SOFTWARE DEVELOPMENT PROJECT

Alva Fandrey
1dv600

2019-04-19

| Contents

1 | Revision History

Date	Version	Description	Author
5/2-19	V1.0	Assignment 1	Alva Fandrey
21/2-19	V2.0	Assignment 2	Alva Fandrey
8/3-19	V3.0	Assignment 3	Alva Fandrey
12/3-19	V1.1	Re-take assignment 1	Alva Fandrey
26/3-19	V2.1	Re-take assignment 2	Alva Fandrey
19/4-19	V4.0	Assignment 4	Alva Fandrey

2 | General Information

Project Summary	
Project Name	**Project ID**
Hangman	af222ug_1dv600
Project Manager	**Main Client**
Alva Fandrey	Dog breed enthusiasts
Key Stakeholders	
Manager Developer/Software engineer Tester End-user (Dog breed enthusiasts) Teachers/tutors (in course 1dv600)	
Executive Summary	
In this software development project a console application of the game “Hangman” will be developed as a project for the course 1dv600. The basic features of “Hangman” are that the system randomly picks a word from a predefined list and presents it to the user with equally many underscore signs. Then it is up to the user to guess that word using one letter at the time. To satisfy our main client the predefined list will only contain different dog breeds.	

3 | Vision

The goal with this project is to create a JavaScript console application of the game "Hangman". The basic idea of Hangman is to let the player guess a word one letter at a time.

When the application is started the user will login by entering a nickname. The player should be greeted with a menu where they can choose to start the game, watch the high score list or quit the application. The player should be able to quit the application at all times while playing the game. If the user chooses to start the game the system will then ask the user to choose a level from 1-3 or a random level. Then a word from a predefined list will be randomly picked and the number of letters will be displayed with equally many underscore signs. The player will then guess the word by suggesting letter after letter. For every wrong guess the player has the game will build a part of a man getting hanged. The player can have eight faulty guesses before the game is over. The high score list will present how many guesses a player has had.

Hangman is a very popular game to play and there are many different versions of the game available to play on the internet. To make this game differentiate from other Hangman games found on the internet our version will contain a high score function based on the number of guesses the player had.

3.1 Reflection

I found it a little bit hard on what to include in the vision since it was described different in the template as to what was in the lecture. According to the template, and what was said in slack from the tutors, the vision was supposed to describe the system being created so this is what I have done. In the template from lecture 3 you were supposed to include "Problem statement" and "Stakeholders". However, I chose to not include these parts because there will be more information about stakeholders in the project plan and the "Problem statement" was hard to describe since this application won't really solve a problem it's just something that has to be done as a part of this course? I did however include some bits from the "Product position statement" when I explained what would differentiate this application from other similar applications.

4 | Project Plan

4.1 Introduction

The purpose of this project plan is to gather all necessary information to plan and control the project being developed during this course. The project plan will describe the approach to develop the Hangman game. This project plan will be used by the project manager for planning and by the project team members to understand what needs to be done. This project plan will describe the different stakeholders and what resources will be available during this project development. There will also be an overall project schedule provided. This is a living document and it will be updated throughout the course.

4.2 Justification

This game is created to fulfil our main clients' goal about having a word guessing game available online for all dog breed enthusiasts and therefor this game will only have one category, with different levels, for the words being guessed and that is different dog breeds.

4.3 Stakeholders

Manager – is responsible for project planning, breaking down the work into parts and assign them to team members, anticipate what problems might arise, and prepare tentative solutions.

Developer/Software engineer – is responsible for developing the planned project.

Tester – is responsible for testing the functionality of the project with manual and unit tests.

End-user – main client who will approve the finished project.

Teachers/tutors (in course 1dv600) – will grade the assignments as well as the finished project.

4.4 Resources

The time budget for this project is 9 weeks, with 20 hour weeks per person.

Personnel available during the development of this project are the manager, one developer/software engineer and one tester. However, these roles are actually all played by one single person.

IDE – Visual Studio Code

Programming Language – JavaScript

4.5 Hard- & Software Requirements

Hardware: Processor Intel® Core™ m3-6Y30 (or similar).

Software: Visual Studio Code v1.30.2 (or newer), Node v8.11.1 (or newer) and npm v6.1.0 (or newer).

4.6 Overall Project Schedule

Start Date	End Date	Deliverables
22/1-19	8/2-19	Project plan, skeleton code (GitHub release)
8/2-19	22/2-19	Updated project plan, design documentation, GitHub release
22/2-19	8/3-19	Updated project plan, test documentation, GitHub release
8/3-19	19/4-19	Updated project plan, updated design documentation, updated test documentation, GitHub release

4.7 Scope, Constraints and Assumptions

4.7.1 Scope

The general idea of the project is to create an ordinary hangman game where the player will be provided with a number of underscore signs, which are equal to the number of letters in a randomly picked word from a predefined list. The player will then guess this word by suggesting letter after letter. The player can have eight faulty guesses at most and for every faulty guess the game will build a part of a man getting hanged. When starting the application the user will be asked to enter a name which will then be used to add in the extra functionality of a high score list, to make this game different from other hangman games available to play online. In the high score list we will present the players name and how many guesses the player had while playing the game. To satisfy our main client this game will also differentiate from other “Hangman” games because the predefined list will only contain different dog breeds. The user will also be able to choose what level of difficulty the word should be.

4.7.2 Constraints

This should not be a very time consuming project, however the time could be a limitation anyways if the time is not well planned. Since we do not have all knowledge about the project from the start, it will make it harder to plan the time.

4.7.3 Assumptions

For this project it is assumed that the end-users know how to use the console to play this application.

4.8 Reflection

I thought some parts of creating the project plan was a bit difficult since there was not that much of a description in the project plan and I did not find that there was that much of information in the lecture either. I did look through the book and found some information that combined with the information in the template made it a little bit easier. I felt unsure what was supposed to be written under “Resources” and maybe we get more knowledge on what resources will be needed in the future assignments. Also it seems unnecessary to include “costs” under resources because I do not believe there will be any costs.

5 | Iterations

5.1 Iteration 1

Task	Estimated Time	Responsible
Create GitHub repo	15min	Developer
Write project documentation	4h	Manager
Implementation – idea and skeleton code	3h	Developer

5.2 Iteration 2

Task	Estimated Time	Responsible
Plan for Assignment 2	0.5h	Manager
Update Use Case Diagram	0.5h	Manager
Write UC2 – Play Game	1h	Manager
Write UC for extra features	2h	Manager
Create state machine for Play Game	2h	Manager
Implement the modelled behaviour	8h	Developer
Create class diagram from implementation	2h	Manager
Update Project Plan	1h	Manager

5.3 Iteration 3

Task	Estimated Time	Responsible
Plan for Assignment 3	0.5h	Manager
Learn about JavaScript Test Framework “Mocha”	3h	Tester
Create Test Plan	2h	Manager
Create and execute Manual Test Cases	3h	Tester
Create and execute Automated Unit-tests	7h	Tester
Restructure code for testing	3h	Developer

5.4 Iteration 4

Task	Estimated Time	Responsible
Plan for Iteration 4	1h	Manager
Implementation – add more/new words with level categories	2h	Developer
Implementation – fix: word cannot contain same letter	2.5h	Developer
Implementation – dog breeds with “two words”	2h	Developer
Update Project Plan, replaced admin with levels	3h	Manager
Update Use Case Model	2h	Manager
Update State Machine	3h	Manager
Update Class Diagram	2h	Manager
Implementation – add levels	6h	Developer
Implementation – add high score functionality	3h	Developer
Update Test Plan	4h	Manager
Update/create and execute Manual Test Cases	5h	Tester
Update Automated Unit-tests	4h	Tester
Implementation – fix appearance in console	1h	Developer

6 | Risk Analysis

6.1 List of risks

Risk	Probability	Effects
The time required to develop the software is underestimated	High	Serious
Team members are ill and unavailable at critical times	Moderate	Serious
Facts needed to complete the system architecture are not known, or known imprecisely	Moderate	Catastrophic
Changes to requirements that require major design rework are proposed	Low	Serious
Lack of definition in the system requirements	High	Catastrophic

6.2 Strategies

Risk	Strategy
Underestimated development time	Exaggerate when doing time estimation and plan with “slack”.
Staff illness	Make sure team members have understanding about each other’s job.
Lack of knowledge	Investigate what facts will be needed to complete the system at the beginning of the project.
Requirements changes	Measure the impact in changing the requirements and explain the consequences.
Lack of definition	Try to get as much information about the requirements at the start of the project.

6.3 Reflection

There were many examples in the book about different risks, but many of these risks I excluded since I believe this is not a huge project with no real budget (except for time) and it’s only one person working all roles as well as I’m being my own “customer” also. Therefore risks like “impossible to recruit staff” and “organization is restructured” is not necessary to include here, but I wrote down the risks I thought was relevant for this project. I also used some of the risk examples from the lecture. When it came to “Strategies” I tried to come up with my own strategies and not use exactly what was said in the book.

7 | Time log

7.1 Assignment 1

Date	Task	Estimated Time	Actual Time
24/1-19	Create GitHub repo	15min	15min
5/2-19	Create project documentation	4h	5.5h
7/2-19	Start implementation	3h	1h 15min
12/3-19	Update documentation for re-take	1.5h	2.5h

7.1.1 Reflection

When it comes to creating the project documentation I did not include the time it took to watch the lectures and read the pages in the book, however sometime I had to go back and watch some parts of the lecture or read about something in the book again when I was unsure on what to write and that time is included and I think that is why there is a time difference because I was often unsure on what to write because some parts in the template was not that well describe according to me. I did add extra time to implementation because when I estimated the time I was unsure on how much code we were supposed to implement at this stage and therefore there is a big time difference.

7.2 Assignment 2

Date	Task	Estimated Time	Actual Time
13/1-19	Plan for assignment 2	0.5h	45min
13/1-19	Update Use Case Diagram	0.5h	1h
13/1-19	Write Use Cases	3h	4.5h
18/1-19	Create state machine for Play Game	2h	4h
20-21/1 -19	Implement the modelled behaviour	8h	12h
21/1-19	Create class diagram from implementation	2h	1h
13/1-19	Update Project Plan	1h	45min
	Re-take:		
25/3-19	Update Use Case Diagram	1h	0.5h
25/3-19	Update Class Diagram	1h	1h
26/3-19	Update State Machine	3h	3.5h

7.2.1 Reflection

I did not include how long time it took to read chapters in the book or watch lectures, because I did not know this was supposed to be included until it was said in a lecture and then I had not estimated time for that so then I found it redundant to include. It took some time to create the state machine because I was unsure on what to and changed my mind a couple of times. The implementation took longer because I had to change some stuff around because I could not get the remainingTries to function as I wanted and when I got that to work there were other things that got messed up. I also wanted to spend some time on cleaning up my code so it would look a little bit neater and easier to follow along. Since I had all of my code in one file, there was not that much to do in the class diagram. What took time is that I was unsure if I should include the app.js file and tried to do some research about this. However, I could not find any information and eventually I just decided to include it since it does use the other file.

7.2.2 Reflection re-take

I started with updating the use case diagram according to feedback. I did get an OK on my class diagram, but since my code structure has changed a lot since handing in this assignment the first time and I felt like I would need a new class diagram for assignment 4, I chose to update my class diagram also for the re-take. I felt like I did not get that much feedback on the state machine other than that it was not correct and I thought it was very hard to understand how to create the state machine so I had to do more research on creating the state machine and then I created a new state machine for "Play Game", which I hope is working. What I got from reading about state machines is kind of that the boxes are what the application does and the describing text above the lines, connecting these boxes, is what the user does.

7.3 Assignment 3

Date	Task	Estimated Time	Actual Time
27/2-19	Plan for Assignment 3	0.5h	0.5h
4/3-19	Learn about JavaScript Test Framework "Mocha"	3h	4.5h
1/3-19	Create Test Plan	2h	2.5h
6/3-19	Create and execute Manual Test Cases	3h	4h
8/3-19	Create and execute Automated Unit Tests	7h	6.5h
7/3-19	Restructure code for testing	3h	5h

7.3.1 Reflection

I started out with looking into which JavaScript Test Framework I should use and chose "Mocha" and tried to learn as much as possible about that Framework since I have not written any automated unit tests before. I chose to not implement any additional features during this iteration; instead I restructured a whole lot of code because I did not like how most of it looked and worked. Also I did not have that many functions that returned something and from what I could understand the functions needed to return something for me to be able to test them and therefor I needed to refactor some code to make it easier to create unit tests. The additional features will instead be implemented during next iteration.

I thought it was very hard to create the unit tests and therefor they are not very advanced. This was the first time I learned about unit testing and creating them and also using mocha so I think my unit tests work fine for those reasons.

The manual test cases took a little bit longer to execute because I wanted to try multiple times to see if the test case always succeeded or if it could fail. If the test case could fail sometimes I chose to write this as a comment to that test and mark it as a success. I was unsure how to present the results if the test case was executed multiple times.

I did not present my results as in the "greeter" example because that not how I interpreted the assignment instructions.

7.4 Assignment 4

Date	Task	Estimated Time	Actual Time
29/3-19	Plan for Iteration 4	1h	1h
30/3-19	Implementation – add more/new words with level categories	2h	1h 45min
30/3-19	Implementation – fix: word cannot contain same letter	2.5h	3h

30/3-19	Implementation – dog breeds with “two words”	2h	3h
31/3-19	Update Vision and Project Plan, replaced admin with levels	3h	2h
3/4-19	Update Use Case Diagram	1h	2.5h
3/4-19	Update Use Cases	3h	2h 45min
4/4-19	Update State Machine	3h	3.5h
12/4-19	Update Class Diagram	2h	1.5h
7/4-19	Implementation – add levels	6h	3h
7/4-19	Implementation – add high score functionality	3h	6.5h
8/4-19	Implementation – draw hangman	3h	4h
10/4-19	Update Test Plan	4h	2h
10/4-19	Update/create and execute Manual Test Cases	5h	7h
11/4-19	Update Automated Unit-tests	4h	2h
12/4-19	Implementation – fix appearance in console	1h	2h

7.4.1 Reflection

I decided to change some things in the vision and project plan because I did no longer want to implement the “Admin menu options”, instead I wanted to implement that the user could choose a level when playing the game.

I had to redo the use case diagram many times because I was unsure if I was supposed to include “set name” and “set level”, that’s why it took a bit longer time. I then thought that I would have use cases and manual test cases for these things and therefor I think they should be included in the use case diagram also. I was also unsure if all arrows should come from “Start game”, but I thought that the “set level” should be included from “Play game” because it’s not a use case until you have chosen to play a game, but I am still a bit unsure on how the arrows should work in a use case diagram.

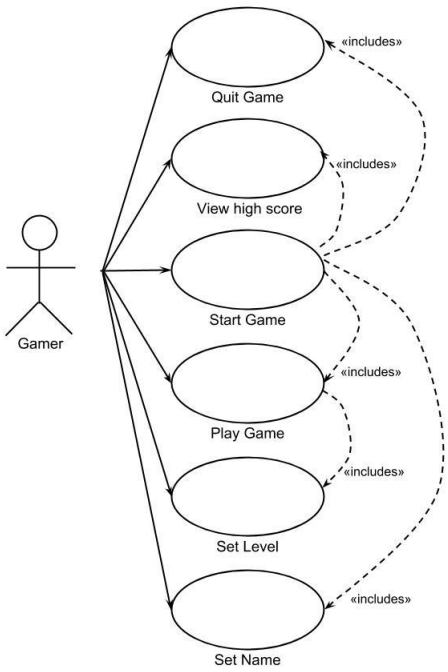
Implementing the high score took a bit longer then I thought because it did not work as I wanted it to at first. I wanted to save the high scores in a json file, but when I saved in new high scores the old ones was deleted. So I struggled with that for some time, but then I got I working as I wanted to. The only thing that did not work as I wanted to with high score was that the game ends after watching high scores in the terminal; I wanted to be able to continue making choices in the menu and continue playing the game.

It took me some time to update and create new manual test cases because I wanted to have a few test cases for each use case. I also executed some of the manual test cases multiple times to make sure they always had the same outcome and did not change from time to time, in that case a manual test case should be a fail.

Did not know how I was supposed to hand-in this iteration, but I chose to include the design documents in this file with the vision, project plan and iteration planning. Then I created a new document for the testing to not make this document too big and hard to navigate.

8 | Use Case Model

8.1 Use Case Diagram



8.2 Use Cases

UC1 – Start Game

Precondition: none.

Post condition: the game menu is shown in the console.

Main scenario

1. Starts when the user wants to begin a session of the hangman game.
2. The system asks for the user nickname.
3. The user logins with a nickname.
4. The system presents the main menu with the options to play game, view the high score list, and quit the game
5. The user makes the choice to play a game.
6. The system asks the user to choose a level (see Use Case 2).

Alternative scenarios

5.1 The user makes the choice to quit the game.

1. The system quits the game (see Use Case 4).

6.1 Invalid menu choice.

1. The system presents an error messages.
2. Go to 4.

UC2 – Choose a level

Precondition: the game is running in the terminal.

Post condition: the game is started.

Main scenario

1. Starts when the user has started the game.
2. The system asks the user to choose a level.
3. The user chooses a level.
4. The system starts the game (see Use Case 3).

Alternative scenarios

3.1 The user does not choose a level.

1. The system returns to its previous state.

UC3 – Play Game

Precondition: the game is running in the terminal.

Post condition: the game is finished.

Main scenario

1. Starts when the user has started the game.
2. The system randomly picks a word and presents the number of letters. The system will also show number of guesses remaining and an option for quit the game.
3. The user inputs a correct letter.
4. The system exchanges the underscores for the correct letter and presents number of guesses remaining.
5. Repeat step 3-4 until the word is completed.
6. The system displays “Congratulations!”, and then presents the main menu (see Use Case 1).

Alternative scenarios

3.1 The user makes the choice to quit the game before completing the word.

1. The system goes back to main menu (see Use Case 1).

3.2 The user inputs a faulty letter.

1. The system reduces number of guesses remaining and displays it to the user.
2. Go to 3.

5.1 The user runs out of number of guesses and lose the game.
1. The system displays “You Lost!”, and then presents the main menu (see Use Case 1).

UC4 – Quit Game

Precondition: the game is running in the terminal.

Post condition: the game is terminated.

Main scenario

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

Alternative scenarios

4.1 The user does not confirm.
1. The system returns to its previous state.

UC5 – View high score list

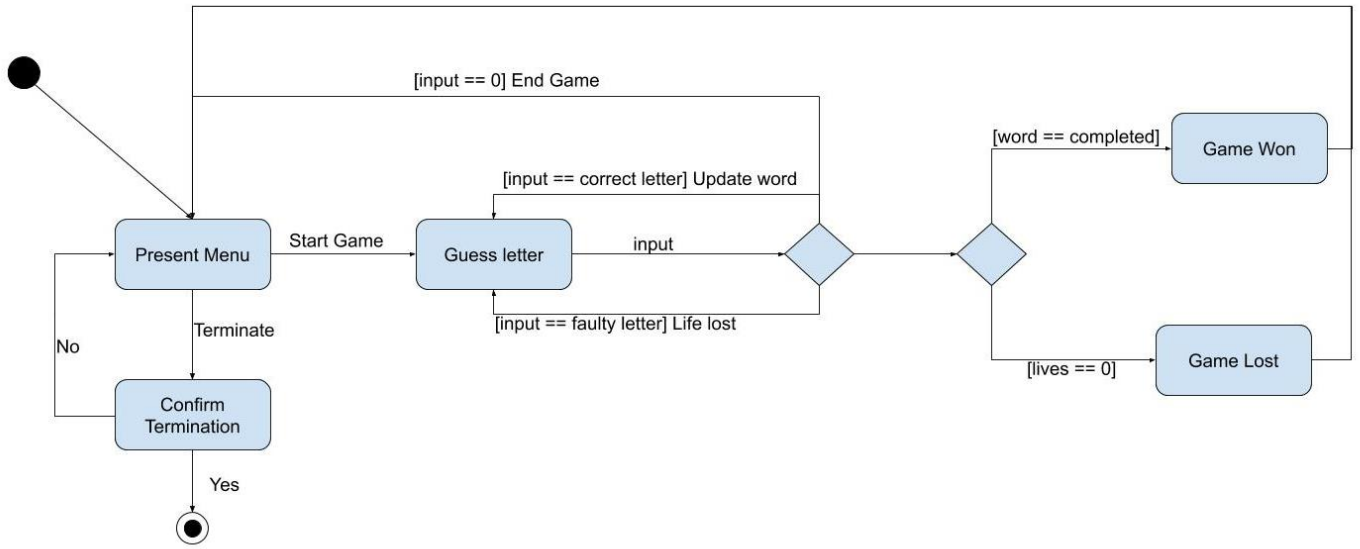
Precondition: the game is running in the terminal.

Post condition: the high score list is shown in the terminal.

Main scenario

1. Starts when the user wants to view the high score list.
2. The system presents the saved high score list, and then presents the main menu (see Use Case 1).

9 | State Machine



10 | Class Diagram