SOFTWARE DEVELOPMENT PROJECT

LNU 1dv600 Programvaruteknik

NAME: Niall Thurrat

Date Started: 23 Jan 2019

DOCUMENT VERSION: 1.1 (Iteration 2)

Contents

1 Revision History

Date	Version	Description	Author
7/2/19	1.0	~~Basic structural code of hangman game~~ Initial Project plan fully completed with following exceptions: sections 5.2-5.4 (Iteration 2-4) to be developed in later revisions, sections 7.2-7.3 (Timelogs) to be completed in future revisions	Niall Thurrat
21/2/19	1.1	Project Plan has had following changes: - Section 1: revision history amended - Section 3: System Vision amended - Section 4.1.5 Hard- and Software Requirements has had 2 unnecessary items removed - Section 5.2 Iteration 2 has been developed - Section 7.2 Time log Iteration 2 has been added	Niall Thurrat

2 General Information

Project Summary	
Project Name	Project ID
Hangman	#1
Project Manager	Main Client
Niall Thurrat	LNU teachers
Key Stakeholders	
LNU 1dv600 Software Engineering teachers Project Manager – Niall Thurrat	
Executive Summary	
The game "Hangman" will be developed as a console application using JavaScript and displayed in a text based fashion in a computer console for a single user to play, and released on Github. The project will have four iterations and the application will therefore be released 4 times incrementally. Deliverables will include the four releases of the application, as well as this Project Plan document which will be reviewed and developed at the end of each iteration, and additional documentation regarding modelling and testing during the second and third iteration respectfully.	

3 Vision

3.1 System Vision

SUMMARY

Hangman is a word guessing game where the user must guess letters of a predefined word (nouns from the English dictionary will be used) with a view to finally guessing the whole word. With each wrong guess and a piece will be added to a picture of a hanging man which will be constructed using common keyboard characters. If all the pieces which complete the hangman are added to the picture, the game is lost. The player will win the game if they guess the correct word or complete the word by guessing all the letters before the whole hangman is complete.

FUNCTIONALITY

~~The game will be developed in a text based fashion and presented on a console terminal for a single user to play. On entering 'npm start', the game will begin and 2 things will happen 1) a secret noun with at least 6 letters will be randomly selected and 2) the user will be asked to enter a username. Once the username is entered, the terminal will present a number of underscore characters each representing a letter of the randomly chosen secret noun, as well as a menu where the user can either choose to guess a single letter or the whole word. A help message will always be displayed to provide guidance to the player about, for example, the rules or how many wrong tries they have remaining.~~

The game will be developed in a text based fashion and presented on a console terminal for a single user to play. When the user starts the application they will see a main menu which will give them the option to play the game, see a high score chart, change the game difficulty, read the rules or quit the application. When the gamer starts a new game, two things will happen: 1) a secret noun will be randomly selected and 2) the user will be asked to enter a username. Once the username is entered, the terminal will present a number of underscore characters each representing a letter of the randomly chosen secret noun, as well as a menu where the user can either choose to guess a single letter or the whole word. A help message will always be displayed to provide guidance to the player about, for example, the rules or how many wrong tries they have remaining.

~~If the player correctly guesses a letter which exists in the word, that letter is revealed (multiple times if appropriate) and the hangman picture remains the same. If a guessed letter is not a part of the secret word, a piece of the hangman will be added and the letter will be added to a list of 'Wrong Letters'. Likewise, a wrongly guessed word will see the hangman develop and will show in a 'Wrong Words' list. The application will make use of data persistence to enable progression through the game and a high-score chart will be used to show the top 5 results. A player's final winning score will be a number representing how few wrong guesses they have made when they win the game.~~

If the player correctly guesses a letter which exists in the word, that letter is revealed (multiple times if appropriate) and the hangman picture remains the same. If a guessed letter is not a part of the secret word, a piece of the hangman will be added and the letter will be added to a list of 'Wrong Letters'. Likewise, a wrongly guessed word will see the hangman develop and will show in a 'Wrong Words' list.

The application will make use of data persistence to enable progression through the game and to store the highest game scores. The high score chart which is accessible from the main menu will show the top 5 results.

3.2 Reflections on Creating a Vision Document

I've structured my system vision document with a *Summary* section to help stakeholders understand the system at a glance, as well as a *Functionality* section to give further detail which will be used as guidance (by myself, the development team) to create a more detailed breakdown of necessary project tasks to be completed in each application iteration (see chapter 5 regarding project iterations).

I consider vision documents to be an important method to allow stakeholders to understand an application better as well as to help project workers gain a common understanding about what it is they must create. Vision documents are an effective way to outline key functionality which can later be used to identify project tasks, without being as time consuming as a full specification requirements document.

4 Project Plan

4.1.1 Introduction

The primary objective of this project is to create a software application called “Hangman”, based on the classic game of the same name. Hangman is a word guessing game where the user must guess letters of a predefined secret word with a view to finally guessing the whole word. This Hangman application will be developed using JavaScript and displayed in a text based fashion on a console terminal for a single user to play.

4.1.2 Justification

The primary purpose of developing this application is to demonstrate my understanding and ability to utilize and document a software engineering process as a requirement for my Software Engineering course (1DV600) at Linnaeus University. A secondary reason for the creation of the application is to develop my knowledge of the core elements of the course and to further develop my programming skills and experience in relation to that knowledge.

4.1.3 Stakeholders

- System end-users: The people who will use the finished system (The Software Engineering teachers at Linnaeus University, Niall Thurrat, the public)
- Project manager: Coordinates who does what and when during the project and makes sure that this plan is followed, that tasks are completed on time, and that risks and delays are managed (Niall Thurrat)
- System architect: Responsible for the structuring and design of the system (Niall Thurrat)
- Client engineer – responsible for implementing the plan and coding the application (Niall Thurrat)
- Tester: responsible for ensuring the system functions as it should without bugs (Niall Thurrat)

4.1.4 Resources

People (support and advice)	- Teachers (MyMoodle, slack, tutorials, QA) - Other students (slack)
Info, guidance, project brief and tutorials	Instructions and advice provided on course’s MyMoodle page, internet/google
Literature	Software Engineering (10th Edition) by Ian Sommerville
Version control and project repository	Github
Project task timer	Toggl timer app tool
Development tools and environment (inc. app dependencies)	Node.js, npm, visual studio code

4.1.5 Hard- and Software Requirements

Hardware	- Computer with internet

Software	- Node.js and npm - Visual Studio Code IDE

4.1.6 Overall Project Schedule

DATE	DELIVERABLE
23 January 2019	- Project begins
8 February 2019	- Iteration 1 of Hangman app - Project Plan document
21 February 2019	- Iteration 2 of Hangman app - UML documentation
8 March 2019	- Iteration 3 of Hangman app - Test documentation
Week 12 (22 March?)	- Iteration 4 of Hangman app (final app completion)

4.1.7 Scope, Constraints and Assumptions

The scope of the project is to create an application which can be played in a computer console and will therefore not have a GUI. The files will be sourced in a github repository and available to any member of the public to play on their home computer. The game will be completely generated by common keyboard characters and thus will not contain any actual image files. Furthermore sound will not be used. The game will include a high-score chart, username entry, a difficulty setting and will be able to fetch a random noun from the internet.

A major constraint will be time, given I do not have much more than 20 hours a week to devote to the project and it appears that more than half of this time will be dedicated to reading and tutorials. My lack of knowledge and experience in writing terminal applications is also a constraint.

I assume that end-users will have sufficient knowledge of how to use the console to install the app dependencies and play the game.

4.2 Reflection on Creating a Project Plan

The Project Plan provides project stakeholders with an essential overview of the purpose of the project, interested parties, resources involved and the most important dates during the duration of the project. Creating the document has helped me give further consideration to the resources and time that is available to me, and has helped me clarify in simple terms the purpose for this project.

5 Iteration Plan

5.1 Iteration 1

DATE (start - due date)	TIME ESTIMATE (HRS)	TASK / Sub-Task	RESOURCES REQUIRED
WEEK 4 (21/1 - 27/1) – 10 EFFORT HRS			
23/1	1.75	LEARNING: Lecture - intro	MyMoodle
23/1 – 27/1	8.25 ->	LEARNING: reading literature (ch 2, 3, 22, 23)	book
WEEK 5 (28/1 – 3/2) – 20 EFFORT HRS			
28/1 – 30/1	<- 3.75 (12 hrs total)	LEARNING: reading literature (ch 2, 3, 22, 23)	book
28/1 – 30/1	3.5	LEARNING: pre-recorded lecture vids (x 2)	MyMoodle
30/1	1.75	LEARNING: QA - processes	YouTube, teacher
30/1 ->	7.5 ->	PLANNING: Start Project Plan document	MS Word, book, MyMoodle
31/1 – 1/2	0.5	EXAM: Theme 1 online exam	MyMoodle
2/2 – 3/2	3	LEARNING: Have a think and google about console applications and how mine might function and be structured	Internet, google, past program exercises
WEEK 6 (4/2 – 10/2) – 20 EFFORT HRS			
<- 4/2	<- 5 (12.5 hrs total)	PLANNING: continue and finish 1st draft of Project Plan doc	MS Word, book, MyMoodle
4/2 – 6/2	9	LEARNING: reading literature (ch 4, 5, 20)	book
6/2	1.75	LEARNING: Tutoring – modelling	teachers, MyMoodle
4/2	3	PLANNING/DESIGN: create a flow chart to predict as much basic and additional game functionality as possible from start to finish of application to help me populate this Iteration Plan with project tasks.	MS Word
4/2 – 8/2	1	IMPLEMENTATION: create development environment using IDE and node.js. Create probable modules and structure project directory, export and import as necessary and create basic functions/objects in each which print test messages to console.	vsc, node,
4/2 – 8/2	0.25	IMPLEMENTATION: Do release on Github/MyMoodle	github,MyMoodle
ITERATION 1 DEADLINE: 23:55 8/2			

5.2 Iteration 2

DATE (start - due date)	TIME ESTIMATE (HRS)	TASK / Sub-Task	RESOURCES REQUIRED
WEEK 7 (11/2 – 17/2) – 20 EFFORT HRS			
11/2 – 13/2	1	LEARNING: reading literature (ch 7.1)	book
11/2 – 13/2	3	LEARNING: UML practical tutorials including identifying and downloading appropriate software	MyMoodle, UML tools
13/2	1.75	LEARNING: Tutoring session – UML	teachers, MyMoodle, youtube
13/2	0.25	Reading assignment 2 brief	Github wiki
13/2	0.5	PLANNING: develop Iteration 2 tasks and time predictions in Project Plan document	MS Word, MyMoodle
13/2 – 17/2	2	MODELLING: create Use Case diagram of system as described in the project Vision with UML	UML tool, github wiki, book
13/2 – 17/2	2	MODELLING: create fully dressed use case for basic "Play Game"	MS Word, github wiki, book
13/2 – 17/2	2	MODELLING: create State Machine Diagram for basic "Play Game"	UML tool, github wiki, book
13/2 – 17/2	0.5	IMPLEMENTATION: design a basic game banner image and 8 placeholders of basic hangman image (prints to console), return placeholders from imageGenerator.js	vsc, node.js, google
13/2 – 17/2	2	IMPLEMENTATION: create a skeleton menu which is printed to console from Hangman.js (write from scratch or use npm package/find re-useable code)	vsc, node.js, npm, google
13/2 – 17/2	0.25	IMPLEMENTATION: add 'play new game' option to start menu and configure so that it uses imageGenerator.js to produce banner and hangman	vsc, node.js
13/2 – 17/2	0.5	IMPLEMENTATION: develop wordGenerator.js module which returns a randomly selected noun from a hardcoded list of 10 words. Develop wordBoard.js module to call wordGenerator.js module for a new word and return	vsc, node.js
13/2 – 17/2	0.5	IMPLEMENTATION: develop messageGenerator.js module to return a message from a list of predefined game messages (welcome, unlucky, good guess, game over, etc)	vsc, node.js

13/2 – 17/2	0.25	IMPLEMENTATION: develop Hangman.js to log new game screen to console (inc. banner, hangman base pic, wordBoard, Message, Menu)	vsc, node.js
13/2 – 17/2	3.5 ->	IMPLEMENTATION: develop Hangman.js/wordBoard.js logic to deal with guessed letters and reprint new game state to console	vsc, node.js
WEEK 8 (18/2 – 24/2) – 30 EFFORT HRS			
18/2-20/2	10	LEARNING: reading literature (ch 6, 7, 15)	book
18/2-20/2	5	LEARNING: pre-recorded lecture vids (x 3)	MyMoodle
20/2	1.75	LEARNING: Tutoring – Design	teachers, MyMoodle
21-22/2	1.5	EXAM: Theme 2 online exam including topic revision	MyMoodle
20/2 – 21/2	<- 2 (5.5 hrs total)	IMPLEMENTATION: finish developing game logic to deal with guessed letters	vsc, node.js
20/2 – 21/2	2	IMPLEMENTATION: use local storage to save game progress	vsc, local storage
20/2 – 21/2	0.5	IMPLEMENTATION: create 'game over' and 'you win!' images (placeholders) and develop Hangman/wordBoard to print to console when game lost or won	vsc, node.js
20/2 – 21/2	0.75	IMPLEMENTATION: add 'quit game' item to menu and develop logic to reset game ('game over' or 'game quit') in Hangman.js	vsc, node.js
20/2 – 21/2	1	IMPLEMENTATION: add 'quit application' to start menu and develop logic to exit (terminate) app in Hangman.js	vsc, node.js
20/2 – 21/2	2	MODELLING: Create a class diagram from implementation	UML tool, github wiki, book
21/2	0.5	RELEASE: Do release on Github/MyMoodle	github, MyMoodle
ITERATION 2 DEADLINE: 12:00 21/2			

5.3 Iteration 3

DATE (start - due date)	**TIME ESTIMATE (HRS)**	**TASK / Sub-Task**	**RESOURCES REQUIRED**
WEEK 9 (25/2 – 3/3) – 20 EFFORT HRS			
25/2	1	PLANNING: revise Project Plan document, add further detail to Iteration 3 Plan	MS word
25/2 – 27/2	3	LEARNING: reading literature (ch 8)	book

25/2 – 27/2	4.5	LEARNING: pre-recorded lecture vids (x 3 + greeter)	MyMoodle
27/2	1.75	LEARNING: Tutoring – Test plan	teachers, MyMoodle
27/2 – 3/3	3	IMPLEMENTATION: IMPORTANT EXTRA FEATURE - add 'enter username' request upon opening application, store username in local storage. add 'change username' to start menu and develop Hangman to change local storage data	vsc, node.js
27/2 – 3/3	6.75	IMPLEMENTATION: IMPORTANT EXTRA FEATURE - improve banner, hangman and overall game design	vsc, node.js
WEEK 10 (4/3 - 10/3) – 20 EFFORT HRS			
6/3	1.75	LEARNING: Tutoring - Test	teachers, MyMoodle
4/3 - 6/3	8	IMPLEMENTATION: IMPORTANT EXTRA FEATURE - add high scores chart functionality	vsc, node.js
4/3 - 8/3	10	TESTING: test Hangman application	testing tools
8/3	0.25	IMPLEMENTATION: Do release on Github/MyMoodle	github, MyMoodle
ITERATION 3 DEADLINE: 23:55 8/3			

5.4 Iteration 4

DATE (start - due date)	TIME ESTIMATE (HRS)	TASK / Sub-Task	RESOURCES REQUIRED
WEEK 11 (11/3 – 17/3) – 20 EFFORT HRS			
11/3	1	PLANNING: revise Project Plan document, add further detail to Iteration 4 Plan re final additional features	MS word
11/3 – 13/3	2	MODELLING: model final extra features	MyMoodle, UML tool
13/3	1.75	LEARNING: QA - project	teachers, MyMoodle
1-15/3	0.5	EXAM: Theme 3 online exam	MyMoodle
13/3 – 17/3	10	IMPLEMENTATION: DESIRABLE EXTRA FEATURE - add functionality to wordGenerator.js which fetches a random noun from somewhere online, e.g. google	vsc, node.js, internet/ google
WEEK 12 (18/3 – 24/3) – 20 EFFORT HRS			
20/3	1.75	LEARNING: QA - project	teachers, MyMoodle
18/3 – 22/3	5	IMPLEMENTATION: DESIRABLE EXTRA FEATURE - add difficulty levels menu option and logic	vsc, MyMoodle

18/3 – 22/3	5	TESTING: test final extra features and app as whole??	testing tools
22/3	0.25	IMPLEMENTATION: Do release on Github/MyMoodle	github, MyMoodle
ITERATION 4 DEADLINE: 22/3 ???			

6 Risk Analysis

6.1 List of Risks

RISK (AFFECTS): DESCRIPTION	PROBABILITY	EFFECTS
Loss of project personnel – permanent (project): I leave the project before it finishes	Low	Catastrophic
Loss of project personnel – temporary (project): I am ill at critical times in the project, or cannot commit as much time due to unforeseen circumstances	Moderate	Serious
Programming skills inadequate (project and product): Project developer does not have necessary skills required to develop planned system	High	Serious
Size underestimate (project and product): The size of the application is underestimated	Moderate	Serious
Time underestimate (project and product): The time required to develop the software is underestimated	High	Serious
Requirements changes (project): A larger number of changes to the system requirements than anticipated	Low	Tolerable
Software tool underperformance (product): Software tools that support the project do not perform as anticipated	Low	Tolerable

6.2 Strategies

RISK	STRATEGY (type)
Loss of project personnel – permanent	In the event that I cannot complete the project (and am still alive) I will contact the course responsible and program admin re decisions taken, save any project documentation for possible future use, and inquire about possible opportunities to begin the project again in a future course. (contingency plan)
Loss of project personnel – temporary	- Avoid contracting illness during project duration (e.g. eating healthily, taking supplements to ensure adequate vitamin and nutrient intake, using alcohol gel hand sanitizer before eating and constantly washing hands, avoiding unnecessarily leaving apartment). (avoidance strategy) - Plan to have essential project requirements and important extra features completed by the third iteration and use the fourth iteration to add desirable (non-essential) extra features. This way tasks which cannot be completed during iteration 1-3 can be moved to the final iteration at the expense of unnecessary feature completion if time is lost due to unforeseen circumstances. (minimization strategy)

	- Ask my employer for time off to devote additional hours to working on the project and thus increase the effort (person-months) initially assigned to the project. (minimization strategy)
Programming skills inadequate	Allow extra time for learning when estimating project tasks in the Iteration Plan, especially for aspects of coding which I am unsure about (minimization strategy)
Size underestimate	- Plan to have essential project requirements and important extra features completed by the third iteration and use the fourth iteration to add desirable (non-essential) extra features. This way tasks which cannot be completed during iteration 1-3 can be moved to the final iteration at the expense of unnecessary feature completion if time is lost due to the application being larger than anticipated. (minimization strategy) - Ask my employer for time off to devote additional hours to working on the project and thus increase the effort (person-months) initially assigned to the project. (minimization strategy)
Time underestimate	- Plan to have essential project requirements and important extra features completed by the third iteration and use the fourth iteration to add desirable (non-essential) extra features. This way tasks which cannot be completed during iteration 1-3 can be moved to the final iteration at the expense of unnecessary feature completion if time is lost due to the application taking longer to develop than anticipated. (minimization strategy) - Ask my employer for time off to devote additional hours to working on the project and thus increase the effort (person-months) initially assigned to the project. (minimization strategy)
Requirements changes	- This is a low probability as the essential application requirements are known from the start of the project and will not change. It is possible that desirable additional features may change the requirements, or that I have misunderstood the project requirements in the brief to a certain degree. Any effect should be minimized by dropping non-essential features from the final iteration to accommodate for extra time and resources necessary to cope with changes. I may ask my employer for time off also, to devote more hours to the project (minimization strategy) - Project instructions should be triple-checked to ensure understanding of system requirements before designing and implementing system. (avoidance strategy)
Software tool underperformance	- Try to use software which I am familiar with (when possible) to complete planned tasks in the project. (avoidance strategy) - Use slack and other communication channels to discuss software tool underperformance and seek suggestions for alternatives from other student and teachers (minimization)

6.3 Reflections on Risk Analysis

As this project is quite small with an eight week duration (approx. one person-month of effort) and one project team-member, there is not a large list of moderate-high risks to consider and, as a consequence, all but one risk have consequences for the project. The risks highlight that due to having a one-man team, the effects of a number risks which might delay the project are categorised as serious.

As a minimizing strategy to address the effects of four risks in the list, I have decided to plan to have all essential tasks regarding system requirements (as well as important non-essential features) complete by the third iteration, allowing desirable extra features to be implemented on the final iteration. This allows flexibility to move essential tasks to the final iteration, if for some reason they cannot be completed on schedule. Writing the document also made me realise that a viable risk minimization strategy is to increase the project effort (person-months) to ensure timely completion of the project (i.e. ask for time off work).

7 Time Log

7.1 Iteration 1 Time Log

DATE (start - due date)	TIME ESTIMATE (HRS)	TASK / Sub-Task	TIME TAKEN (HRS)
WEEK 4 (21/1 - 27/1) – 10 EFFORT HRS			
23/1	1.75	LEARNING: Lecture - intro	as estimated
23/1 – 27/1	8.25 ->	LEARNING: reading literature (ch 2, 3, 22, 23)	total below
WEEK 5 (28/1 – 3/2) – 20 EFFORT HRS			
28/1 – 30/1	<- 3.75 (12 hrs total)	LEARNING: reading literature (ch 2, 3, 22, 23)	13
28/1 – 30/1	3.5	LEARNING: pre-recorded lecture vids (x 2)	as estimated
30/1	1.75	LEARNING: QA - processes	1.5
30/1 ->	7.5 ->	PLANNING: Start Project Plan document	total below
31/1 – 1/2	0.5	EXAM: Theme 1 online exam	as estimated
2/2 – 3/2	3	LEARNING: Have a think and google about console applications and how mine might function and be structured	2
WEEK 6 (4/2 – 10/2) – 20 EFFORT HRS			
<- 4/2	<- 5 (12.5 hrs total)	PLANNING: continue and finish 1st draft of Project Plan doc	14.75
4/2 – 6/2	9	LEARNING: reading literature (ch 4, 5, 20)	10
6/2	1.75	LEARNING: Tutoring – modelling	as estimated
4/2	3	PLANNING/DESIGN: create a flow chart to predict as much basic and additional game functionality as possible from start to finish of application to help me populate this Iteration Plan with project tasks.	as estimated
4/2 – 8/2	1	IMPLEMENTATION: create development environment using IDE and node.js. Create probable modules and structure project directory, export and import as necessary and create basic functions/objects in each which print test messages to console.	1.5
4/2 – 8/2	0.25	IMPLEMENTATION: Do release on Github/MyMoodle	as estimated
ITERATION 1 DEADLINE: 23:55 8/2			

7.2 Iteration 2 Time Log

DATE (start - due date)	TIME ESTIMATE (HRS)	TASK / Sub-Task	TIME TAKEN (HRS)
WEEK 7 (11/2 – 17/2) – 20 EFFORT HRS			
11/2 – 13/2	1	LEARNING: reading literature (ch 7.1)	1.25
11/2 – 13/2	3	LEARNING: UML practical tutorials including identifying and downloading appropriate software	2.5
13/2	1.75	LEARNING: Tutoring session – UML	2.25
13/2	0.25	Reading assignment 2 brief	0.75
13/2	0.5	PLANNING: develop Iteration 2 tasks and time predictions in Project Plan document	0.75
13/2 – 17/2	2	MODELLING: create Use Case diagram of system as described in the project Vision with UML	3.25*
13/2 – 17/2	2	MODELLING: create fully dressed use case for basic "Play Game"	4.5*
13/2 – 17/2	2	MODELLING: create State Machine Diagram for basic "Play Game"	2.5
13/2 – 17/2	0.5	IMPLEMENTATION: design a basic game banner image and 8 placeholders of basic hangman image (prints to console), return placeholders from imageGenerator.js	0.25
13/2 – 17/2	2	IMPLEMENTATION: create a skeleton menu which is printed to console from Hangman.js (write from scratch or use npm package/find re-useable code)	8**
13/2 – 17/2	0.25	IMPLEMENTATION: add 'play new game' option to start menu and configure so that it uses imageGenerator.js to produce banner and hangman	as estimated
13/2 – 17/2	0.5	IMPLEMENTATION: develop wordGenerator.js module which returns a randomly selected noun from a hardcoded list of 10 words. Develop wordBoard.js module to call wordGenerator.js module for a new word and return	0.25
13/2 – 17/2	0.5	IMPLEMENTATION: develop messageGenerator.js module to return a message from a list of predefined game messages (welcome, unlucky, good guess, game over, etc)	0.5
13/2 – 17/2	0.25	IMPLEMENTATION: develop Hangman.js to log new game screen to console (inc. banner, hangman base pic, wordBoard, Message, Menu)	as estimated
13/2 – 17/2	3.5 ->	IMPLEMENTATION: develop Hangman.js/wordBoard.js logic to deal with guessed letters and reprint new game state to console	total below

WEEK 8 (18/2 – 24/2) – 30 EFFORT HRS			
18/2-20/2	10	LEARNING: reading literature (ch 6, 7, 15)	7
18/2-20/2	5	LEARNING: pre-recorded lecture vids (x 3)	as estimated
20/2	1.75	LEARNING: Tutoring – Design	2
21-22/2	1.5	EXAM: Theme 2 online exam including topic revision	1
20/2 – 21/2	<- 2 (5.5 hrs total)	IMPLEMENTATION: finish developing game logic to deal with guessed letters	7.25
20/2 – 21/2	2	IMPLEMENTATION: use local storage to save game progress	3
20/2 – 21/2	0.5	IMPLEMENTATION: create 'game over' and 'you win!' images (placeholders) and develop Hangman/wordBoard to print to console when game lost or won	1
20/2 – 21/2	0.75	IMPLEMENTATION: add 'quit game' item to menu and develop logic to reset game ('game over' or 'game quit') in Hangman.js	0.25
20/2 – 21/2	1	IMPLEMENTATION: add 'quit application' to start menu and develop logic to exit (terminate) app in Hangman.js	0.25
20/2 – 21/2	2	MODELLING: Create a class diagram from "Play Game" use case after implementation	1.5
21/2	0.5	RELEASE: Do release on Github/MyMoodle	
ITERATION 2 DEADLINE: 12:00 21/2			

* the modelling of the use case diagram and the fully dressed use case took much longer than expected as a significant amount of time was spent learning how to do these, as well as having to redraw them both due to not reading the instructions properly

** finding the right npm package to use and learning how to implement it took 4 times longer than anticipated. This was always a risk due to my lack of experience and knowledge with node.js, npm packages and coding command line interfaces.