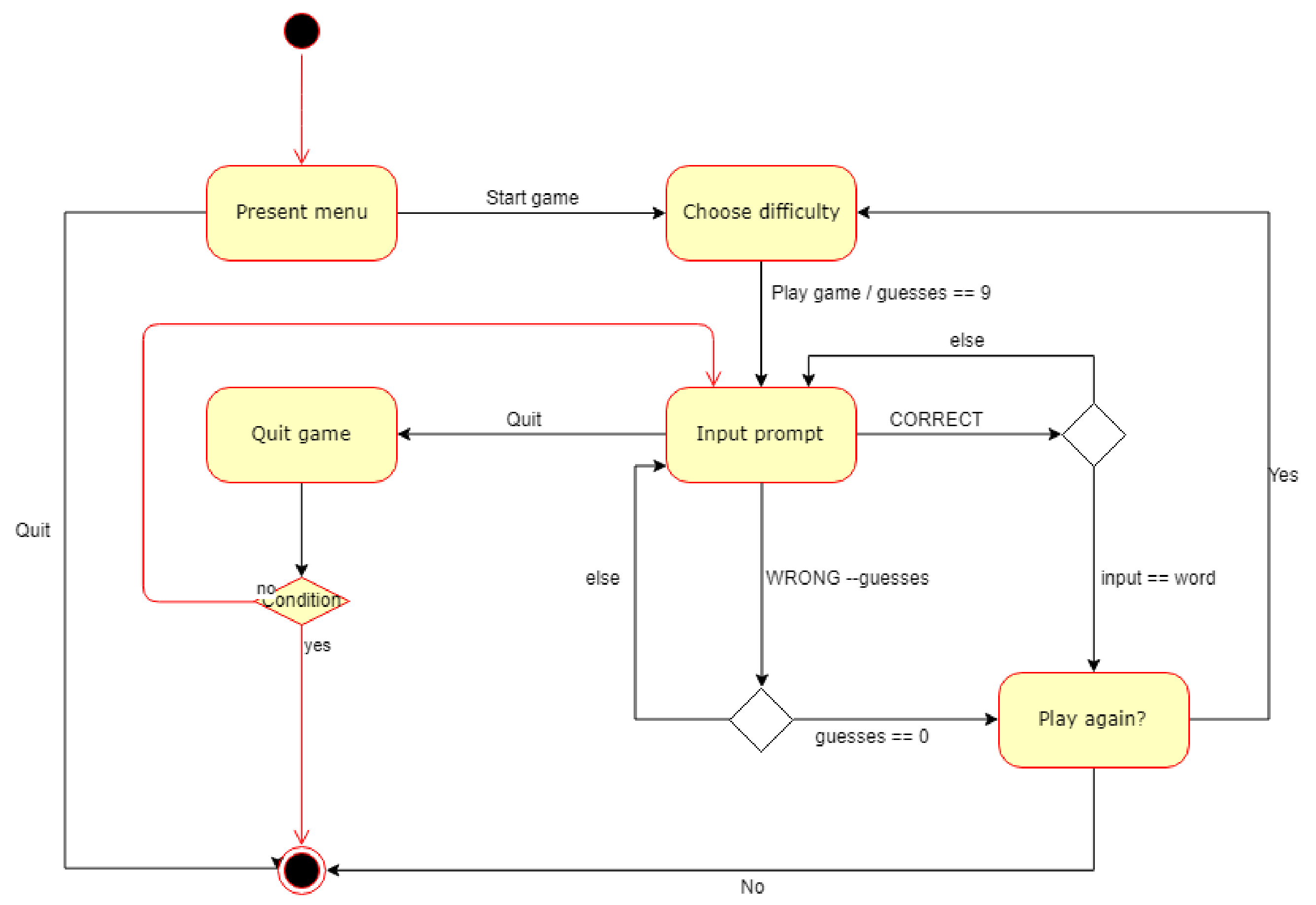
Alternative scenarios

3.1. The user does not confirm

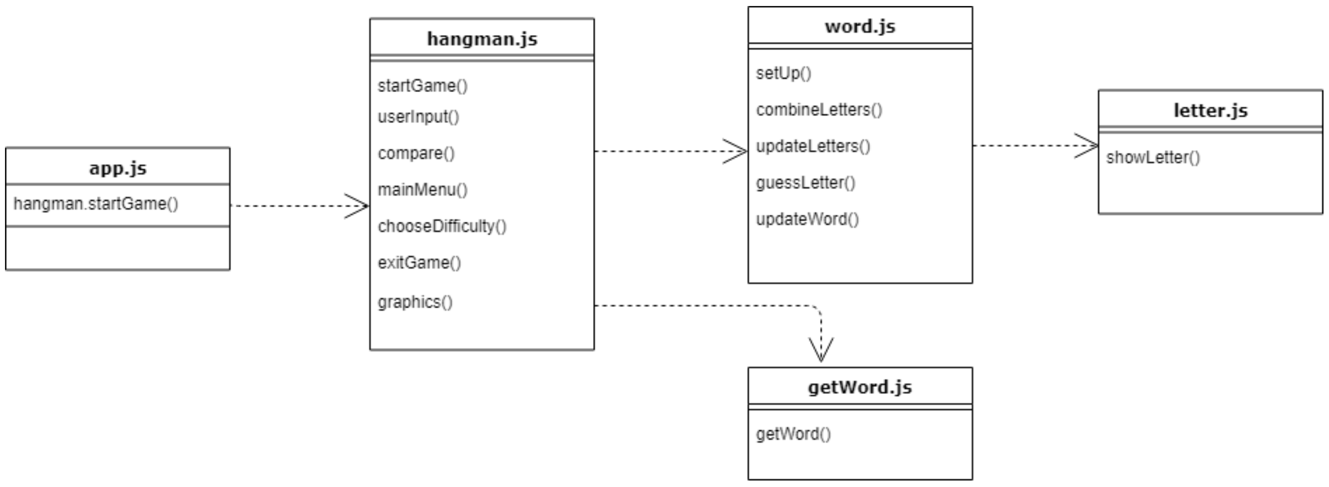
The system returns to its previous state

7.2 Diagrams

7.2.1 State Machine Diagram



7.2.2 Class Diagram



8. TEST PLAN

8.1 Objective

The objective of this test plan is to test the code which was implemented in the last iteration during assignment 2.

8.2 What to Test

Tests will be made for all four use cases. Starting the game, selecting difficulty, playing the game and quit game. Testing will focus on manual test cases because of the use of inquirer which requires user interaction. For automatic testing the framework Chai was used.

8.3 Time Plan

Task	Estimated	Actual
Manual Test Cases	2h	1h
Unit Tests	1h	3h
Running Manual Tests	10m	2m
Inspect Code	30m	25m
Test Report	1h	10m

8.4 Manual Test-Cases

8.4.1 TC1.1 From UC1 Start Game

Use case: UC1 Start Game

Scenario: The player successfully starts the game.

The main scenario of use case 1 is tested where a player successfully starts a game of hangman from a presented menu.

Pre-condition: none

Test steps:

- Start the application
- System displays “Want to play a game? (Use arrow keys) followed by answers “Yes” and “No” on separate lines
- Press Enter to select “Yes” which is the default option

Expected:

- System continues to Use Case 2

8.4.2 TC2.1 from UC2 Choose difficulty

Use Case: UC2 Choose difficulty

Scenario: The player selects a difficulty and starts a round of the game

Precondition: The player has successfully started the game from the main menu

Test steps:

- System displays “Choose a difficulty: (Use arrow keys)” followed by answers “Easy” and “Hard”
- Press enter to select “Easy” which is the default option

Expected:

- System continues to Use Case 3

8.4.3 TC3.1 From UC3 Play Game

Use Case: UC3 Play Game

Scenario: The player completes a round of the game

The main scenario of use case 3 is tested where a complete round of the game is played and the correct word is known. For the purpose of this test the words able to be chosen by the game has been limited to the word "test".

Pre-condition: Use case 1 and 2 has been followed through, i.e. the player has started the game and selected a difficulty.

Test steps:

- System displays "guess a letter" and awaits input
- Press "t" and then press enter
- System displays "t _ _ t" and "remaining guesses: 9" on a new line
- Press "e" and then press enter
- System displays "t e _ t" and "remaining guesses: 9" on a new line
- Press "s" and then press enter

Expected:

- System displays "t e s t" and "remaining guesses: 9" on a new line, as well as "you win!" on a new line
- System displays main menu
- Press down arrow key to select "No" and press enter
- System displays "Thank you, come again"

Comments:

8.4.4 TC4.1 From UC4 Quit Game

Use Case: UC4 Quit game

Scenario: The main scenario is when the user chooses to quit the application from inside UC3 Play game.

Pre-condition: Use case 1 and 2 has been followed through, i.e. the player has started the game and selected a difficulty.

Test steps:

- System displays “guess a letter” and awaits input
- The user inputs a capital “Q” and press Enter
- The system displays “Sure you want to exit? (Use arrow keys)”
- The user press Enter

Expected:

- The game has quit and system displays “Thank you, come again”

8.5 Test Report

8.5.1 Manual tests

Test	UC1	UC2	UC3	UC4
TC1.1	1/OK	0	0	0
TC2.1	0	1/OK	0	0
TC3.1	0	0	1/OK	0
TC4.1	0	0	0	1/OK
COVERAGE& SUCCESS	1/OK	1/OK	1/OK	1/OK

8.5.2 Automatic tests

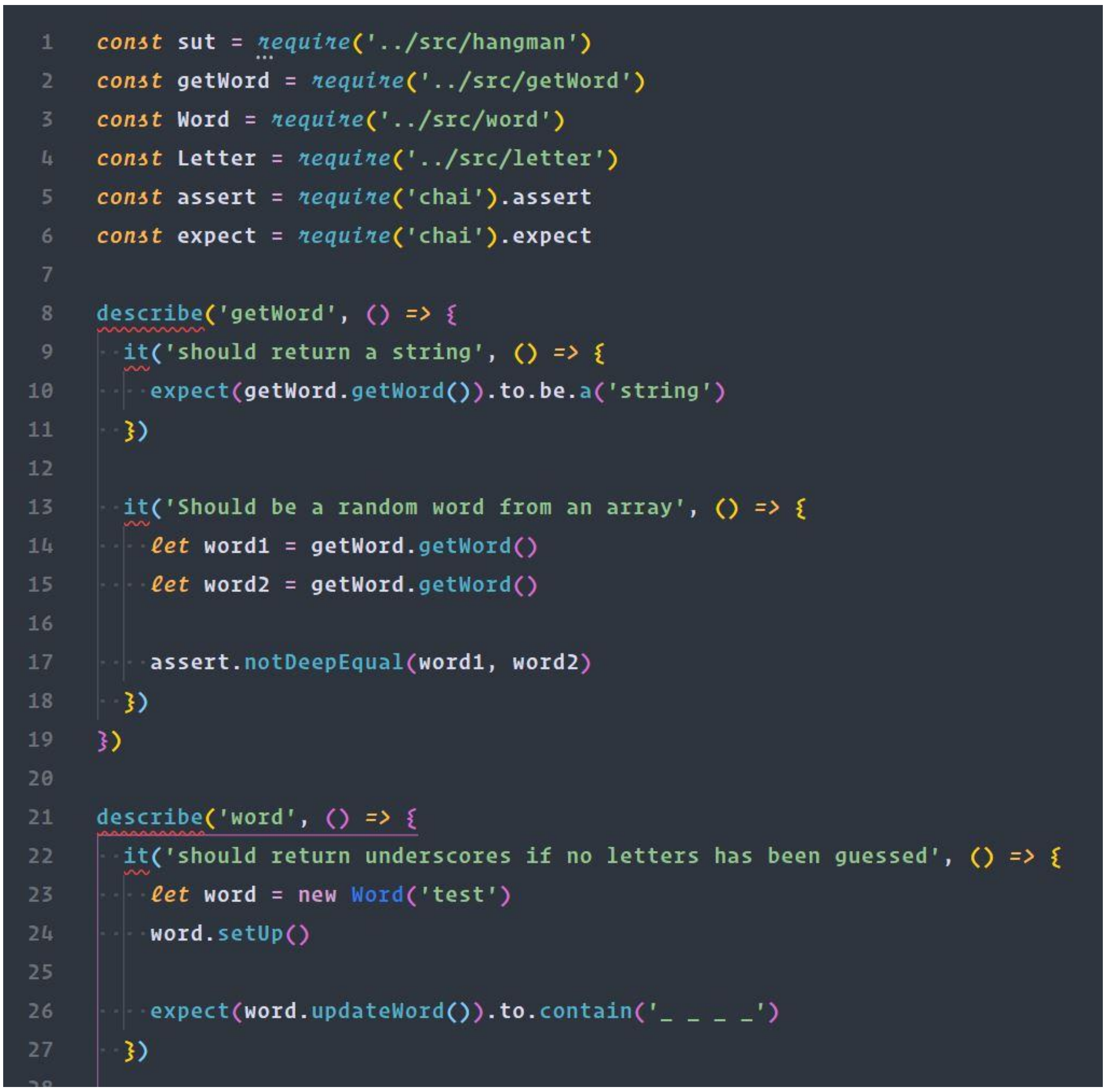
Test	hangman	getWord	Word	letter
COVERAGE&SUCCESS	0	100%/OK	100%/OK	0

Comments:

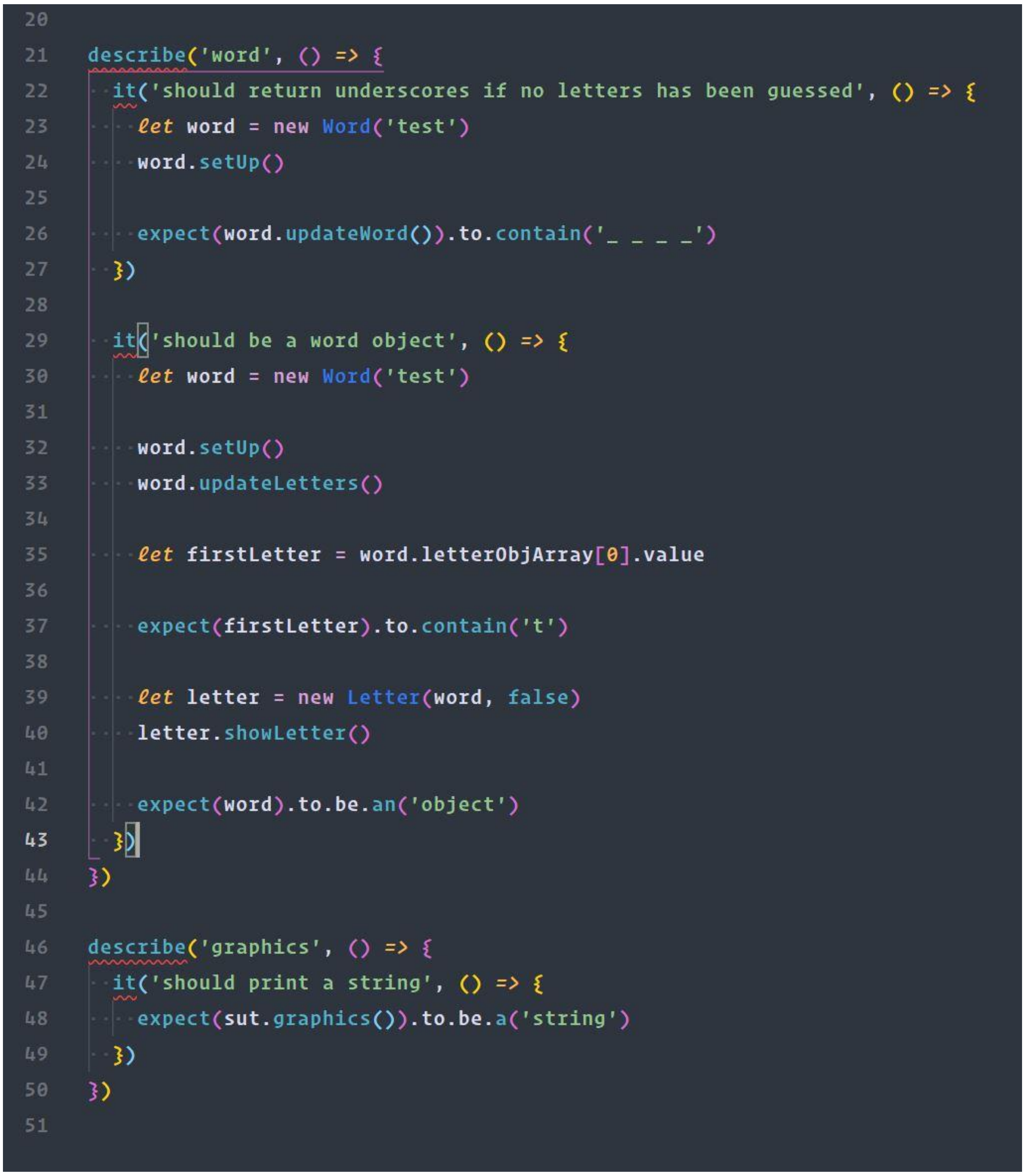
All manual tests function as specified, the automated tests for “getWord” and “Word” functions as specified.

8.6 Automated Unit Tests

Test Code:

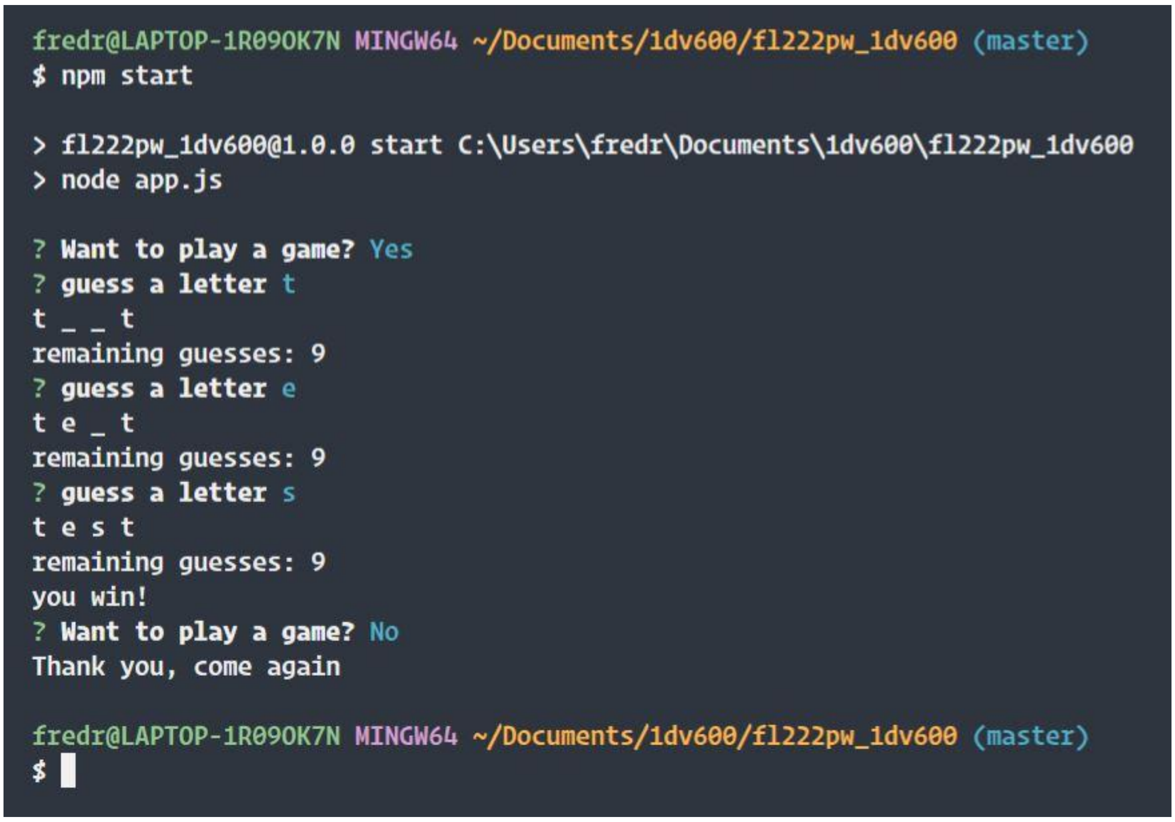


```javascript
const sut = require('../src/hangman')
const getWord = require('../src/getWord')
const Word = require('../src/word')
const Letter = require('../src/letter')
const assert = require('chai').assert
const expect = require('chai').expect

describe('getWord', () => {
  it('should return a string', () => {
    expect(getWord.getWord()).to.be.a('string')
  })

  it('Should be a random word from an array', () => {
    let word1 = getWord.getWord()
    let word2 = getWord.getWord()

    assert.notDeepEqual(word1, word2)
  })
})

describe('word', () => {
  it('should return underscores if no letters has been guessed', () => {
    let word = new Word('test')
    word.setUp()

    expect(word.updateWord()).to.contain('_ _ _ _')
  })
```



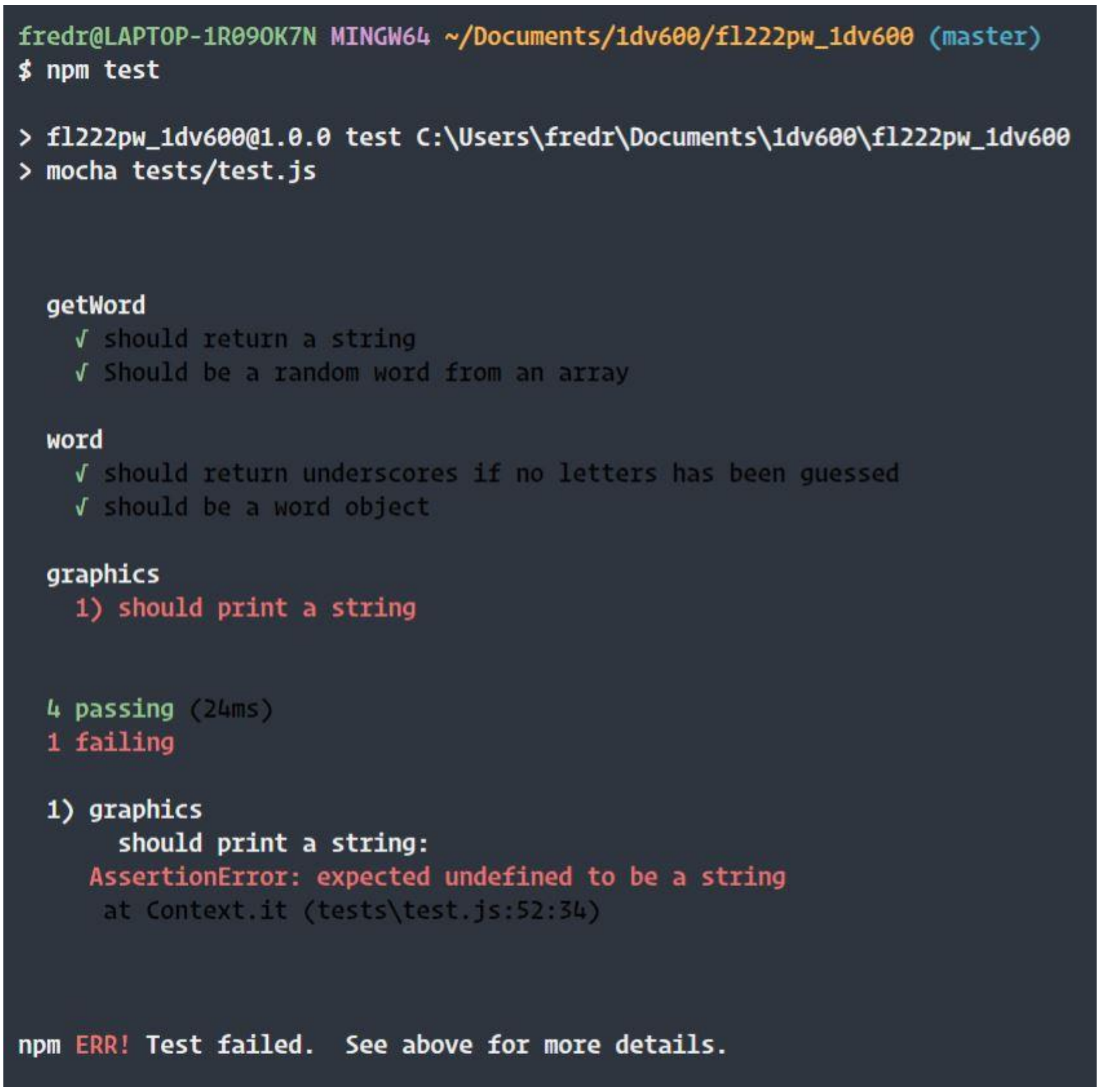
```

describe('word', () => {
  it('should return underscores if no letters has been guessed', () => {
    let word = new Word('test')
    word.setUp()

    expect(word.updateWord()).to.contain('_ _ _ _')
  })

  it('should be a word object', () => {
    let word = new Word('test')

    word.setUp()
    word.updateLetters()

    let firstLetter = word.letterObjArray[0].value

    expect(firstLetter).to.contain('t')

    let letter = new Letter(word, false)
    letter.showLetter()

    expect(word).to.be.an('object')
  })
})

describe('graphics', () => {
  it('should print a string', () => {
    expect(sut.graphics()).to.be.a('string')
  })
})

```

Manual Test Results:



```
fredr@LAPTOP-1R090K7N MINGW64 ~/Documents/1dv600/fl222pw_1dv600 (master)
$ npm start

> fl222pw_1dv600@1.0.0 start C:\Users\fredr\Documents\1dv600\fl222pw_1dv600
> node app.js

? Want to play a game? Yes
? guess a letter t
t _ _ t
remaining guesses: 9
? guess a letter e
t e _ t
remaining guesses: 9
? guess a letter s
t e s t
remaining guesses: 9
you win!
? Want to play a game? No
Thank you, come again

fredr@LAPTOP-1R090K7N MINGW64 ~/Documents/1dv600/fl222pw_1dv600 (master)
$
```

Automatic Test Results:



```
fredr@LAPTOP-1R090K7N MINGW64 ~/Documents/1dv600/fl222pw_1dv600 (master)
$ npm test

> fl222pw_1dv600@1.0.0 test C:\Users\fredr\Documents\1dv600\fl222pw_1dv600
> mocha tests/test.js



  getWord
    √ should return a string
    √ Should be a random word from an array

  word
    √ should return underscores if no letters has been guessed
    √ should be a word object

  graphics
    1) should print a string


  4 passing (24ms)
  1 failing

  1) graphics
       should print a string:
     AssertionError: expected undefined to be a string
      at Context.it (tests\test.js:52:34)



npm ERR! Test failed.  See above for more details.
```

Reflections

Writing unit tests in JavaScript I thought was very difficult, especially since I’m using the Inquirer module to use a menu controlled by arrow keys. This meant that it was hard to run a “system under test” since anytime I instantiated a new hangman game from unit tests, the menu would load forcing me to press down arrow key to select “no” and quit the game. I did not want to modify my code just to implement the tests.

This showcases that I did not have testing in mind as I wrote the source code, which has left me in a tough situation, as I right now don’t have time to create new tests as the deadline for attempt 2 iteration 4 is approaching (within hours)

9. TIME LOG

9.1 Iteration 1.

Activity	Est. time	Date	Actual time
Reading of chapter 2, 3	4 hrs	190126	5 hrs
Reading of chapter 22, 23	6 hrs	190129	7 hrs
Viewing of online lectures	4 hrs	190125	7 hrs
Course lectures on campus	4 hrs	190123	4 hrs
Creation of repository on GitHub	1 hrs	190129	0.5 hrs
Writing of Project document	19 hrs	190208	24 hrs
Total Effective Time			47.5 hrs

9.2 Iteration 2

Activity	Est. Time	Date	Actual time
Reading of Chapter 4,5,6,7,15 and 20	20 hrs	19-02-12/13/15/16/19	16 hrs
Viewing of online lectures	4 hrs	19-02-13/15	7 hrs
Course lectures on campus	8 hrs	19-02-06/13/20	6hrs
Design features to hangman	2 hrs	19-02-19/20	1 hr
Implement features to hangman	10 hrs	19-02-19/20/21/22	9 hrs
Developing project plan	4 hrs	19-02-19	1 hr
Working on diagrams	4 hrs	19-02-19/20/21	4 hrs
Studying for test	3 hrs	19-02-21	3 hrs
total	55 hrs		47 hrs

9.3 Iteration 3

Activity	Est. Time	Date	Actual time
Reading of course literature chapter 8	2 hrs	19-03-02	3 hrs
Viewing of online lectures	6 hrs	19-03-02/19-03-22	5 hrs
Course lectures on campus	8 hrs	19-02-27/19-03-03	4 hrs
Add additional features	10 hrs	19-02-18/19-02-22	3 hr
Create test	4 hrs	19-04-05	6 hrs
Plan, perform and document tests	15 hrs	19-04-05/19-04-18	12 hrs
total	45 hrs		33 hrs

9.4Iteration 4

Activity	Est. Time	Date	Actual time
Course lectures on campus	4 hrs	19-03-13/19-03-20	2 hrs
Combine all previous documents	2 hrs	19-04-18	3 hrs
Implement new feature	2 hrs	19-02-27/19-03-03	3 hrs
Create new use cases	1 hrs	19-02-18/19-02-22	1 hrs
Implement new manual tests	2 hrs	19-04-18	2 hrs
Finish implementation of code	4 hrs	19-04-05/19-04-18	5 hrs
Update diagrams	2 hrs	19-04-18	2 hrs
Finish the project plan	2 hrs	19-04-18	2 hrs
total	19 hrs		20 hrs