Hang the Man Project Plan
Updated for Iteration 4

By

Thimmy Stenlund
ts222ub
ts222ub@student.lnu.se

Index for the revised Project Plan

Revision History

Date	Version	Description	Author
08/02/19	1.0	Hang The Man project setup and Project Plan creation	Thimmy Stenlund
15/02/19	1.1		
18/04/19	1.2	Finished Project Plan, ready for handover	Thimmy Stenlund

General Information

Project Summary	
Project Name	Project ID

Hang The Man	1073
Project Manager	Main Client
Thimmy Stenlund	Friends & Family
Key Stakeholders	
Thimmy Stenlund ReturnsVoid	
Executive Summary	

VISION

Returns Void envisions a Hangman Game slightly outside the ordinary boundaries of the classic Hangman specifications. While it will support (and start as) the basic, normal game

flow that everyone has come to love (and hate), the aim is also to support a secondary game mode that aims to hang the man

– thus, showing how the name ‘Hang the Man” came to be. First, core features will be implemented, supporting the original game. As you start a new game, you will see a series of underlines showing how many letters the word contains. In turns, you will guess a letter. If the letter is found within the word, it will then display the letter instead of underline in the correct position of the word. Should the game not find your guessed letter in the word, you will see that with each failed letter – an island, gallows, nose and finally a person hang in the noose will appear in the console. When all parts are visible – you lose. If you manage to solve the word, there should be a winner announcement and in future iterations of the game – a scoreboard displaying number of guesses and time it took. When the base game is functionable, to a reasonable degree, development of the “Hang The Man” special mode will begin – where you aim to hang the man, and not save him from hanging. In this mode, there will only be words with 7 letters. If you manage to guess all 7 letters without more than 5 missed guesses, the (guilty) man will hang. It will be developed with JavaScript as its main language.

Key Needs & Features:

* Core Game – Guess letters in a word and either fail or win depending if you can guess it before losing.
* Feature – Reverse the game mode, so that the plan is to actually hang the man.
* ~~Feature – Time it took to find out the word.~~
* ~~Feature – High score list sorted after the time it took.~~
* Feature (New) – Random Question, true or false, needed to be answered to win Hang the Man playmode
* Feature (New) – Expanded list of words that could appear

Revision 1.2 – What changed and Why (Motivation)

The feature regarding including a high score list and a timer that counted how long it took to figure out the word, was scrapped after having both the kids (two of the main clients) play test the current game. They proposed to instead build out the library of available words and also include that the user must answer a random question (true or false) in the Hang the Man game, to eventually actually Hang the Man. If you fail to answer correctly, the Man goes free.

Project Plan

Introduction

Hang The Man is a classic with a twisted take on the game mode that has enthralled millions across the world, both on pen & paper but also through games. We aim to create a new variant of this classic – with a new game mode that aims to have you hang the (guilty) man before the time runs out.

Justification

Todays minigames can't hold up to the true classics. One important factor is that it's a learning game – meaning that a child/youth can learn English and English words from playing the game. The twist with having a game mode that will instead hang the man, might be borderline offensive to some – but there are a lot of other games that brings much more violence and gore to young children. This is just a fun twist.

Stakeholders

Thimmy Stenlund – The current owner and founder of ReturnsVoid.

ReturnsVoid – A programmer group that focuses on creating simple and reliable applications & games.

Resources

Resource – Time:

For this project, roughly 20 hours per week will be allocated in different spans over the weekdays and weekends.

Resource – Money:

The budget set aside for this project is currently standing on 0$. Time spent will allot for the actual cost (as time always has a monetary value in reality). Should a cost occur, that will be discussed and agreed upon before added to the resource field.

Hard- and Software Requirements

We are using JavaScript as our coding language to develop Hang The Man. The application will be developed on both laptop and stationary computer. Requirements to run the game, would be a PC, Phone or Tablet that can handle a web-browser and thus see the console, as the game will be played there.

Resource – Personnel:
Project Manager – Thimmy Stenlund
Lead Developer – Thimmy Stenlund
Environment Specialist – Thimmy Stenlund
Test Engineer – Thimmy Stenlund

Overall Project Schedule (Rev 1.2)

First step (Process and Planning) is to be finished by 8/2 2019.

Second step (Modelling and Software Design) is to be finished by 21/2 2019.

Third step (Software Testing) is to be finished by 8/3 2019.

Fourth and final step (Iteration 4 – Finished Game/Project) is to be finished by 21/3 2019. Update 2019-04-18: This was delayed until this date due to illness and lack of time.

Slight delays can impend on the time plan and later dates can be determined, should the time plan fail. Second iteration will need to be polished, motivated by the corrections addressed by the assigned teacher. Those edits will be present further down in this document.

Scope, Constraints and Assumptions

Scope for the project is displayed using the MoSCoW method.

Must:

- Core gameplay, the possibility to play the game in Console.

Should:

- Optional game mode that consists of actually hanging the man
- ~~High Score list~~ Feature has been suspended indefinitely.
- ~~Time taken~~ Feature has been suspended indefinitely.

Could:

- Database or Storage that store the high scores and/or the word list that is used to randomize a word.

Won't:

- Graphics
- Multiplayer
- Global Score list
- Installation executable

Added Features with iteration 4

- Expanded Words library
- Hang the Man game mode gives a true or false question that need to be correct for winning the game.

ITERATIONS

Plan for four iterations, including this. This is a fine-grained plan on what is to be done in each iteration and with what resources. To begin with, this is a plan of what we *expect* to do, update this part with *additions* (never remove anything) when plans do not match up with reality. Also make time estimates for the different parts. In this course the overall planning has in some ways already been decided, so use the template to provide more details on specific tasks that define *your* project. Remember that you can plan to add features to any of the phases as long as the main focus is also met. The first assignment is to complete iteration one.

5.1 Iteration 1

What to do	How to do it	Who does it	Est. time	Deadline
Complete Project Plan	Add text and data into the current document.	Thimmy Stenlund	10 hours	8/2 2019
Create Github Repo	Setup a new github repo from the instructions on the moodle page.	Thimmy Stenlund	15 minutes	8/2 2019
Add Skeleton Code	Use Visual Studio Code to add “Skeleton” code to the Git repo.	Thimmy Stenlund	15 minutes	8/2 2019
Turn in Assignment 1	Check that documents are uploaded. Send link to git repo on moodle.	Thimmy Stenlund	15 minutes	8/2 2019

5.2 Iteration 2

What to do	How to do it	Who does it	Est. Time	Deadline
Model w/ UML	Model features in UML.	Thimmy Stenlund	4h	13/2 2019
Add Diagrams (UML) to Project Plan	Edit this document to have it display the UML for the features to be implemented.	Thimmy Stenlund	30m	13/2 2019
Start the game with a const word.	Add a const word, display _ for each letter. Guess all letters to win. (No fail available)	Thimmy Stenlund	4h	14/2 2019
Construct a counter for number of guesses to fail.	Add the counter that counts failed guesses. Make the player lose if that counter reach x.	Thimmy Stenlund	2h	14/2 2019
Arrange a way to display the hanging.	Add console.log calls that shows each step of the hanging as the player fails. “You lose” message to be displayed.	Thimmy Stenlund	2h	15/2 2019

Timer for the game.	Add a timer that count as the play2er guesses.	Thimmy Stenlund	2h	16/2 2019
Create a “menu” to start the game or optional game mode.	Menu addition will go fast. What is also required is turning around the main mechanic (new graphics). Rest can be reused and just revamped.	Thimmy Stenlund	4h	18/2 2019
Turn in Assignment 2	Add the code and documents to the git repository.	Thimmy Stenlund	30min	21/2 2019

5.3 Iteration 3

What to do	How to do it	Who does it	Est. Time	Deadline
Plan Tests	Plan the tests that needs to run through for these steps to be completed.	Thimmy Stenlund	12h	24/2
Perform	Implement and run tests on the application. Review and correct any issue on the code.	Thimmy Stenlund	16h	2/3 2019
Document the tests , Turn in Assignment 3	Make sure that the documentation of the tests, the results and all Is collected and presented. Make Release	Thimmy Stenlund	4h	8/3 2019

5.4 Iteration 4

What to do	How to do it	Who does it	Est. Time	Deadline
Check towards the Project Plan	Compare requirements, see to it that all test run through.	Thimmy Stenlund	6h	11/3 2019
Review what features could be desired in the future.	Review and go through the code and playtest the completed product. Anything small to be added, removed or changed?	Thimmy Stenlund	2h	15/3 2019
Turn in Assignment 4	Add the code, documentation and the tests to the git repository. Build Final Release.	Thimmy Stenlund	2h	20/3 2019

Risk Analysis

All projects face risks that make it important to prepare for what might happen. Use the chapters in the book as well as the content of the lectures to identify the risks within this project. As always, write down your reflections on creating a risk analysis. This reflection should be about 100 words.

List of risks

List the identified risks and specify, as far as possible, the probability of them happening as well as the impact they would have on the project.

Risk	Probability	Impact on the Project
Project team member will not be in place when required	High	Severe
Risks with the hardware and software for the development	Low	Mediocre
Risk that the workstation environment of the user will experience malfunction	Low	Low
Risk to the project resulting from a mandatory completion date connected with the project	Medium	Low
Risk of exceeding the current expected deadline(s)	Medium	Mediocre

Revision 1.2 – Review

None of the above risks did affect the project, however due to an issue with Iteration 2, the state machine will be re-reviewed and thus the final deadline for the separate iteration completions will not be met (as in – the first final deadline will not be met. Assignment 2 will be completed by 5th of April). Update #2: Assignment 4 will be completed 2019-04-18.

Strategies

Project team member will not be in place when required:

To minimize this risk, there will be a fluent schedule to have multiple times available to complete it (before its deadline). There should also be gaps that has not been accounted for

that can be used to catch up, if the schedule would appear to not hold. It would lower the Impact from Severe to Low.

Risks with the hardware and software for the development:

Make sure to test any type of modules or solutions before using them in a release environment. Do not add new technologies without making sure they work on the end environment.

Risk that the workstation environment of the developer will experience malfunction:

There are several workstations available, all setup with VSC and would just need GitHub to connect to the git repository. This remains a low risk.

Risk to the project resulting from a mandatory completion date connected with the project:

As mentioned before – this will be minimized as the time estimation is put high, thus making sure that extra time can be used to complete the iterations, if needed. There are also soft deadlines, that means that a later turn in can be done – although it should be avoided.

Risk of exceeding the current expected deadline(s):

There is a risk that the deadlines could be breached but with fluently allocated time and pockets of extra time that can be spared, chained with the deadlines being “soft” deadlines (a later turn in can be done but should always be avoided), it’s as good as it can get.

Iteration 2 – Use Case, Unit Tests and General Testing

Use Case Model UC 1 Start Game

Precondition: none.
Postcondition: the game menu is logged in console.

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 a standard game, the Hang The Man version of the game and quit the game.

3. The Gamer makes the choice to play the standard game.

4. The system starts the standard 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 3)

4.1 Invalid menu choice
1. The system presents an error message.

2. Go to Main scenario 2

5.1 The Gamer makes the choice to play the Hang The Man version of the game.
1. The system loads the alternative Hang The Man version. (see Use Case 4)

UC 2 Play Game Precondition: The game is running. Postcondition: The standard game mode is active

Main scenario

1. Starts when the user chooses the "Play Standard Game" option in the menu.

2. A random word is selected from an array, which is not shown to the user.

3. For each letter in the word, an underline is displayed.

4. The user can guess letters, if it's correct – the user can continue. If its wrong, the counter will increase by one.

5. If the user guesses all letters before the counter reaches 8, the user will see a "You Win!" being logged to the console, together with the time it took, and an option to go back to the menu. (see Use Case 1)

6. If the counter goes to 8, i.e. the user guesses wrong letter 8 times, the user will see a "You lose!" being logged to the console, with an option to go back to the menu. (see Use Case 1) 2.1 The Gamer wants to leave the game and go back to the main menu. (see Use Case 1)

Alternative scenario

UC 3 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
1. The system returns to its previous state

UC 4 Start Hang The Man Game (Updated rev 1.2, Itr 4)

Precondition: The game is running.
Postcondition: The game menu is logged

Main scenario

1. Starts when the user chooses the "Play Standard Game" option in the menu.
2. A random word is selected from an array, which is not shown to the user.
3. For each letter in the word, an underline is displayed.
4. The user can guess letters, if it's correct – the user can continue. If it's wrong, the counter will increase by one.
5. If the user guesses all letters before the counter reaches 8, the user will be presented with a true or false question. If answered correctly, the user will see a "You Win! The Man is Hanged" being logged to the console, together with the time it took, and an option to go back to the menu. (see Use Case 1). If the user answers incorrect, the man will go free and the User will have lost (see step 6 below). If the user will write neither true or false, the User will be presented with a degrading comment and the man will go free (see step 6 below.) // Updated for Rev 1.2 (Itr 4)
6. If the counter goes to 8, i.e. the user guesses wrong letter 8 times, the user will see a "You lose! The Man is Free" being logged to the console, with an option to go back to the menu. (see Use Case 1)

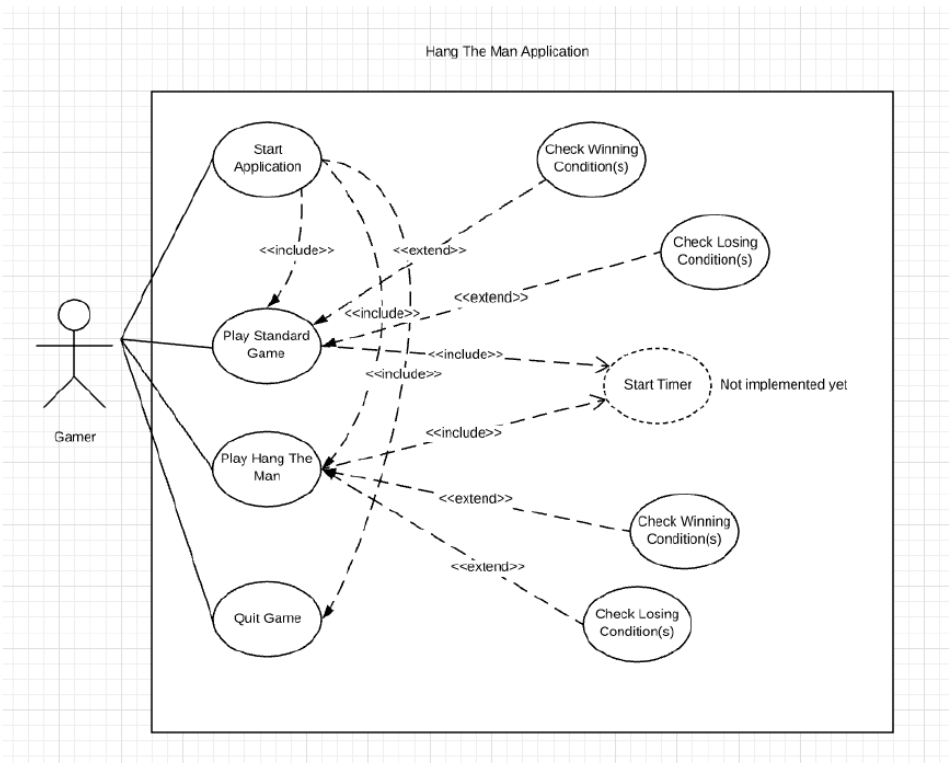
Note: The difference between the game mode "Standard" and "Hang The Man" is purely graphical. In standard, you are aiming for not hanging the man. In "Hang The Man" you are struggling to actually hang the man.

Use Case for "Play Game"

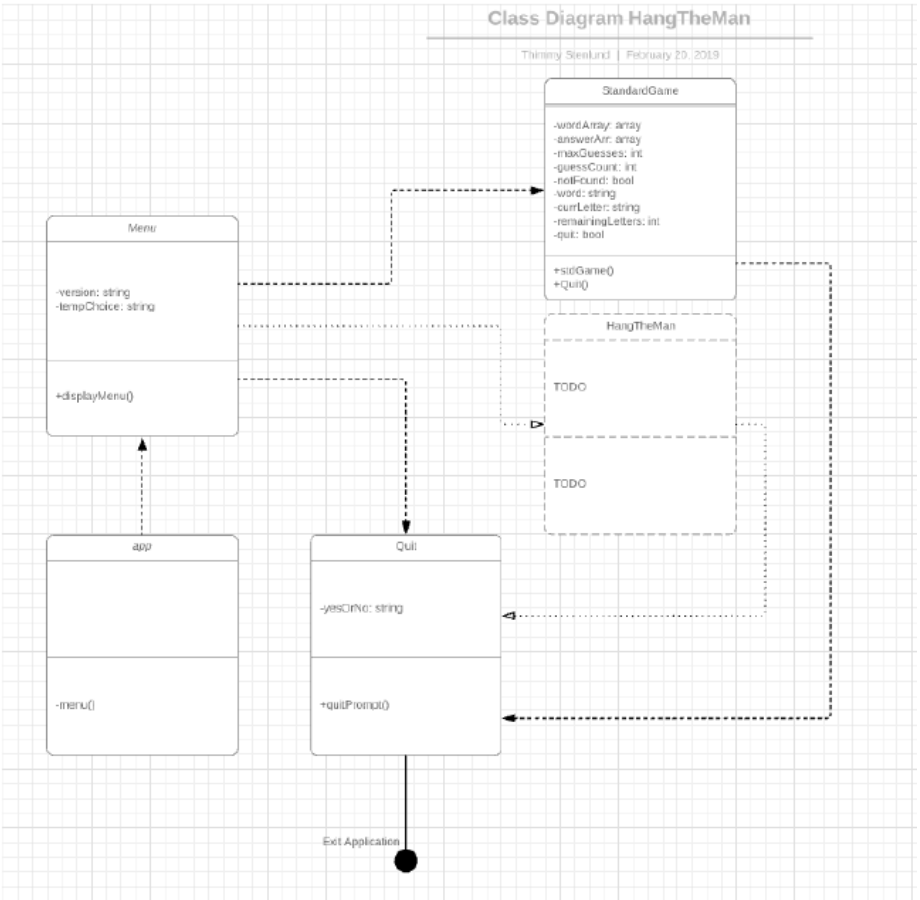
Scenario: Winning the game A gamer wants to play the standard version of the hangman game. The game displays a menu to display the choices for the gamer to choose from. The gamer then chose to initiate the "Play Game" menu choice. The game boots up the standard version of the game, choosing a word at random from the pre-defined array and display several underlines consistent with the number of letters in the (now hidden) word. The gamer will then guess letters that are found inside the word. As the gamer finds the correct letters and thus unveils the word, the game prints out "You win!" followed by the time it took for the user to guess.

Scenario: Loosing the game A gamer wants to play the standard version of the hangman game. The game displays a menu to display the choices for the gamer to choose from. The gamer then chose to initiate the "Play Game" menu choice. The game boots up the standard version of the game, choosing a word at random from the pre-defined array and display several underlines consistent with the number of letters in the (now hidden) word. The gamer will then guess letters that are not found inside the word. As the gamer guesses the incorrect letters more than 7 times, and thus hangs the man, the game prints out "You lose!". The options to either Quit or go back to Main menu is then presented to the gamer.

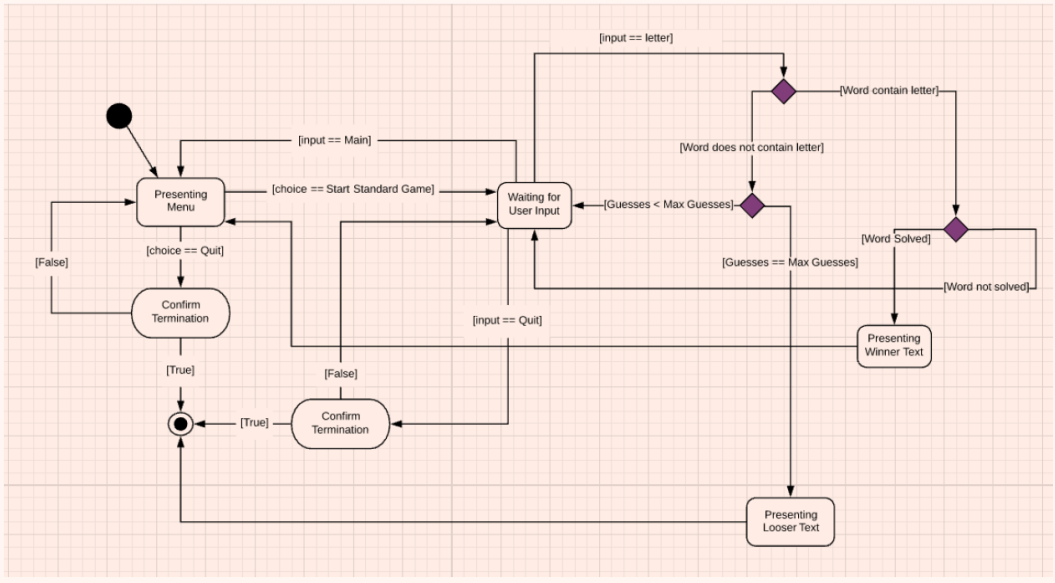
Use Case Diagram



Class Diagram for HangTheMan Application



Play Game State Machine Diagram



Time Log for Use Cases and Diagrams

Date	Time	Task	Actual Time	Analysis
12/2 2019	4h	Model w/ UML	3 hour	Close to the calculated time.
13/2 2019	30m	Add Diagrams (UML) to Project Plan	30m	-
13/2 2019	4h	Start the game with a const word.	2h	Modified to start with a constant array of words
14/2 2019	2h	Construct a counter for number of guesses to fail.	30min	Took less time than expected
14/2 2019	2h	Arrange a way to display the hanging.	1h	The route of “easy” was taken. It displays “_” for each unknown letter in the randomly selected word.
15/2 2019	2h	Timer for the game.	-	Not Implemented (Saved for Assignment 3)
16/2 2019	4h	Create a “menu” to start the game or optional game mode.	4h	Due to issues with reloading the menu from playing the game, it took about the time set aside to get it to function.
20/2 2019	30min	Turn in Assignment 2	30 min	-
25/2 2019	4h	Create Tests and run them, Assignment 3	4 h	-
18/3	4h	Complete code for Assignment 4	3h	Took shorter than expected. Most was done.

Test Plan

Objective

The object is to test the code that was implement during assignment 2, the core game play, but also get a glimpse of possible future testing of the “Hang The Man” extra feature mode that will be implemented in iteration 4.

What to test and how

Our intent is to test Use Case 2 (“Play Game”), Use Case 3 (“Quit Game”) and Use Case 4 (“Play Hang The Man”) by writing and running dynamic manual test-cases, to begin with. We will also examine the code for Use Case 2 and for the unimplemented Hang The Man feature mode. We will review the code as well, to see where and how we could create unit tests to span all methods eventually. IMPORTANT: The Test.js file content will be included in the end of this document.

(Screenshot included from the Use Model and Use Cases documentation, from Assignment 2)

UC 2 Play Game

Precondition: The game is running.

Postcondition: The standard game mode is active

Main scenario

1. Starts when the user chooses the "Play Standard Game" option in the menu.
2. A random word is selected from an array, which is not shown to the user.
3. For each letter in the word, an underline is displayed.
4. The user can guess letters, if it's correct – the user can continue. If its wrong, the counter will increase by one.
5. If the user guesses all letters before the counter reaches 8, the user will see a "You Win!" being logged to the console, together with the time it took, and an option to go back to the menu. (see Use Case 1)
6. If the counter goes to 8, i.e. the user guesses wrong letter 8 times, the user will see a "You lose!" being logged to the console, with an option to go back to the menu. (see Use Case 1)

Alternative scenario

2.1 The Gamer wants to leave the game and go back to the main menu. (see Use Case 1)

UC 4 Start Hang The Man Game

Precondition: The game is running.

Postcondition: The game menu is logged

Main scenario

1. Starts when the user chooses the "Play Standard Game" option in the menu.
2. A random word is selected from an array, which is not shown to the user.
3. For each letter in the word, an underline is displayed.
4. The user can guess letters, if it's correct – the user can continue. If it's wrong, the counter will increase by one.
5. If the user guesses all letters before the counter reaches 8, the user will see a "You Win! The Man is Hanged" being logged to the console, together with the time it took, and an option to go back to the menu. (see Use Case 1)
6. If the counter goes to 8, i.e. the user guesses wrong letter 8 times, the user will see a "You lose! The Man is Free" being logged to the console, with an option to go back to the menu. (see Use Case 1)

Note: The difference between the game mode "Standard" and "Hang The Man" is purely graphical. In standard, you are aiming for not hanging the man. In "Hang The Man" you are struggling to actually hang the man.

UC 3 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

1. The system returns to its previous state

How the testing will be done

UC2 , UC3 and UC4 will be dynamically tested through Unit Tests, covering different sections of the UC's across several tests and there will also be manual tests done on at least two of the User Cases.

Time plan

Task	Estimated	Actual
Manual TC 1.1	30m	20m
Manual TC 1.2	30m	15m
Unit Test 1.1	30m	10m
Unit Test 1.2	30m	30m
Unit Test 2.1	30m	45m
Unit Test 2.2	30m	30m
Unit Test 2.3	30m	30m
Running Manual Tests	15m	15m
Running Unit Tests	30m	60m

Comment on Time Plan: Mostly accurate. Time exceeding estimating was mostly due to refactorizing the whole game to have it fit a somewhat decent Test Strategy and Test Execution. Hope to get more tuned in to actual versus estimated.

Updated comment for Rev 1.2: Made a new test for the new section of the Hang the Man game mode, as a new test for Iteration 4 and it's new functionality.

Manual Test Cases

TC1.1 The player will choose an allowed, not previously used letter that is found in the word.

The Hangman application will ask the user for a letter. In this TC the user will select a non-reused letter that can be found in the hidden word itself.

Precondition: The game must be running. The hidden word must be pre-determined ("FOX").

UC: 2

Test Steps:

(* Start the application) - Precondition

(* Play a standard Game) - Precondition

* The pre-defined word will be "FOX"
* The user will try to guess the letter "O" by pressing the "O" button and follow that up with Enter key.
* The application will display the letter "O" in the centre of the word and let the user guess again.

☒ Test Succeeded

Comments: No irregularities detected. Test Steps was followed and the Test itself returned the expected result.

Screenshot snippet of the Letter handling in the StandardGame.js and the console.

```
26    let currLetter = readlineSync.question('Guess a letter (type "Quit" to quit, "Main" to go back to Main Menu): ')
27    currLetter = currLetter.toLowerCase()
28    notFound = true
29    if (currLetter.length !== 1 && currLetter !== 'quit' && currLetter !== 'main') {
30      console.log('Please enter a single letter! (type "Quit" to quit, "Main" to go back to Main Menu)')
31    } else if (currLetter === 'quit') {
32      quit = Quit()
33    } else if (currLetter === 'main') {
34      console.log('')
35      clearGameState()
36      return true
37    } else {
38      for (let j = 0; j < word.length; j++) {
39        if (word[j] === currLetter && answerArr[j] !== currLetter) {
40          answerArr[j] = currLetter
41          remainingLetters--
42          notFound = false
43        }
44      }
45      if (notFound && currLetter !== 'quit' && currLetter !== 'main') {
46        guessCount++
47        console.log(`Letter ${currLetter} not found.`)
48        console.log(`You have ${(maxGuesses - guessCount)} guesses left!`)
49      }
50    }
```

```
> ts222ub_1dv600@1.0.0 start C:\Users\Kong\Documents\Högskola\Webbprogrammering LNU\1dv600\ts222ub_1dv600
> node app.js

HangTheMan v1.0
--------MENU-------
1: Play Standard Game
2: Play HangTheMan
3: Quit
------------------
Option: 1

_ _ _
Guess a letter (type "Quit" to quit, "Main" to go back to Main Menu): o
_ o _
Guess a letter (type "Quit" to quit, "Main" to go back to Main Menu):
```

TC2.1 The user decides to quit the application in the menu

The menu will ask for an option that the user will choose. The user will select 3 for Quit and confirm that he wants to quit the application.

UC: 3

Precondition: The application must be running. The user must be situated in the main menu.

Test Steps:

(* Start the application) - Precondition

* The user will select “3”
* The application will display a question “Are you sure you want to quit? (Yes or No): “ * The user will type “Yes” and press the Enter key. * The application will stop running.

☒ Test Succeeded

Comments: No irregularities detected. Test Steps was followed as per above definition and the Test itself returned the expected result.

Screenshot of Quit.js and the Manual Testing Session.

```
'use standard'

const readlineSync = require('readline-sync')

function quitPrompt () {
  let yesOrNo = readlineSync.question('Are you sure you want to quit? (Yes or No): ')
  yesOrNo = yesOrNo.toUpperCase()
  if (yesOrNo === 'YES') {
    return true
  } else if (yesOrNo === 'NO') {
    console.log('')
    return false
  } else {
    quitPrompt()
  }
}

module.exports = quitPrompt
```

```
> ts222ub_1dv600@1.0.0 start C:\Users\Kong\Documents\Högskola\Webbprogrammering LNU\1dv600\ts222ub_1dv600
> node app.js

HangTheMan v1.0
--------MENU-------
1: Play Standard Game
2: Play HangTheMan
3: Quit
--------------------
Option: 3
Are you sure you want to quit? (Yes or No): Yes
PS C:\Users\Kong\Documents\Högskola\Webbprogrammering LNU\1dv600\ts222ub_1dv600>
```

Unit Tests (Automatic Tests)

Unit Test 1.1: For each letter in a "random" word – create a underscore that will be presented to the User

StandardGame (UC2) Testing - #createUnderscoreArr: Create Empty Array with _ for each char

When the User opt to play a standard game, the application will display a number of underscores (_) connected to the number of characters in the random selected word.

UC: 2 (Play Standard Game)

Precondition: The application must be running. A preselected word will be used - "Stormcloud".

Test Steps:

(* Start the application) – Precondition

* The user selects "1" (Play Game)
* The Game will display a total of 10 underscores, each representing a character of the "random" (preselected in the Test) word

☒ Test Succeeded

Comments: No irregularities detected. Test Steps was followed as per above definition and the Test itself returned the expected result.

Screenshot of the createUnderscoreArr function from the StandardGame.js module.

```
function createUnderscoreArr (w) {
  let aArr = []
  for (let i = 0; i < w.length; i++) {
    aArr[i] = '_'
  }
  return aArr
}
```

```
> ts222ub_1dv600@1.0.0 test C:\Users\Kong\Documents\Högskola\Webbprogrammering LNU\1dv600\ts222ub_1dv600
> mocha


  StandardGame (UC2) Testing
    #createUnderscoreArr: Create Empty Array with _ for each char
      √ should return ('_','_','_','_','_','_','_','_','_','_')
```

Unit Test 1.2: When there is time to play a new game or when game is lost, the game need to clear the values and return true (so that the program might now that the game has been cleared)

StandardGame (UC2) Testing - #clearGameState: Clear the values and return true

When the user either returns to main menu, quits the game or wins – the game will clear the game state (as in – put the counter of moves back to zero and return true

UC: 2 (Play Standard Game)

Precondition: The application must be running. We assume that the user chose to go to Main.

Test Steps:

(* Start the application) – Precondition

* The User opts to go back to main

☒ Test Succeeded

Comments: No irregularities detected. Test Steps was followed as per above definition and the Test itself returned the expected result.

Screenshot of the clearGameState function and the test case:

```
function clearGameState () {
  answerArr = []
  guessCount = 0
  return true
}
```

```
  describe('#clearGameState: Clear the values and return true ', function () {
    it('should return true ', function () {
      let shouldBeTrue = false
      shouldBeTrue = stdGame.clearGameState()
      assert.deepEqual(shouldBeTrue, true)
    })
  })
})
```

Unit Test 2.1

Play Hang the Man testing - '#manGoesFree: console.logs that the man goes free and returns false
If the user either runs out of guesses or if the (not implemented) checkFinalAnswer function ends with a faulty answer to the last question, this function is run.

UC: 4 (Play Hang the Man)

Precondition: The game mode “Hang the Man” must be running.

Test Steps:

(* Start the application) – Precondition

(* Start Hang the Man) - Precondition

* The user runs out of guesses
* The manGoesFree function is activated, send a console.log and returns false.

☒ Test Succeeded

Comments: No irregularities detected. Test Steps was followed as per above definition and the Test itself returned the expected result.

Screenshot of the manGoesFree function and the test case:

```
function manGoesFree () {
  console.log('The man goes free. No one will be hanged today. You lose!')
  return false
}
```

```
describe('Play Hang the Man(UC4) Testing', function () {
  describe('#manGoesFree: console.logs that the man goes free and returns false', function () {
    it('should return false', function () {
      let testVal = hangTheMan.manGoesFree()
      assert.equal(testVal, false)
    })
  }),
```

Unit Test 2.2

Play Hang the Man Testing - '#rndWordFromArr: Returns a random word from an array of words

When the User starts Hang the Man, there is a random word that is selected from an array from a function outside of the normal "Play Game" function.

UC: 4 (Play Hang the Man)

Precondition: The application is started.

Test Steps:

(* Start the application) – Precondition

* Start Hang the Man
* The function selects a random word from an array (predetermined to only contain the word 'fox') * The function returns the word

☒ Test Succeeded

Comments: No irregularities detected. Test Steps was followed as per above definition and the Test itself returned the expected result.

Screenshot of the rndWordFromArr function and the test case:

Reflections regarding the Hang the Man project:
So, the project has come to an end. With the above information and the code base itself, the project can easily be continued upon. Many features could be added, graphics could be drawn and the idea itself polished until one couldn't stare on it without being dazzled by its shine. I decided to not add more tests, the addition of the True or False question on the end of the Hang the Man play mode felt like it spoke for itself. It gave a little twist, and a three-way path (two that would end up losing you the game).

Things that would need to improve, coming to a new project, would especially be the time estimation (breaking things down to smaller bits) and securing the project towards any towering risks. Delays have been a staple for this project and I would say it will surely continue but making sure the risks are low and that measures are prepared to counter said delays, are key details to make any project work.

I'm fairly satisfied with how the project ended. Hopefully it gives a clear path on to how the Hang the Man game was designed, what features was scrapped and what was added. It should give any designer a good plateau to stand on when continuing on the legacy of what is Hangman.