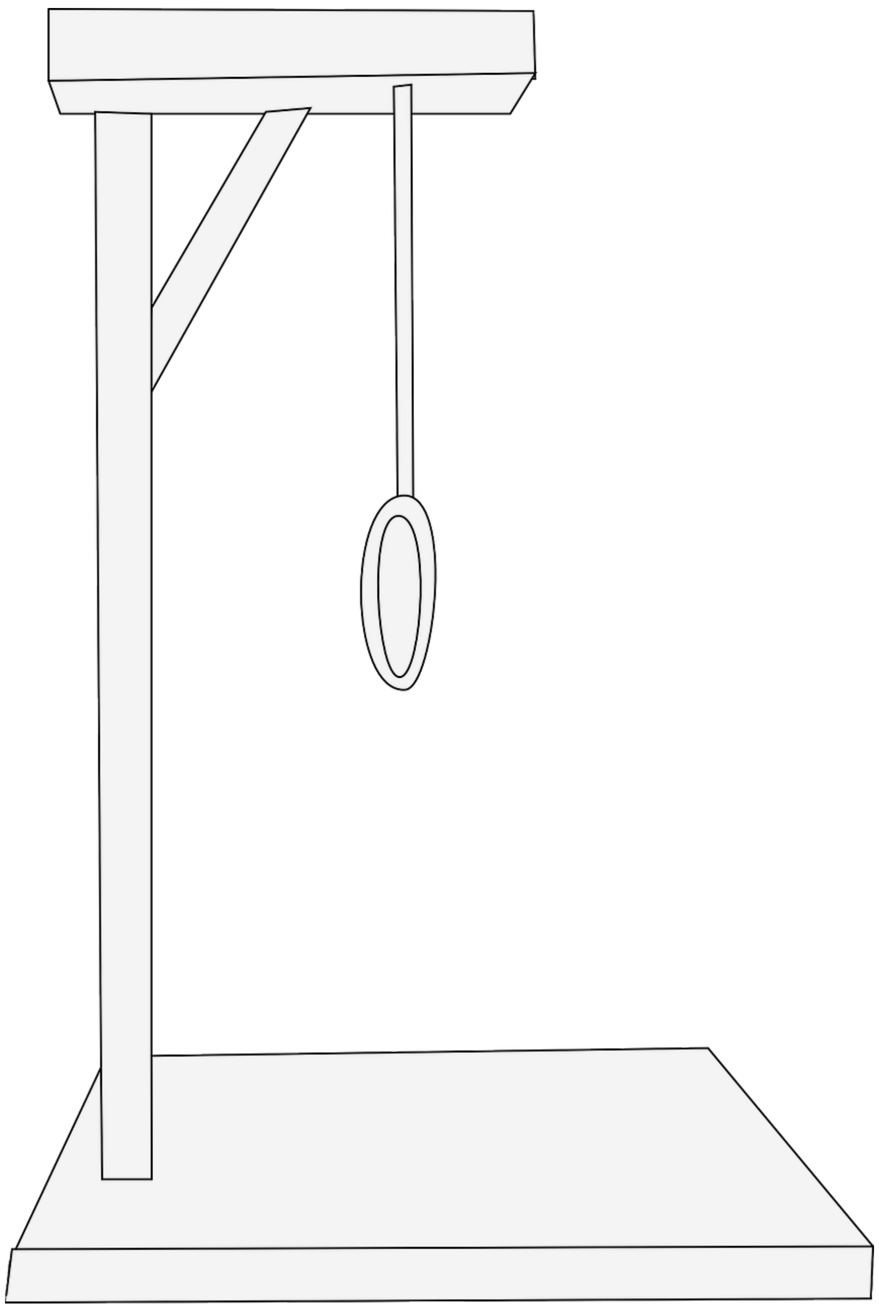


Terminal Hangman Project Plan

Viktor Ödman

Linnéuniversitet

1 februari 2020

GitLab: https://gitlab.lnu.se/vo222dq

1 | GENERAL INFORMATION

PROJECT SUMMARY	
Project Name	Terminal Hangman
Project Manager **Project Tester** **Project Developer**	Viktor Ödman
Main Client	Linnéuniversitetet
Executive Summary	The product is going to be a single player text-based console game. The game is going to be based on the traditional pencil and paper game ”Hangman” The application will be developed by only one developer. The project is going to be spilt up to 4 iterations.

2 | REVISION HISTORY

DATE	VERSION	DESCRIPTION	AUTHOR
01/02/2020 - 18:00	v1.0	Added Vision and Vision reflection	Viktor Ödman
02/02/2020 - 19:05	v1.0	Made Changes in the Vision Document	Viktor Ödman
02/02/2020 - 19:11	v1.0	Updated Vision reflection	Viktor Ödman
02/02/2020 - 19:30	v1.0	Added The project plans introduction.	Viktor Ödman
02/02/2020 - 21:32	v1.0	Added Justification, Stakeholders and Resources	Viktor Ödman
03/02/2020 - 11:00	v1.0	Added Hard and Software Requirements. Also added Overall project schedule.	Viktor Ödman
03/02/2020 - 14:40	v1.0	Added Scope, Constraints and Assumptions	Viktor Ödman
03/02/2020 - 17:20	v1.0	Added Iterations and Risk Analysis	Viktor Ödman
03/02/2020 - 17:45	v1.0	Added Time Log and a reflection to the project plan.	Viktor Ödman
15/02/2020 - 12:15	v1.1	Updated the Vision document	Viktor Ödman
15/02/2020 - 13:30	v1.1	Updated The Stakeholders table	Viktor Ödman
15/02/2020 - 14:00	v1.1	Updated The Time Log	Viktor Ödman
15/02/2020 - 15:30	v1.1	Added a reflection on the Risk Analysis	Viktor Ödman
16/02/2020 - 19:15	v1.1	Added two new risks in the Risk Analysis	Viktor Ödman
24/02/2020 - 19:00	v2.0	Updated Iteration 2 Time estimation and actual time. Also updated Time Log	Viktor Ödman
09/03/2020 - 20:00	V3.0	Updated Iteration 3 Time estimation and actual time. Also updated Time Log	Viktor Ödman
20/03/2020 - 15:00	v4.0	Adding Information about Iteration 4 in Iteration section.	Viktor Ödman
20/03/2020 - 16:00	v4.0	Adding Use Cases section to the document	Viktor Ödman
20/03/2020 - 17:00	v4.0	Adding Diagrams section to the document	Viktor Ödman
20/03/2020 - 18:00	v4.0	Adding Manual and Automated test cases to the document.	Viktor Ödman
20/03/2020 - 19:00	v4.0	Updated Iteration 4 Time estimation and actual time. Also updated Time Log	Viktor Ödman

3 | VISION

The vision for this project is to be able to plan, design and develop a simple console application based on the traditional paper and pencil game ”Hangman”.

The game will differ from the traditional game in some ways. Traditionally the game is played by several players, but this game is going to be a single player game. The word is not going to be chosen by the player, but will be generated from a list of words.

Even though the application is limited to the console, and is not going to have a graphical user interface, the application will be easy to understand and easy to use. By using a text-based interface in the console, the user will have an easy time navigating though the start menu and the game.

The main goal of the project is to create a game that can be enjoyed by anyone. Experienced or non experienced hangman players will have a good time playing this game.

3.1 | Vision reflection

When creating the vision I tried to present a simple overview of what is going to be developed without going into too much detail. In the vision I was describing what the final product should be, how it should work, how the user should use the product, what the user will see, who is going to develop the project, why I'm developing the project and what hope to get out of this project.
The difficult part of writing the "Vision" was to know what details that should be included and what should be left out.

4 | PROJECT PLAN

4.1 | Introduction

The application will be a text-based console game that is based on the old pencil and paper game ”Hangman” https://en.wikipedia.org/wiki/Hangman_(game). Traditionally the game is played by at least two players, where one player comes up with a word and the other player tries to guess the word letter by letter.

In this game the objective is going to be too guess a word based on a number of underscores, each underscore representing a letter and the letters combined will represent a word. This application will be a single player game, where the words are randomly picked from a list.

There will be at least two wordlists, each following a specific theme. The first list will be a list with all the countries in Europe. The second list will be a list of car brands.

There will be a menu when starting the game where the user should be able chose what list the word should be generated from and then start the game. The game is going to randomly pick a word from the chosen list.

The players task will be to figure out what word that the underscores represent by guessing a letter. After each guess, if the word consist of the guessed letter, the underscore/underscores that represented that letter will be changed to that letter. If the guessed letter is not included in the word, a part of the building of a man getting hanged begins. Each part of the man represents the number of wrong guesses allowed to guess the word.

In the final iteration of the application we be adding new features including the possibility to change difficulty and to change to a new game mode.

The different difficulties will change how many body parts that the game will add to the man getting hanged when the player makes a wrong guess.

The new game mode will hide the underscores that represents the letters in the word, to make the game more of a challenge.

4.2 | Justification

The reason for creating this application is to get experience on how to develop an application from the perspective of a project. To learn how to plan, structure and design the development of an application.

4.3 | Stakeholders

Stakeholder	Intrest
Developer	The Developer wants maintainable code with good structure and documentation.
Project Manager	The Project Manager wants the project to be finished on time and that the project delivers what was planned.
End-User	The End-User (The person that is going to use the application) wants a stable application that is easy and fun to use.
Project Tester	The Project Tester wants to be able to test the application so that the application delivers what was planned.

4.4 | Resources

The available resources for this project is:

- About 20 hours a week in case of time available for this project.
- Knowledge gained from this course.
- Software Engineering - TENTH EDITION by Ian Sommerville (Book)
- The developers computer.
- A keyboard and mouse.
- The latest version of Node.js.
- Visual studio code.
- A bash terminal.

4.5 | Hard and Software Requirements

Hard and Software requirements for running this application:

- The latest version of node
- Any pc that can run node applications.
- A keyboard
- A terminal/console

4.6 | Overall Project Schedule

The schedule for this project will be based on the different deadlines for this course. The deadlines for this course is split in to 4 iterations spanning over 9 weeks. The different iterations are:

Iterations	Weeks	Theme
Iteration 1	4-5	Process and Planning
Iteration 2	6-8	Modeling and Software Design
Iteration 3	9-10	Software Testing
Iteration 4	11-12	Final iteration

4.7 | Scope, Constraints and Assumptions

This application will be a single player game based on the traditional pencil and paper game hangman. The game will only be played in a console/terminal environment, and will not have any graphical user interface. Here are some Requirements, Constraints and Assumptions.

4.7.1 | Requirements

Functional requirements

- When starting the application the user shall be greeted with a start menu.
- ~~One menu item shall lead to a description on how to play the game.~~
- The words shall be randomly picked from at least two lists.
- Every wordlist shall follow a theme.
- The user shall be able to choose what list the word should be picked from.
- The letters of the word shall be displayed as underscores.
- The maximum number of wrong guesses before the player loses the game shall be 8.
- The parts of the man getting hanged shall be represented by the maximum of wrong guesses.
- When the guessed letter is correct, the underscores that represent that letter shall be changed to that letter.

- When the guessed letter is incorrect, a new part to man getting hanged shall be added.
- If the user is able to guess the full word, the game shall present that the user has won.
- If all the parts of the man getting hanged is added, the game shall present that the user has lost.
- The user should be able to play only using the keyboard.

Non-functional requirements

- The code should follow the code and JSDoc conventions of https://www.npmjs.com/package/@lnu/eslint-config.
- There should a describing README file on how to install the application and how to start the application.
- While being in the game directory, the user should be able to install the game and it's dependencies by running "npm install" in the console/terminal.
- While being in the game directory, the user should be able to start the game by running "npm start" in the console/terminal.

4.7.2 | Constraints

- The game will be a text-based console application and will not have any graphical user interface.
- The application will be written in javascript using the node.js platform.
- The time for developing this application will be limited to the timeframe of this course.
- Limited knowledge and experience in project planning.

4.7.3 | Assumptions

- That the project staff working on this project will be able to allocate at least 20 hours a week for this project.
- That node and the npm dependencies that will be used will still work when the assignment is done.
- That the limited knowledge that the project staff have in project planning will be enough to finish this project.

4.8 | Project Plan Reflection

In the project plan I tried to paint a complete picture of what the actual application should do when the project is done. I also tried to describe how the application should work, who is involved and why the application should be developed. I think that this part of the project documentation will really help me to define the application and make it easier to design and develop it. I not sure if was supposed to add functional and non-functional requirements, but I thought it was fitting when describing the scope of the application.

5 | ITERATIONS

This part will be going through the different iterations of this project. There are some time estimations of what is going to be accomplished in the different iterations.

5.1 | Iteration 1

In this iteration we are going to create a project plan for the Terminal Hangman project and also add some skeleton code for the application.

Week 4-5 Process and Planning

Tasks	Time Estimation	Actual Time	Description
Reading Chapters 2-3 and 22-23	10 hours	12 hours	The chapters includes information about Software processes, Agile software development, Project management and project planning
Watching pre-recorded lectures	3 hours and 23 minutes	3 hours and 23 minutes	The course provided lectures
Create The Vision Document	3 hours	5 hours	Creating an overall vision for the project
Create The Project Plan	3 hours	5 hours	Creating an Project Plan Document for the project
Planning For The Iterations	2 hours	3 hours	Planning for the different iterations of the project and make some time estimations
Identifying Risks And Create Risk Strategies	2 hours	2 hours	Identifying some risks and create appropriate risk strategies.
Add skeleton code	1 hour	1 hour	Add basic skeleton template

5.2 | Iteration 2

In this iteration we are going to implement the basic menu flow for the application. We are also going to create a fully dressed Use Case for playing the game and create a State Machine for playing the game.

Week 6-8 Modeling and Software Design

Tasks	Time Estimation	Actual Time	Description
Reading Chapters 4, 5, 20 and 7.1	15 hours	12 hours	The chapters includes information about Requirements engineering, System modeling, System of systems and Object oriented design using the UML
Watching pre-recorded lectures	10 hours	10 hours	The course provided lectures
Update the project plan	2 hours	3 hours	Change the project plan with the new information gathered in this iteration.
Making time estimations	2 hours	2 hours	Make time estimations on the different tasks.
Expand the use case model	2 hours	1 hour	Add a new use case to the current Use Case Diagram
Creating a fully dressed use case for Play Game	2 hours	3 hours	Create a fully dressed use case for play game.
Creating a Play Game State Machine	3 hours	6 hours	Create a State Machine for the play game use case, that describes the transitions between the different states of the use case.
Implementation	4 hours	6 hours	Create a basic menu flow, so that the player can navigate in the game menu according to the state machine
Modeling Structure	2 hours	1 hour	Create a class diagrams from the implementation

5.3 | Iteration 3

In this iteration we are going continue the implementation of the game and also create tests for the application. We are going to create manual test cases for Use Case 3 (Quit Game) and Use Case 2 (Play Game). We will also create automated unit tests for some methods in the application (The methods that are going to be tested will be described in the Test section of this document).

Week 9-10 Software Testing

Tasks	Time Estimation	Actual Time	Description
Reading Chapter 8	4 hours	4 hours	The chapter includes information about Software Testing
Watching pre-recorded lectures	4 hours	4 hours	The course provided lectures
Update the project plan	1 hour	1 hour	Change the project plan with the new information gathered in this iteration.
Writing Manual Test Cases	2 hours	3 hours	Writing manual test cases for UC2 and UC3
Running the manual tests	30 min	10 min	Running the test cases TC 3.1 and TC 2.1
Writing Unit tests	4 hours	6 hours	Writing 5 automatic unit tests
Writing a reflection	1 hour	1 hour	Writing a reflection for this iteration

5.4 | Iteration 4

In this iteration we are going to finalize the application and also add some new features. The new features will be a new game mode called ”Invisible” and the possibility to change difficulty to ”Easy”, ”Normal” and ”Hard”.

The game mode ”Invisible” will hide the underscores so that the player can’t see how many letters there are in the word.

The different difficulties will change the number of wrong guesses that the player can make before losing the game. ”Easy” will be the default difficulty and will allow the user to make 8 wrong guesses before losing the game(”Medium” will allow 4 wrong guesses and ”Hard” will allow 2 wrong guesses).

We are also going to implement a settings menu to allow the player to change the difficulty and game mode.

We are going to create a fully dressed Use Case for changing the game mode and for changing the difficulty.

Week 11-12 Final Iteration

Tasks	Time Estimation	Actual Time	Description
Create Use Cases	1 hour	2 hours	Creating fully dressed Use Cases for UC4 and UC5.
Implement new features	2 hours	3 hours	Implementing the settings menu, the possibility to change difficulty and the possibility to change game mode.
Create Test Cases	1 hour	1 hour	Creating Test Cases for UC4 and UC5
Run Test Cases	10 minutes	5 minutes	Running the test cases created in this iteration
Update the project plan	4 hours	Unknown	Change the project plan with the new information gathered in this iteration.

6 | RISK ANALYSIS

The risk analysis will cover some of the risks that might occur during this project and some strategies to avoid or minimize the impact of the risks.

6.1 | List of risks

Risk	Affects	Description	Effects	Probability
Not meeting deadlines	Project / Product	Not being able to hand in the assignments before the deadline.	Serious	Moderate
Tasks taking up more time then estimated	Project / Product	Self describing	Tolerable	High
Computer Malfunction	Project / Product	The developers computer is malfunctioning and he can't get access to his project files.	Tolerable	Low
Staff illness	Project / Product	The developer is too ill to able to work on the project during the course.	Serious	Low
Feature Creeping	Project / Product	More Features are promised but not delivered. And the fundamental features are not delivered	Serious	Moderate
The used npm dependencies not working as intended	Project / Product	Self describing	Tolerable	Low

6.2 | Strategies

Risk	Strategy
Not meeting deadlines	Avoidance strategy: Put it the time that is needed to meet deadlines. Contingency plan: Hand in the assignment on a later occasion.
Tasks taking up more time then estimated	Minimization strategy: Allocating more time from other Tasks.
Computer Malfunction	Minimization strategy: Doing frequent commits to GitHub so that the files are available elsewhere.
Staff illness	Contingency plan: Hand in the assignment on a later occasion.
Feature Creeping	Avoidance strategy: Focusing on the products fundamentals features first
The used npm dependencies not working as intended	Avoidance strategy: Not using to many npm dependencies Minimization strategy: Finding other similar dependencies that can be tested if the first didn't work as intended.

6.3 | Risk Analysis Reflection

When creating the list of risks I tried to focus on what I think is the most basic risks that can occur. I tried stay away from risks like ”Natural Disasters” or ”Getting into a traffic accident” and focus more on the regular everyday risks that could happen to me as a student. I think the idea of defining risks before they occur and develop a plan for those risks seems very useful. I'm not sure on where to draw the line on how many and what risks that should be included in the list, but I think I got the most essential risks down.

8 | USE CASES

In this section we are going to create some fully dressed Use Cases to get an overview to what our application Is supposed to do.

8.1 | UC 2 Play Game

Precondition: The player want to play the game.

Postcondition: The player chooses to stop playing

Main scenario

1. Starts when the Player has chosen a word theme.
2. The system presents placeholders for the letters of the word.
3. The Player enters a letter.
4. The system replaces the placeholders to the letter(Goto step 3)
5. The Player enters all the letters of the word without exceeding the number of allowed guesses.
6. The system presents the Game win confirmation and two alternatives, Play Again and Quit Game.
7. The Player chooses Quit Game.

Alternative scenarios

3.1 The Player chooses to stop playing

1. The System asks for the Player confirmation.
2. The Player confirms and exits the game

3.1.2 The player chooses to continue playing.

1. Goto 3

4.1 The Player enters a letter not included in the word.

1. The system adds a part to the man getting hanged.
2. Goto 3.

4.2 The Player enters a letter that's already been entered.

1. The System informs the player.

2. Goto 3

5.1 The player exceeds the number of allowed guesses.

1. The system presents the Game lose confirmation and two alternatives, Play Again and Quit Game.
2. The player chooses Play Again(See 7.1)

7.1 The Player chooses Play Again

1. Goto 1

8.2 | UC4 Change Game Mode

Precondition: The player wants to change the game mode.

Postcondition: The player has chosen a game mode.

Main scenario

1. Starts when the Player has chosen the menu item ”Change Settings” in the main menu
2. The system prompts a list of menu items (”Change Game Mode”, ”Change Difficulty” and ”Go Back”)
3. The Player makes the menu choice ”Change Game Mode”
4. The system prompts a list of menu items (”Normal” and ”Invisible”)
5. The Player makes the menu choice ”Invisible”
6. The system saves the menu choice

Alternative scenarios

5.1 The player makes the menu choice ”Normal”

1. Goto 6

8.3 | UC5 Change Difficulty

Precondition: The player wants to change the difficulty.

Postcondition: The player has chosen a difficulty.

Main scenario

1. Starts when the Player has chosen the menu item ”Change Settings” in the main menu
2. The system prompts a list of menu items (”Change Game Mode”, ”Change Difficulty” and ”Go Back”)
3. The Player makes the menu choice ”Change Difficulty”
4. The system prompts a list of menu items (”Easy”, ”Normal” and ”Hard”)
5. The Player makes the menu choice ”Hard”
6. The system saves the menu choice

Alternative scenarios

5.1 The Player makes the menu choice ”Normal”

1. Goto 6

5.2 The Player makes the menu choice ”Easy”

1. Goto 6

9 | DIAGRAMS

9.1 | Use Case Diagram

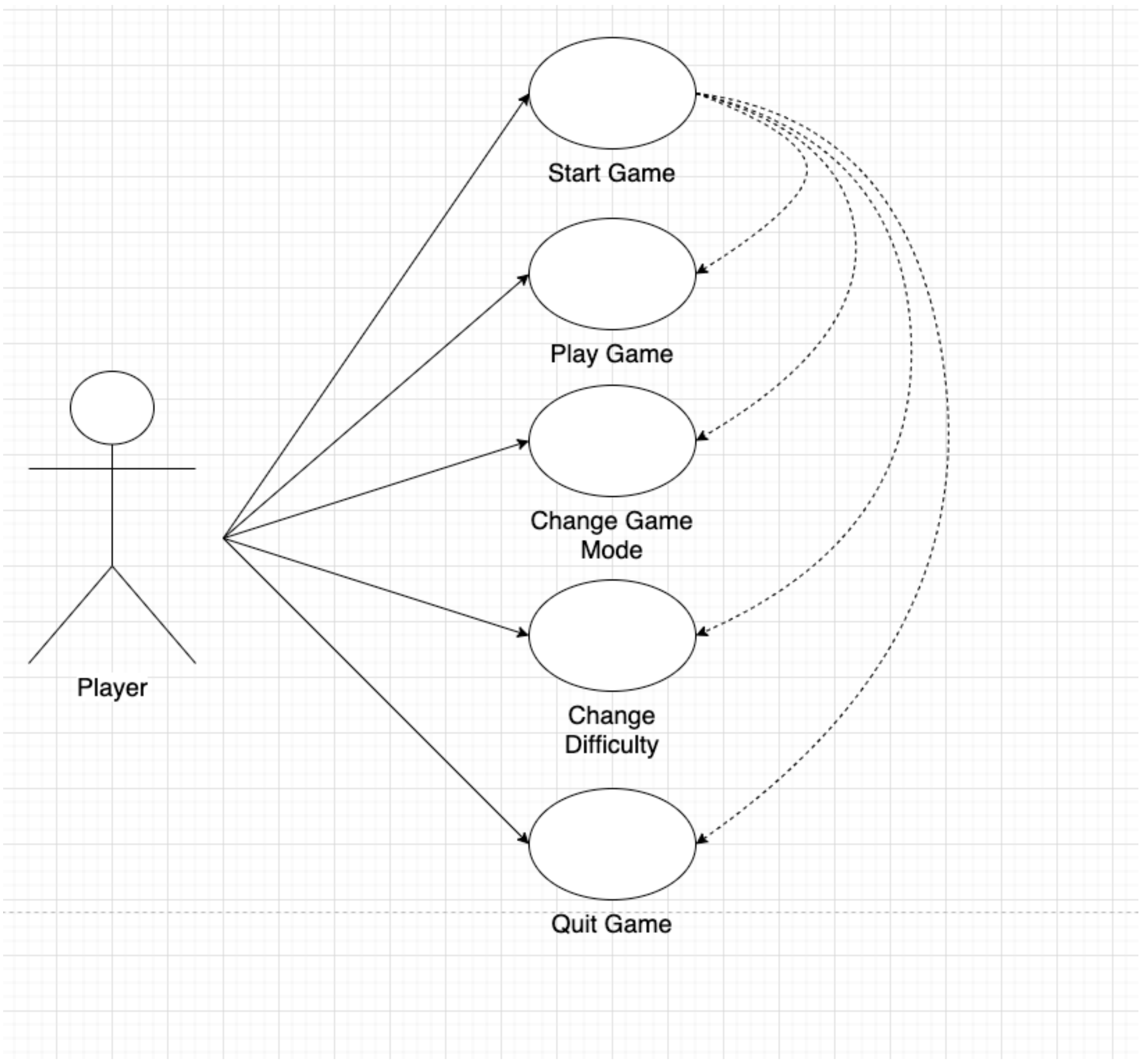


Fig 1 Use Case Diagram

9.2 | State Machine Diagram For Play Game

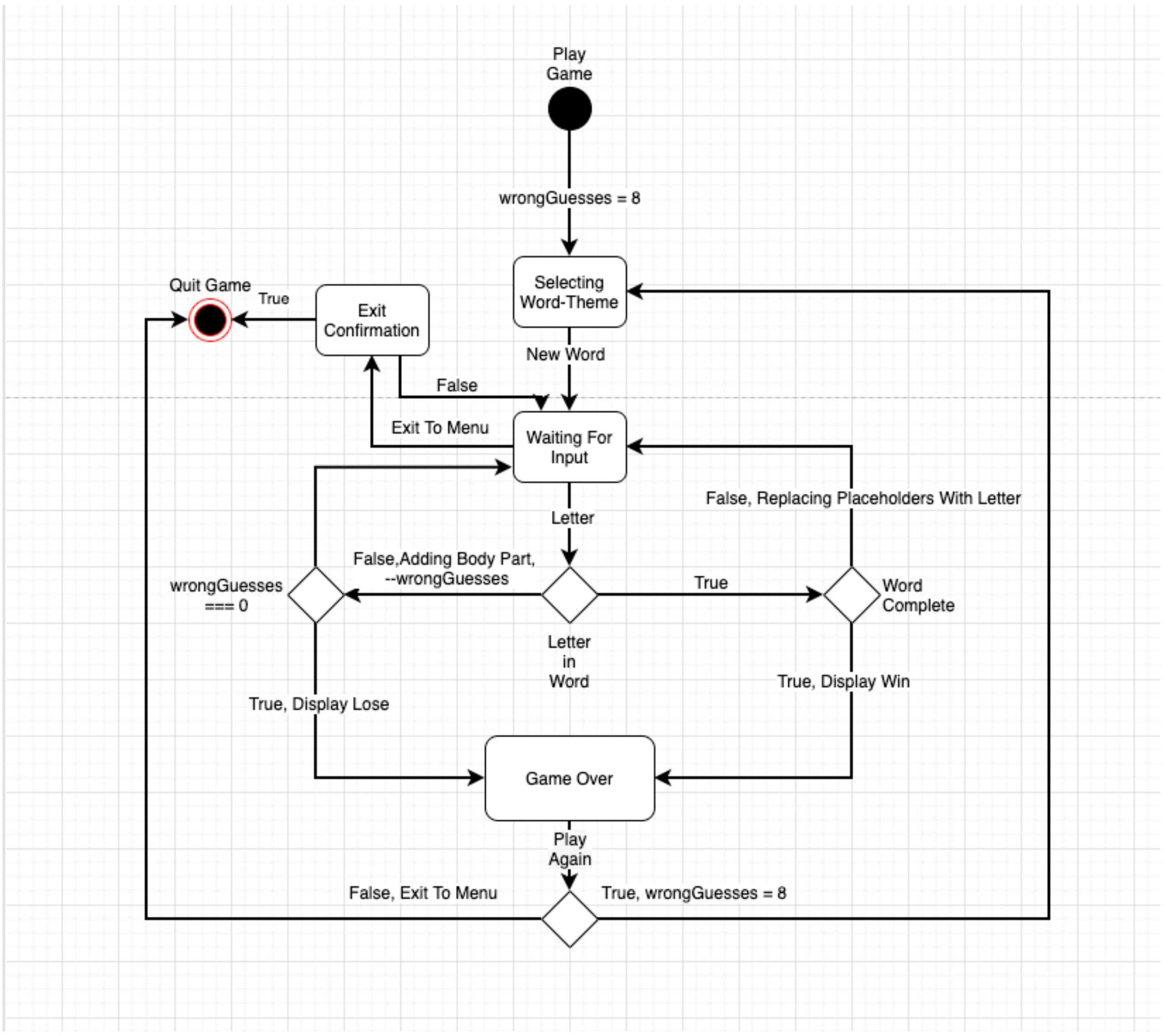


Fig 2 Play Game State Machine

9.3 | Class Diagram

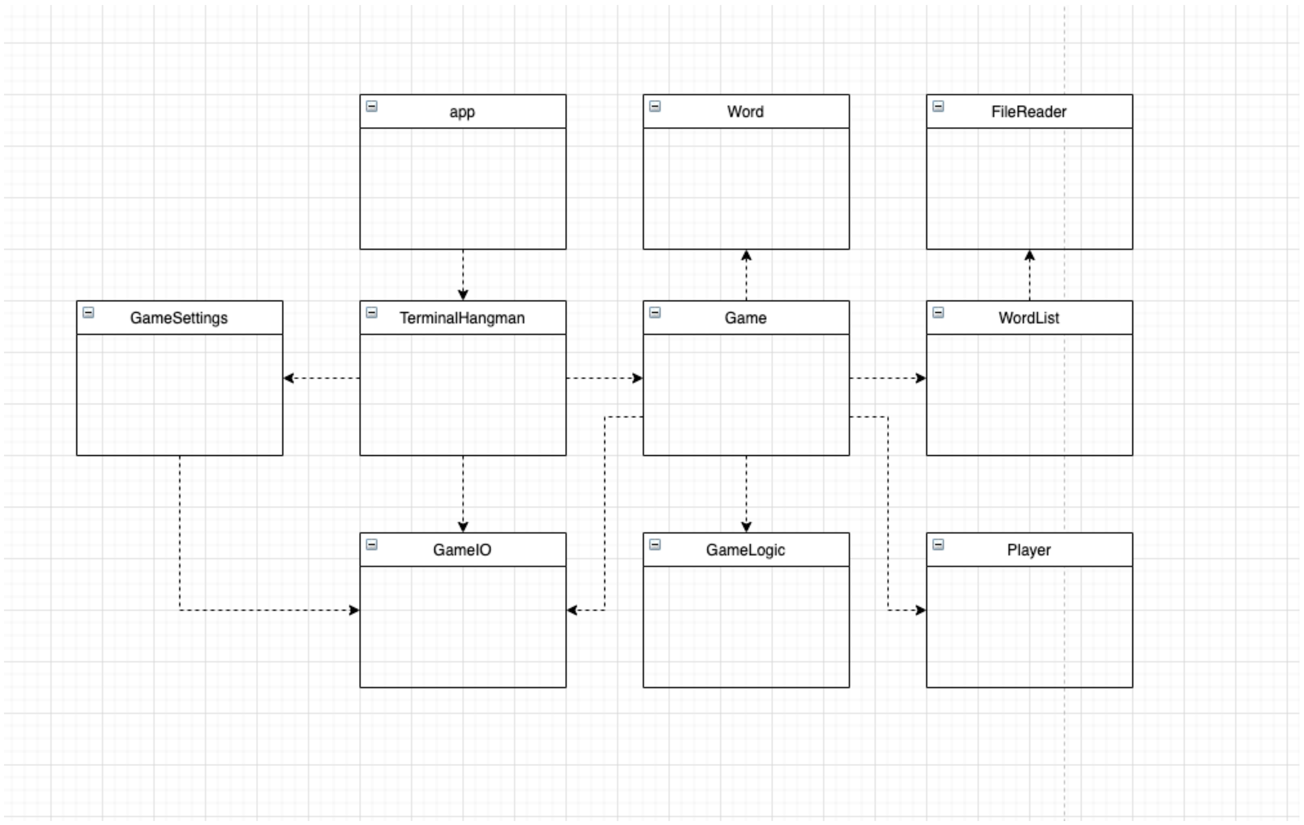


Fig 3 Class Diagram

10 | TESTING

10.1 | What to test and how

We are going to test Use Case 3 (Quit Game), Use Case 2 (Play Game), Use Case 4 and Use Case 5 by running dynamic test-cases. Use Case 2, 4 and 5 will not be tested until the functionality to test that use-case is implemented. We will also write automated unit test for the methods ”getFilesInDir” and ”getJsonFile” in the FileReader Class.The method getFilesInDir will have two tests and getJsonFile will have three tests.

10.2 | Rationale

The reason for choosing to test Use Case 3 is to see if it is possible to exit the application. The reason for choosing Use Case 2 is to test if it was possible to play through the game the way that the Use Case described it. The reason for testing Use Case 4 and 5 is to test some menu flow when the user navigates trough different menus and also to test that the ”Game Mode” and the ”Difficulty” actually change.
The reason for choosing to write unit tests for ”getFilesInDir” and ”getJsonFile” is to see if the error handling for those methods works as intended. Since the file path for the word lists are currently ”hard coded” in the program, this will be a good way to find bugs if the file path is changed.

10.3 Manual Test-Cases

10.3.1 | TC 3.1 To quit the game

Link to UC3
Description: The test checks if the player can successfully exit the application. This is tested to check if the player can successfully exit the application from the main menu and to check if the exit confirmation works.
Use case: UC3 Quit Game
Scenario: To quit the game

The main scenario of UC3 is tested when the player successfully quits the game.

Precondition: The game should be started

Test steps

- The system shows the menu with a selected menu item. (The selected item has a blue color and a greater-than sign to the left of the text)
- By using the arrow keys on the keyboard, select the menu item with the text ”Quit Game” and press the enter key.
- The system displays the text ”Are you sure you want to quit” and asks the player to confirm by entering the letter y or n.
- By using the keyboard, enter the letter y and press enter.

Expected

- The system should now be exited

10.3.2 | TC 2.1 Winning the game

Link to UC2
Description: The test checks if the player can successfully play the game and win. This is tested to check that the output of the game changes as intended and that the input works.
Use case: UC2 Play Game
Scenario: Playing the game and winning

The main scenario of UC2 is tested when the player successfully wins a game of hangman.

Precondition: While being in the directory /assignment-3/Terminal-Hangman, the game should be started by entering ”npm start test”.

Test steps

- The system displays the text ”Guessed Letters:”, a drawing of the ”gallows pole”, the text ”Placeholders: __” and asks the user to enter a letter.
- By using the keyboard, enter the letter t the press enter.
- The system displays the text ”Guessed Letters: t”, a drawing of the ”gallows pole”, the text ”Placeholders: t_” and asks the user to enter a letter.
- By using the keyboard, enter the letter e the press enter.

Expected

- The system presents the texts ”YOU WIN”, ”Wrong Guesses: 0” and ”
Correct Word: te”
- The system asks the player to play again.

10.3.3 | TC 4.1 To Change Game Mode

Description: The test checks if the player can successfully change the game mode. This is tested to check if the player can successfully change the game mode and that the game mode actually changes.
Use Case: UC4 Change Game Mode

The main scenario of UC4 is tested when the player successfully changes the game mode.

Precondition: While being in the directory /assignment-3/Terminal-Hangman, the game should be started by entering ”npm start test settings”

Test steps

- The system shows the text "Current Game Mode: Normal" and the text "Current Difficulty: 1" and the settings menu with a selected menu item. (The selected item has a blue color and a greater-than sign to the left of the text)
-
- By using the arrow keys on the keyboard, select the menu item with the text ”Change Game Mode” and press the enter key.
-
- The system shows the text "Invisible mode hides the underscores" and the settings menu with a selected menu item.
-
- By using the arrow keys on the keyboard, select the menu item with the text ”Invisible” and press the enter key.

Expected

- The system shows the text "Current Game Mode: Invisible" and the text "Current Difficulty: 1" and the settings menu with a selected menu item.

10.3.4 | TC 5.1 To Change Difficulty

Description: The test checks if the player can successfully change the games difficulty. This is tested to check if the player can successfully change the games difficulty and that the games difficulty actually changes.
Use Case: UC4 Change Difficulty

The main scenario of UC5 is tested when the player successfully changes the games difficulty.

Precondition: While being in the directory /assignment-3/Terminal-Hangman, the game should be started by entering ”npm start test settings”

Test steps

- The system shows the text "Current Game Mode: Normal" and the text "Current Difficulty: 1" and the settings menu with a selected menu item. (The selected item has a blue color and a greater-than sign to the left of the text)
- By using the arrow keys on the keyboard, select the menu item with the text ”Change Difficulty” and press the enter key.
- The system shows the difficulty menu with a selected menu item.
- By using the arrow keys on the keyboard, select the menu item with the text ”Hard” and press the enter key.

Expected

- The system shows the text "Current Game Mode: Normal" and the text "Current Difficulty: 4" and the settings menu with a selected menu item.

10.4 | Test Report

Test traceability matrix and success

TEST	UC2	UC3	UC4	UC5
TC3.1	0	1/OK	0	0
TC2.1	1/OK	0	0	0
TC4.1	0	0	1/OK	0
TC5.1	0	0	0	1/OK
COVERAGE & SUCCESS	1/OK	1/OK	1/OK	1/OK

Comment

All test cases where successful.

10.5 | Unit Tests

Link to the test code

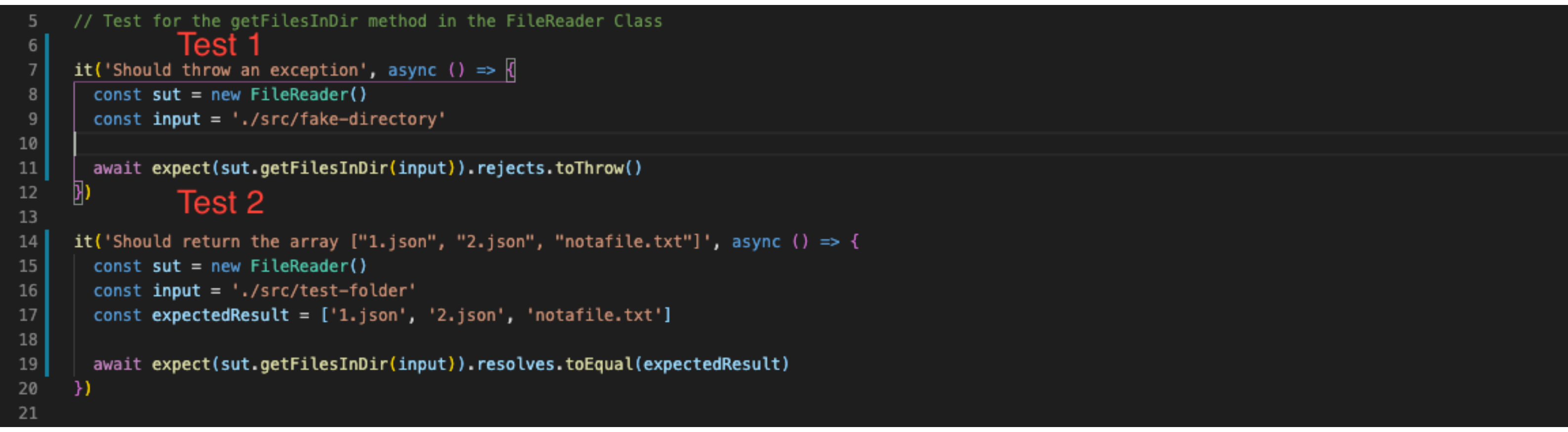
The methods that will be tested ”getFilesInDir” and ”getJsonFile” in the FileReader class. The getFilesInDir method will have two tests and the getJsonFile will have three tests.

10.5.1 | Unit tests for getFilesInDir

Link to getFilesInDir

The function takes a file path to a directory as an argument and returns the files in that directory as an array. If the directory does not exist, the function should throw an exception.

Test 1 will test if the function throws an exception if the directory does not exist.
Test 2 will test if the function returns the files in a specific directory.



```
// Test for the getFilesInDir method in the FileReader Class

it('Should throw an exception', async () => {
  const sut = new FileReader()
  const input = './src/fake-directory'

  await expect(sut.getFilesInDir(input)).rejects.toThrow()
})

it('Should return the array ["1.json", "2.json", "notafile.txt"]', async () => {
  const sut = new FileReader()
  const input = './src/test-folder'
  const expectedResult = ['1.json', '2.json', 'notafile.txt']

  await expect(sut.getFilesInDir(input)).resolves.toEqual(expectedResult)
})
```

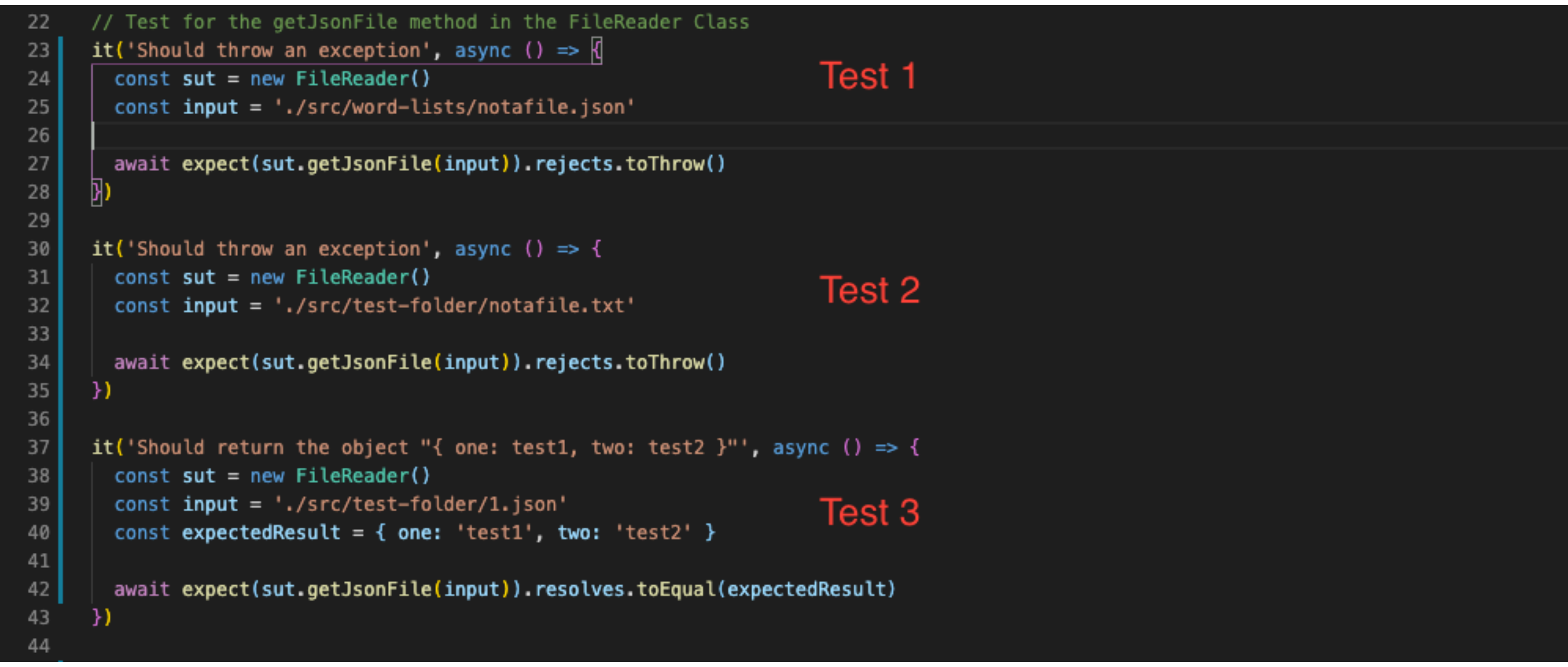
Fig 4 Test for getFilesInDir

10.5.2 | Unit test for getJsonFile

Link to getJsonFile

The function takes a file path to a specific json file as an argument and returns the json file converted to a javascript object. If the passed file path does not exist, the function should throw an exception. If the passed file is not an json file, the function should throw an exception.

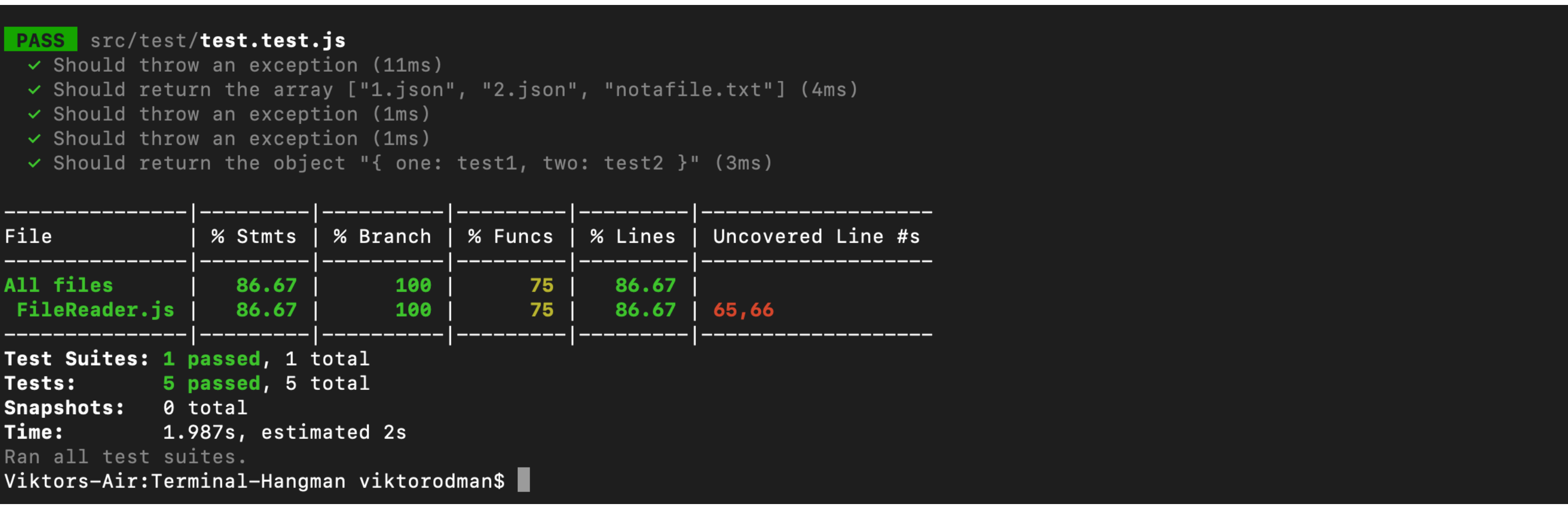
Test 1 will test if the function throws an exception if the passed file does not exist.
Test 2 will test if the function throws an exception if the passed file is not a json file.
Test 3 will test if the function returns a specific javascript object when passed an existing json file.



```
// Test for the getJsonFile method in the FileReader Class
it('Should throw an exception', async () => {
  const sut = new FileReader()
  const input = './src/word-lists/notafile.json'

  await expect(sut.getJsonFile(input)).rejects.toThrow()
})

it('Should throw an exception', async () => {
  const sut = new FileReader()
  const input = './src/test-folder/notafile.txt'

  await expect(sut.getJsonFile(input)).rejects.toThrow()
})

it('Should return the object "{ one: test1, two: test2 }"', async () => {
  const sut = new FileReader()
  const input = './src/test-folder/1.json'
  const expectedResult = { one: 'test1', two: 'test2' }

  await expect(sut.getJsonFile(input)).resolves.toEqual(expectedResult)
})
```

Fig 5 Test for getJsonFile

10.5.3 | Test Result



```
 PASS  src/test/test.test.js
  ✓ Should throw an exception (11ms)
  ✓ Should return the array ["1.json", "2.json", "notafile.txt"] (4ms)
  ✓ Should throw an exception (1ms)
  ✓ Should throw an exception (1ms)
  ✓ Should return the object "{ one: test1, two: test2 }" (3ms)

---------------|----------|----------|----------|----------|-------------------
File           |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s
---------------|----------|----------|----------|----------|-------------------
All files      |    86.67 |      100 |       75 |    86.67 |
 FileReader.js |    86.67 |      100 |       75 |    86.67 | 65,66
---------------|----------|----------|----------|----------|-------------------
Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        1.987s, estimated 2s
Ran all test suites.
Viktors-Air:Terminal-Hangman viktorodman$ 
```

Fig 6 Test Result

CLASS	TESTED METHOD	TEST RESULT
FileReader	getFilesInDir	2/2 Passed
FileReader	getJsonFile	3/3 Passed

10.6 | Reflection

This was an interesting assignment. I had to refactor a lot of code to make it more testable. It took some time to understand to be very descriptive when writing the manual test cases. Writing a test ”flag” for the game really helped me shorten the test case for UC2.
Learning how to use the tool ”jest” for the automatic unit tests was pretty easy when trying to test simpler functions. I had some struggle when I was testing asynchronous functions and trying to make the function throw exceptions. I can see how creating test for future implementation could be useful.

11 | TIME LOG

The Time Log will present estimation about how much time a particular task Is going to take in terms of learning about the task and implementing it. It will also keep track on how much time the task actually took, and compare it to the estimation.

11.1 | The Time Log

Task	Date	Estimated Time	Actual Time	Analysis
Adding The skeleton-code.	31/01/2020	1 hour	1 hour	Went as estimated.
Reading Chapters 2-3 and 22-23	20/01/2020 - 30/01/20	10 hours	12 hours	Was hard to grasp new concepts. Had to go back in the book reread a lot.
Watching pre-recorded lectures	20/01/2020 - 30/01/2020	3 hours and 23 minutes	3 hours and 23 minutes	Went as estimated.
Create Project Documentation (Creating project-plan, planning for the iterations, identifying risks and create risk strategies)	01/02/2020	10 hours	15 hours	Most of the time was spent on trying to figure what to include and what to leave out. It was hard to make a good time estimate since this is the first iteration.
Reading Chapters 4, 5, 20 and 6, 7, 8	03/02/2020 - 03/20/2020	15 hours	12 hours	Reading went faster than estimated
Watching pre-recorded lectures	03/02/2020 - 03/20/2020	10 hours	10 hours	Went as estimated.
Update the project plan	24/02/2020	2 hours	3 hours	Change the project plan with the new information gathered in this iteration.
Making time estimations	10/02/2020	2 hours	2 hours	Went as expected time-wise.
Expand the use case model	12/02/2020	2 hours	1 hour	No problems occurred during this task

Task	Date	Estimated Time	Actual Time	Analysis
Creating a fully dressed use case for Play Game	12/02/2020	2 hours	3 hours	Most of the time was spent figuring out how to write Usecases
Creating a Play Game State Machine	13/02/2020	3 hours	6 hours	Changed the diagram many times. At first, the diagram looked too much like a Sequence Diagram.
Implementation	17/02/2020	4 hours	6 hours	Figuring out how the different dependencies that will be used work, took longer than expected
Modeling Structure	24/02/2020	2 hours	1 hour	Made a simple class diagram that describes the dependencies between the different files
Reading Chapter 8	25/02/2020 - 06/03/2020	4 hours	4 hours	Went as estimated.
Watching pre-recorded lectures	25/02/2020 - 06/03/2020	4 hours	4 hours	Went as estimated.
Writing Manual Test Cases	06/03/2020	2 hours	3 hours	Went pretty good time wise.
Writing Unit tests	09/03/2020	4 hours	6 hours	Some problems occurred when trying to test asynchronous code. So went over the estimated time
Running the manual tests	09/03/2020	30 minutes	10 minutes	No problems occurred during this task
Writing a reflection	09/03/2020	1 hour	1 hour	No problems occurred during this task

Task	Date	Estimated Time	Actual Time	Analysis
Creating Fully Dressed Use Cases for UC4 and UC5	19/03/2020	1 hour	2 hour	No problems occurred during this task
Implementing new features for iteration 4	19/03/2020	2 hours	3 hours	Some code had to be refactored to make the new features work.
Create Test Cases	20/03/2020	1 hour	1 hour	A new start flag was added to make the test cases easier.
Running the manual tests	20/03/2020	10 minutes	5 minutes	No problems occurred during this task
Updating the project plan	20/03/2020	4 hours	5 hours	No problems occurred during this task
TOTAL TIME		90 hours 3 minutes	104 hours 38 minutes	14 hours over the estimated time

11.2 | Reflection

Went 14 hours over the estimated time. Was pretty hard to estimate the time since almost everything that we did in this project was new. The time tracking that went in to this project was not 100% accurate. I made some estimations of what the actual time spent on a task was when I simply forgot to track the time. In my next project I will probably use some kind of tool to keep track of time when working on tasks. It was also hard to keep track of time since it's hard to know if was only supposed to track effective time (when I was actually getting things done) or if should keep track of time when I was researching things.