

Software Development Project - 1DV600

Claes Weyde

March 21, 2019

Contents

1	Revision history	4
2	General Information	5
3	Vision	6
3.1	Reflection	6
4	Project Plan	8
4.1	Introduction	8
4.2	Justification	8
4.3	Stakeholders	8
4.4	Resources	9
4.5	Hard- and Software Requirements	9
4.6	Overall Project Schedule	10
4.7	Scope, Constraints, Assumptions	10
4.7.1	Scope	10
4.7.2	Constraints	11
4.7.3	Assumptions	12
4.8	Reflection	12
5	How to play	13
6	Iterations	13
6.1	Iteration 1	13
6.1.1	To-do	13
6.1.2	Time estimates and Time log	14
6.1.3	Reflection	14
6.2	Iteration 2	14
6.2.1	To-do	14
6.2.2	Time estimates and Time log	15
6.2.3	Reflection	15
6.3	Iteration 3	16
6.3.1	To-do	16
6.3.2	Time estimates and Time log	16
6.4	Iteration 4	17
6.4.1	Iteration Plan	17
6.4.2	Risk	18

6.4.3	To-do	18
6.4.4	Time estimates and Time log	19
6.5	Use cases	20
6.5.1	UC 3	20
6.6	Main Scenario	20
6.7	Alternative scenarios	20
6.8	Testing	20
6.8.1	Unit tests	20
6.8.2	Static and dynamic testing	23
6.8.3	TC 3.1 - Moving to managing words menu	24
6.8.4	TC 3.2 - Inputting an incorrect password	24
6.8.5	TC 3.3 - Inputting a correct password	25
6.9	Class diagram	26
6.10	State machine	27
6.11	Reflection	28
7	Risk Analysis	29
7.1	List of risks	29
7.2	Strategies	30
7.2.1	Hardware failure	30
7.2.2	Illness	30
7.2.3	Underestimation of time requirements	30
7.2.4	Too many features required in application	31
7.3	Reflection	31

1 Revision history

Date	Version	Description	Author
05.02.2019	Version 1.0	Skeleton document provided	Claes Weyde
06.02.2019	Version 1.1	Vision, first draft of project plan section	Claes Weyde
07.02.2019	Version 1.2	Project plan finished	Claes Weyde
07.02.2019	Version 1.3	General information finished	Claes Weyde
07.02.2019	Version 1.4	Iteration and Risk analysis	Claes Weyde
14.02.2019	Version 2.0	Detail planning Iteration 2	Claes Weyde
14.02.2019	Version 2.1	Updated time log after UML work	Claes Weyde
21.02.2019	Version 2.2	Updated time log after implementation work	Claes Weyde
21.02.2019	Version 2.3	Updated time log for class diagram	Claes Weyde
21.02.2019	Version 2.4	Reflection writing	Claes Weyde
02.03.2019	Version 3.0	Updated iteration 3	Claes Weyde
17.03.2019	Version 4.0	Planned iteration 4 in detail	Claes Weyde
20.03.2019	Version 4.1	Wrote some test case information	Claes Weyde
21.03.2019	Version 4.2	Finalized the document	Claes Weyde

2 General Information

Project Summary	
Project Name	Project ID
The Marvelous Hanging Man	HM-001
Project Manager	Main client
Claes Weyde	Language learners
Key Stakeholders	
Developing team, main client/end user	
Executive summary	
<p>Developing a text and ASCII-art based game of hangman for the language learner, as well as for the gamer looking for a cognitive challenge.</p> <p>The project will be split up into a planning phase, a modeling and implementation phase, a testing phase and an evolutionary phase.</p> <p>The project is planned to take about 7 weeks and the complete application will be delivered on the 22 of March 2019.</p>	

3 Vision

Hangman re-imagined! Our vision for this project is to create a version of Hangman which will keep the user at the edge of her or his seat. To not only provide entertainment and leisure, but also thought-provoking imagery and themes.

The Hangman game we envision will be a text-based console application where the user will play hangman. The end user we primarily envision is a person learning a new language or trying to increase their grasp of a language which they already know. By making the process fun we believe that learning is facilitated and that the user will have a easier time recalling the material. However, the game is not only aimed at language learners but also to the casual player looking for a bit of fun.

To aid learning and encourage competitiveness (which may be a means for aiding learning) a high-score system should be implemented, giving the user the possibility of competing against him or herself and other users. Furthermore, we also envision the usage of incentives for the user who performs well. The high-score is certainly one, but another may be the possibility of adding new words to the game. Thereby the well-performing user have the ability of evolving the game further. To cater to different users, as well as to allow the avid user to constantly evolve, we also envision the possibility for the player to choose a difficulty.

In short, we aim to deliver a game where fun and learning combine in an intuitive end product!

3.1 Reflection

It is hard to provide a vision at such an early phase and for such a bare-bones project without adding all the bells and whistles I can imagine. It would certainly be possible (and fun) to add more and more functionality in the vision, but somewhere the rubber must meet the road, and my goal is to have to remove as few features as possible when that happens. Therefore I tried to keep the vision as realistic as possible given what time I have at my disposal. Since this is one of the first sections I write I feel that my grasp on the project is tenuous at best, which is why my main goal is to provide a

vision with a high probability of success. Perhaps at the cost of any chances for a bestselling application.

4 Project Plan

4.1 Introduction

The aim of this project is to develop a game of hangman which fulfills as many of the properties specified in the vision above as possible, within the time available for the course.

4.2 Justification

The justification for this application is, at the most superficial level, that it is required in order to pass the course 1DV600. However, a deeper, and most probable more important justification is that the development of this application as well as the documentation will help me become a better developer with a deeper understanding of the project of development, i.e. not just the coding but planning, implementation, testing and (perhaps to a lesser extent) maintenance. Furthermore, the game itself may be justified in this day and age by the (at least anecdotally) reduced spelling capabilities of the population, which may stem from the ubiquitous auto-correct functionality used whenever we write.

4.3 Stakeholders

The stakeholders for this project are listed below.

- Management - The management want a product according to their specifications and budget. The management will evaluate the iterations of the product as well as the end-product with a view to costs and possible revenue.
- Project Manager - The project manager is ultimately responsible for the success of the project. Thus, the project manager will focus on the processes, timetables and workings of the group in order to ensure that the project is successfully completed.
- Developer - The person or team performing the actual work of coding the software. The main focus of the developer is to arrive at an executable application of the game, i.e. a playable hangman application.

- Test lead - the person responsible for the validation of the software. The main focus of the test lead will be to ensure that the software does not contain any bugs and works according to the specifications.
- End user - the person(s) that will use the application. The main perspective of the end user is that the experience of using the application is enjoyable and perhaps efficient (depending on the type of application). Since this project is a game the end user most probably is looking for a fun experience, something which he/she wants to do during their down-time. The end user envisioned here is primarily the language learner looking for a fun software to increase their spelling proficiency.

The author of this document will take on the roles of both project manager, developer and test lead.

4.4 Resources

The resources used to develop this application and documentation are as follows.

- Hardware: a custom desktop computer and a Lenovo laptop computer are used for this project.
- Software
 - Documentation: TeXStudio is used in conjunction with MikTeX in order to produce the documentation.
 - Application: The application is written in Visual Studio Code using Node.
- Time: The time available for the project ranges between 6 and 15 hours/week.

4.5 Hard- and Software Requirements

The hardware requirements for this application will be very low. The application will be guaranteed to run on a computer having:

- Hardware

- At least 1 GB RAM
- CPU having a frequency of 1.1 GHz and higher.
- Software
 - Operating system: Win 7 or greater, OS X 10.7 or greater.

4.6 Overall Project Schedule

Iteration	Deliverable	Date
Iteration 1	Documentation draft 1	08.02.2019
Iteration 1	Skeleton Code	02.08.2019
Iteration 2	Documentation draft 2	22.02.2019
Iteration 2	UML documentation	22.02.2019
Iteration 2	Working code implemented	22.02.2019
Iteration 3	Documentation draft 3	08.03.2019
Iteration 3	TBD (testing)	08.03.2019
Iteration 4	Complete documentation	22.03.2019
Iteration 4	Complete project	22.03.2019

4.7 Scope, Constraints, Assumptions

4.7.1 Scope

The scope of the project will now be delimited by the functional requirements set forth below.

- **UI** The user interface will be text-based, thus no GUI will be developed.
- **Menu** The application will have a starting menu consisting of:
 - **Login** Where the user can log in to use a saved account.
 - **Difficulty** Where the user can set the difficulty level of the game.
 - **High-score** Where the user can see the current high scores.
 - **Manage words** Where a "super user" can remove and add words to the game. The "super user" will be a logged in user having special authorization. Thus, this choice cannot be made by regular users. Alternatively a special password will be required in order to access this menu item.

- **Play** - Where the user can start a new game.
- **Difficulty** The game will have at least three difficulty levels: easy, moderate and hard. The difficulty level chosen will determine the words drawn and the number of incorrect letters the user may write. As a starting point the number of incorrect letters will be 3 for hard, 5 for moderate and 7 for easy. The words drawn may be determined based on the number of letters contained within the word. Alternatively the difficulty of the words may be a property or characteristic of the word which is set when the word is first saved. This property may also be adjustable in the manage words category of the menu.
- **High-scores** The 10 highest score for each difficulty will be saved. The score will be based on the amount of words successfully completed before the user quit or lost the game. The player loses the game when the man is hanged.
- **Timer** A timer will be implemented which keeps track of the time used for each word. The mean may be determined and shown if the game makes it to the high-score. The running time will be shown as the player plays the game.
- **Adding words** If the user achieves a high-score he/she will have the possibility of adding a word to the words of the game.
- **Managing words** The super user (or the user having the right password depending on the implementation) will be able to manage the words by choosing this option in the menu. The user should have the ability to remove words, add words as well as changing the words and their difficulty assessment.
- **Graphics** The hanged man will be shown using ASCII-art.

4.7.2 Constraints

The constraints are itemized below

- **Time** The main constraint will be the time available since most of the team (i.e. me) work full-time. Thus, the time available needs to be managed closely and used efficiently.

- **Knowledge/Skill** The knowledge of the team will be the second constraint which is also heavily intertwined with the time constraint above. Since the time is limited the time it takes to solve the problems which arise, as well as learn new stuff is of importance. Thus, the working knowledge available is a further constraint which will hinder the development.

4.7.3 Assumptions

The end user is assumed to have a working knowledge of computers and using the console for playing. Further assumptions are that the end user is literate as well as able to see since the game will not provide any audio.

4.8 Reflection

When preparing the project plan it soon becomes obvious that it is *hard* to plan a project at a detailed level. The work may initially feel a bit wasted, especially since it requires quite a large amount of time. However, as the project plan has taken shape I feel that it will be quite valuable to revert back to in order to see how the application is measuring up to the plan. Furthermore, the more detailed plans with time scheduling is useful when time is a high-valued and sparse commodity. It will also be interesting to update the document as the work progresses and from the document evaluate my ability to estimate and plan. For me personally it is also very good to learn the skill of planning before performing the work, since I have a habit of jumping right into the coding.

5 How to play

In order to play node must be installed on the platform. Thereafter the player need to open a bash of choice, move to the folder containing the source-code and type "node Hangman.js". Voila!

6 Iterations

6.1 Iteration 1

In this iteration the project plan is to be finished as well as some skeleton code. The specific entities to complete are itemized below.

6.1.1 To-do

- Prepare github repository
- Study theory
 - Study chapters 2-3 and 22-23 in the coursebook.
 - View lectures 1-3.
- Project plan
 - Prepare document skeleton using LaTeX in github.
 - Plan the project with regards to features, functions and constraints.
 - Write all sections as thoroughly as possible given the current information available.
- Code implementation
 - Set up environment; package.json, .gitignore etc.
 - Skeleton code
 - * Player class,
 - * Word class
 - * File handler class.
 - * High-scores class.
 - * Game module, starting point module (app.js).

6.1.2 Time estimates and Time log

Task	Estimate [hrs]	Actual time [hrs]
Prepare github repository	0.5	0.25
Study book	2	3
Lectures	5	3.5
Document skeleton	0.5	1
Plan project	1	2
Write sections	4	5
Setup coding environment	0.25	0.25
Skeleton code	2	1.5
Total time	15.25	16.5

6.1.3 Reflection

I underestimated the time required a bit. Some tasks took a bit longer than expected but some could be finished faster. I will need to adjust my estimation somewhat, adding a bit more time for reading. I will not adjust the actual coding time downwards since this skeleton code required very little in way of thinking, it was mostly boilerplate code.

6.2 Iteration 2

In this iteration the game features will be modeled using UML and thereafter implemented.

6.2.1 To-do

- Study theory
 - Study chapters 4-7, 15 and 20 in the course book.
 - View lectures 4-8.
- Update project document.
- Prepare UML for functionality:
 - Use case model.
 - Fully dressed use case.

- State machine.
- Code implementation.
 - Implement the features modeled using UML.
- Prepare class diagram from implemented code.

6.2.2 Time estimates and Time log

A first estimate was performed during iteration 1. Thereafter a re-estimation was performed when beginning iteration 2. As can be seen, some tasks were not anticipated during the estimation in the first iteration.

Task	Estimate [hrs]	Re-estimate [hrs]	Actual time [hrs]
Study book	4	5	3
Lectures	6	7	4
Document update	1	1	2
Use case model	–	3	2
Fully dressed use case	–	3	3
State machine	–	1	3
UML Total	5	7	8
Coding	5	10	8
Class diagram	–	1.5	1
Total time	21	31.5	34

6.2.3 Reflection

Unfortunately I did not have enough time to implement all my requirements, e.g. the word management and the high-scores will be implemented in the coming iterations if time allows it. Some time had to be taken from the theory study to instead implement code and work on the UML. Thus, in the time log it would appear that I needed less time for studying and lectures than anticipated. In reality I would probably need more time but I had to prioritize the practical work. Even though I did not spend the planned amount of time on the theory it can be seen that my re-estimation was on the low side. Thus, in the next iteration I will need to allocate even more time for the work at hand.

During my work with the class diagram I realized that my game class has to many responsibilities. I plan on refactoring the code and break out some functionality to other entities in the next iteration. The main focus in the next iteration for functionality implementation is to implement a graphical view of the word being solved (showing the placement of the letters in the word). If time permits I would also like to add the word management.

6.3 Iteration 3

In this iteration the implemented features of the application will be validated through testing. Furthermore, some functionality may be implemented here as well. This iteration is described in more depth in the document named "Test.pdf". There an updated time plan can be found.

6.3.1 To-do

- Study theory
 - Study chapter 9 in the course book.
 - View lectures 9 and 10.
- Update project document.
- Prepare testing framework and perform tests.
- Code implementation.
 - Implement testing.
 - Implement possible further features.

6.3.2 Time estimates and Time log

The time estimation is provided in more depth in the document "Test.pdf". In fact, all information regarding this iteration (which was not previously described when producing this document initially) can be found in that document.

Task	Estimate [hrs]	Actual time [hrs]
Study book	1.5	–
Lectures	2	–
Document update	1	–
Testing	8	–
Coding	3	–
Total time	15.5	–

6.4 Iteration 4

After this iteration the game is supposed to be finished. Some additional features may be implemented but the focus will be on a well working application.

6.4.1 Iteration Plan

The purpose of this iteration is to finish the application. The requirements which are not finished yet are detailed below.

- High-scores
- Graphics (ascii-art)
- Timer
- Login
- Super-user

Since the allotted time will not be enough for finishing all of the desired functionality a compromise must be reached. Therefore, the most important elements of the application was determined based on the vision and the end-user which the game is geared towards. For a user aiming to learn or increase their knowledge of a language the most important aspects are not the competitive parts. Instead the words and the ability to affect the words are more important. Thus, aspects such as high-score, login and timer are not very important. Furthermore, the graphics are of a secondary importance. Thus, giving the very limited time available it was decided that the timer, the user login functionality and the high-scores would not be implemented.

Furthermore, the "super-user"-functionality which was a way of being able to manage the words will be changed. The purpose of the super-user was to provide a way of managing the words without giving every user that access, thereby mitigating the risk of words being added in the wrong category or with the wrong spelling. Instead of a dedicated user, a password will now be needed in order to access the manage words menu. Thus the same functionality is available but having a simpler implementation (and somewhat less security).

In addition, the graphics will be simplified. The amount of letters left in the word will be shown, as well as the position of the letters within the word. The ascii-art will only be implemented if time allows.

Finally, some bugs which were found during the testing phase of iteration 3 will be fixed. A detailed plan of the work to be performed is itemized under "to-do" below. They are represented in the order in which they will be done. The new features will be implemented using test-driven development.

6.4.2 Risk

The risks are substantially the same as for the project as a whole. It is mostly the time which is the bottleneck in this iteration and the possible risks that may reduce the time available will be most important. Once again, this is mostly illness in the work force as well as people connected to the work force. Furthermore, since there is only about two weeks for the entire fourth iteration the time may be reduced further by other tasks that need to be attended (such as work etc.). Risk mitigation is performed by making sure that a good support network is available.

6.4.3 To-do

- Plan work
- Fix bugs found in iteration 3.
- Provide UML-diagrams for the new functionality.
 - Use case diagram.
 - State machine diagram.

- Write test-code for the functions to be implemented.
 - Password handling for manage-user menu.
 - The graphics presentation.
- Implement the new functionality.
 - Password handling.
 - Graphics presentation.
- Test the new functionality statically and dynamically.
- Write documentation.
 - Update Project document.
 - Write/draw class diagram.
- Complete and hand in project.

6.4.4 Time estimates and Time log

Task	Estimate [hrs]	Actual time
Plan work	1.5	1
Fix bugs found in iteration 3	2	1.5
Provide UML-diagrams for new functionality	3	2
Write test code for new functionality	4	2
Implement new functionality	5	3
Testing of new functionality	2	2
Update documentation	4	3
Finish up and hand in	1	1
Total time	22.5	–

6.5 Use cases

The new functionality only generate one new use-case, that is authenticating the user when trying to access the "manage words" menu. This use-case is shown below under section "UC 3" and can also for completeness be found in the document "use-cases.pdf". The graphics implementation did not generate a new use case since it only relate to showing more information during play.

6.5.1 UC 3

Precondition: The user is located in the "main" menu.

Postcondition: The user has moved into the "manage words" menu.

6.6 Main Scenario

1. Starts when the user has chosen "Manage words." from the main menu.
2. The system asks for an authentication password.
3. The user provides the correct password.
4. The system present the "manage words" menu.

6.7 Alternative scenarios

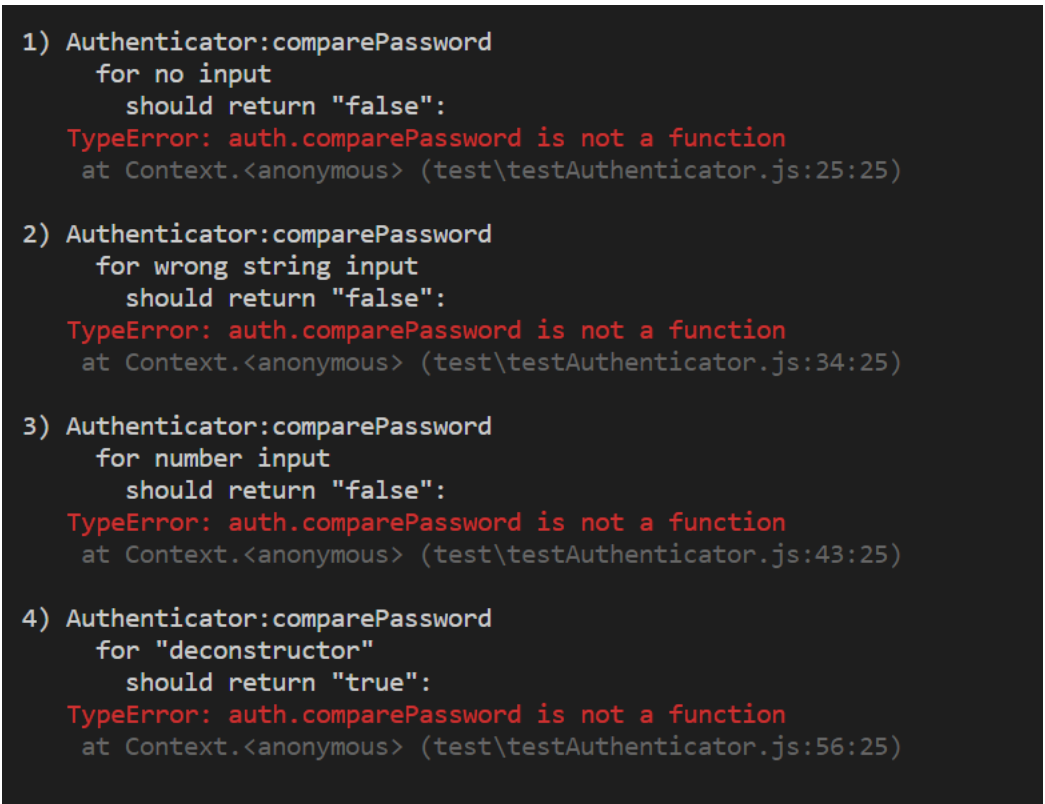
- 3.1 The user provides an incorrect password.
- 3.2 The system informs the user that the password was incorrect.
- 3.3 The system present the main menu.

6.8 Testing

6.8.1 Unit tests

The unit tests are found in the repository in the test-folder. Since I was working with test-driven development this last iteration I first developed text-code for the functionality I was to develop. I developed test-cases for the Authenticator-class which is used for authenticating that a user should have access to the manage word menu. A screen-shot of the unit-test when

run before functionality implementation for the comparePassword method in Authenticator can be found in the figure below.



```
1) Authenticator:comparePassword
   for no input
     should return "false":
     TypeError: auth.comparePassword is not a function
     at Context.<anonymous> (test\testAuthenticator.js:25:25)

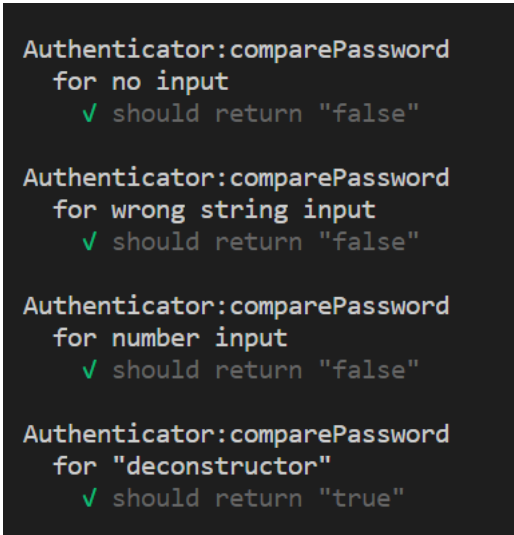
2) Authenticator:comparePassword
   for wrong string input
     should return "false":
     TypeError: auth.comparePassword is not a function
     at Context.<anonymous> (test\testAuthenticator.js:34:25)

3) Authenticator:comparePassword
   for number input
     should return "false":
     TypeError: auth.comparePassword is not a function
     at Context.<anonymous> (test\testAuthenticator.js:43:25)

4) Authenticator:comparePassword
   for "deconstructor"
     should return "true":
     TypeError: auth.comparePassword is not a function
     at Context.<anonymous> (test\testAuthenticator.js:56:25)
```

Figure 1: Test run for Authenticator before implementation

When the functionality had been implemented the test-run worked perfectly. See figure below.



```
Authenticator:comparePassword
for no input
✓ should return "false"

Authenticator:comparePassword
for wrong string input
✓ should return "false"

Authenticator:comparePassword
for number input
✓ should return "false"

Authenticator:comparePassword
for "deconstructor"
✓ should return "true"
```

Figure 2: Test run for Authenticator after implementation

Next I needed to develop a method in the Word-class for generating a string showing the letters found and showing the position of the letters missing, e.g. if the word is "letter" and the letters not found are "e" and "t" the string should be shown in the following manner: "l _ _ _ r". Working with TDD I first created a test for the method to be developed. In the figure below the test has been run before implementation of the method.

```

1) Word:generateWordString
   when called with the array ["s","t","u","d","i","o"] and ["s","u","o"]
   should return "_t_di_":
  TypeError: word.generateWordString is not a function
    at Context.<anonymous> (test\testWord.js:60:25)

2) Word:generateWordString
   when called with the array ["s","t","u","d","i","o"] and [""]
   should return " s t u d i o":
  TypeError: word.generateWordString is not a function
    at Context.<anonymous> (test\testWord.js:69:25)

3) Word:generateWordString
   when called with the array ["s","t","u","d","i","o"] and ["s","t","u","d","i","o"]
   should return " _ _ _ _ _":
  TypeError: word.generateWordString is not a function
    at Context.<anonymous> (test\testWord.js:78:25)

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

```

Figure 3: Test run for "generateStringWord"-method before implementation

After having developed the function the test-run looks as follows. Thus, the method works as expected.

```

Word:generateWordString
  when called with the array ["s","t","u","d","i","o"] and ["s","u","o"]
  ✓ should return "_t_di_"

Word:generateWordString
  when called with the array ["s","t","u","d","i","o"] and [""]
  ✓ should return " s t u d i o"

Word:generateWordString
  when called with the array ["s","t","u","d","i","o"] and ["s","t","u","d","i","o"]
  ✓ should return " _ _ _ _ _"

```

Figure 4: Test run for "generateStringWord"-method after implementation

6.8.2 Static and dynamic testing

The code was tested statically using code review. The code was also tested dynamically by following the test-cases presented below (TC 3.1-3.3). These test-cases are also written in the document "test.pdf" for sake of completeness.

6.8.3 TC 3.1 - Moving to managing words menu

- Name: Authenticating for managing words.
- Test case ref: TC 3.1
- Use case ref: UC 3
- Description: The test case tests how the system responds when the user tries to open the "manage words" menu.

Input

1. Precondition: The user is located in the main menu.
2. Select "Manage word." by using the cursor keys.
3. Press enter.

Expected

- The system prints "You need to be authenticated to manage words."
- The system prompts the user to "Input password:"

Comments No problems encountered.

6.8.4 TC 3.2 - Inputting an incorrect password

- Name: Inputting an incorrect password.
- Test case ref: TC 3.2
- Use case ref: UC 3
- Description: The test case tests how the system responds when the user tries inputs an incorrect password when prompted for a password to move to the "manage words" menu.

Input

1. Precondition: The user has selected "manage words" from the main menu and have been prompted by the system to input a password.

2. Write "Frog"

3. Press enter.

Expected

- The system prints "Unfortunately the password was incorrect"
- The system displays the main menu.

Comments No problems encountered.

6.8.5 TC 3.3 - Inputting a correct password

- Name: Inputting a correct password.
- Test case ref: TC 3.3
- Use case ref: UC 3
- Description: The test case tests how the system responds when the user tries inputs a correct password when prompted for a password to move to the "manage words" menu.

Input

1. Precondition: The user has selected "manage words" from the main menu and have been prompted by the system to input a password.
2. Write "deconstructor"
3. Press enter.

Expected

- The system displays the "Manage Words" menu.

Comments No problems encountered.

6.9 Class diagram

The class diagrams for the fully implemented game can be seen below. The class diagram can also be found in a separate file called "Class Diagram Final.pdf" in the folder "Use Cases".

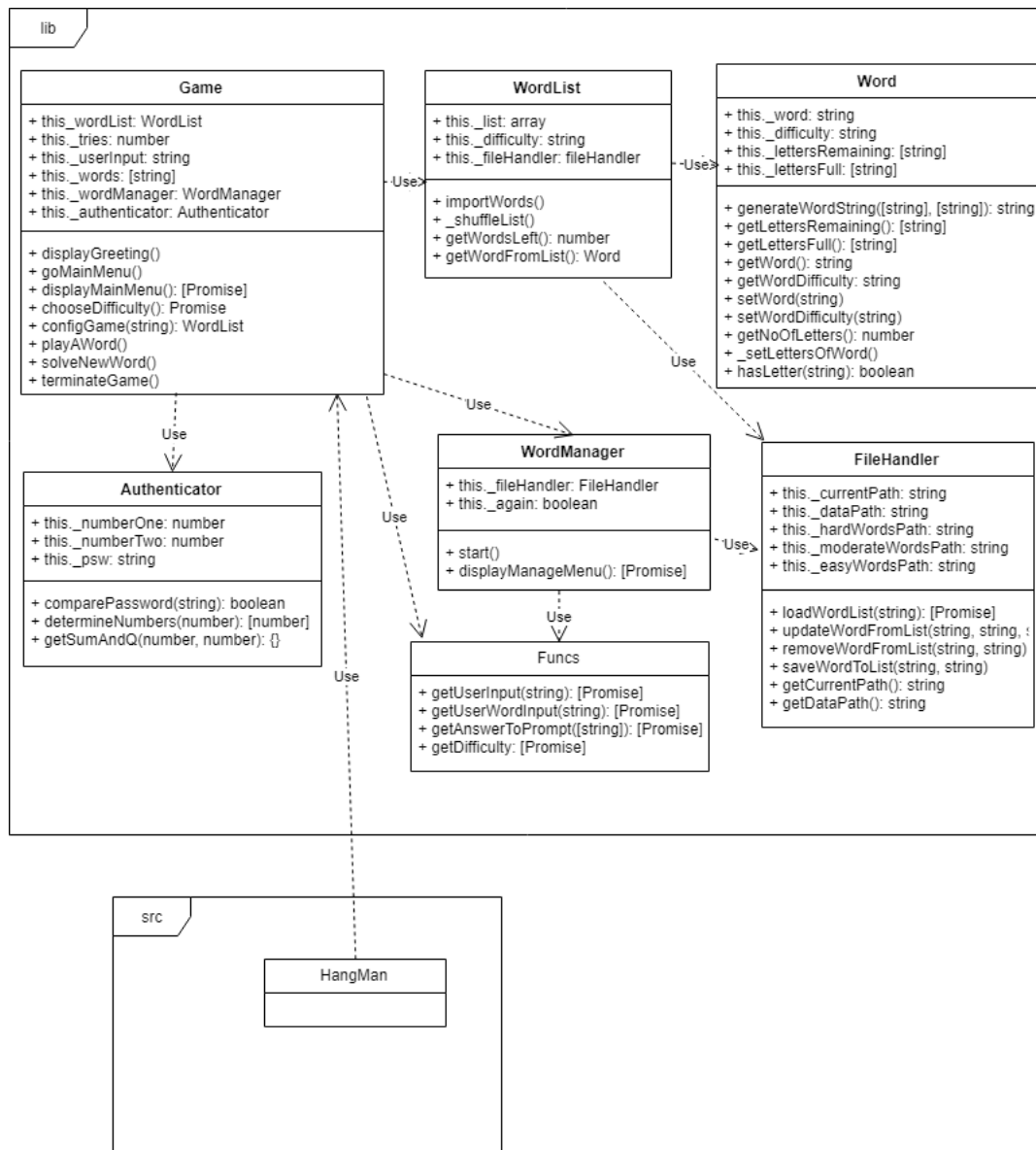


Figure 5: Class diagram for the final implementation

6.10 State machine

I did a basic state machine for the authentication of the user trying to get access to the "manage words" menu. This state machine is shown below and

can also be found in pdf-format in the folder "Use Cases" in the repository. The state machine is very simple since only two paths are available.

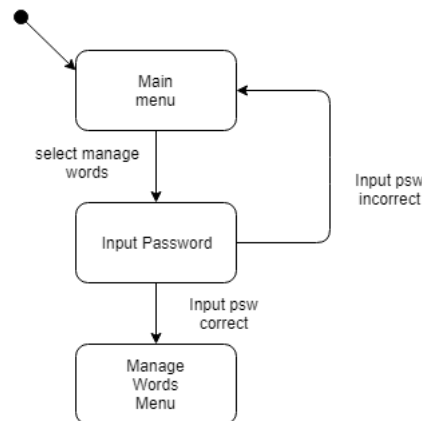


Figure 6: State Machine for the authentication step.

6.11 Reflection

This reflection regarding iteration 4 is also the final reflection of the document. For the final iteration I did not have much time. Therefore I choose to only implement a couple of features such that the game can be played and the words can be managed. I also had to do some bug because of problems with the implementation I found in iteration 3. I really liked the test-driven development methodology, so I tried to follow that method in this iteration. It is quite hard to first develop the tests, even for such simple functionality as I was developing here. But it is fun, and my (admittedly anecdotal and based on very limited data) opinion is that there is less bugs when working in this way. Most of the steps went faster in this iteration, which is understandable since I've already done them once. Even though I want to jump right into the coding I do feel that I now have more of a grasp of why this kind of deliberate development will be faster in the long run. One problem for me during the course was that even though we go through all the steps to develop the code in a deliberate manner I found that I wound up with a quite messy code. I think this is partly because I spent most focus on the actual documentation and the new concepts and I believe that in the long run this methodology will lead to better code and a more streamlined work process.

7 Risk Analysis

The overarching risks for this project is connected to the time available for performing the work. The possibility of a hardware failure is for example relatively low, especially considering the redundancy afforded by having two computers available. However, due to severe constraints regarding the time available all risks which reduces the available time are quite serious. Since all events will impact the time available they are all quite serious which make the most probable events most important (since both the impact and the probability are quite high). Thus, a lot of effort will have to be made to make sure that probable events are avoided or mitigated. The staff has children in pre-school which unfortunately equates to a lot of sickness. The probability that this event occurs is hard to reduce which is why the effect of such a event must be mitigated. In an effort to create somewhat of a buffer for these kind of events it is wise to try and keep ahead of schedule if possible.

Thus, the key risks have to do with estimations and people working within the project, both are risks in relation to the time available. There is no significant risk that changes to the software, as a response to management or costumer wishes, will need to be performed. The hardware risks are low as explained above. Since the application is a very basic application there are also no great risks in tools or technology changes impacting the development process.

7.1 List of risks

Based on the above discussion a list of risks are presented below. Only risks which are deemed at all possible has been accounted for in the list. As can be seen, the risks are all related to the available time.

Risk	Probability	Impact
Hardware failure	Low	Catastrophic
Illness	High	Serious
Underestimation of time requirements	Moderate	Serious
Too many features required in application	Moderate	Serious

7.2 Strategies

Since most of the risks are time-related the most prudent course of action is working ahead of schedule and use all time available. Thus, theory may e.g. be studied during transportation to and from events. Furthermore, in order to be use the available time efficiently it is important that the staff get good sleep and exercise regularly, which is why this is encouraged. This is a more general approach to utilization of the available time in an efficient manner. In the sections below each specific risk will be briefly addressed.

7.2.1 Hardware failure

Even though this risk is quite low it is still considered since such an event would be catastrophic. In order to reduce the likelihood of a hardware failure redundant systems are used (i.e. two computers). In the unlikely event of both systems failing a borrowed computer will have to do. This will of course be costly from a time-perspective but less so than having to buy a new computer.

7.2.2 Illness

This is a quite probable risk. Either that the developer himself falls ill or that his child do. The risk of the first scenario may be reduced by the measures described above as well as by eating healthy. To a certain point the same applies to the developers child. If the child would fall ill a fallback approach would be to enlist the help of other people to take care of the child and in that way increase the time for working on the project. Other remedies are of course medication in case of severe illness.

7.2.3 Underestimation of time requirements

This is a risk which is hard to reduce initially. However, as the project develops the time estimates may be adjusted based on experience. Thus, by minimizing the fault in the estimation the risk of the event happening will be lower. The risk may also be reduced by knowingly overestimating the time for each task by a certain factor, to create a buffer of sorts. If the time requirements have been greatly underestimated some requirements in the coming iterations may need to be scrapped in favor of essential requirements.

7.2.4 Too many features required in application

This risk is also quite probable and is mainly reduced by having a realistic approach in the planning stages. Prioritizing the requirements may be performed in order to let go of features which are not essential if it is determined that the time will not be enough for completing all the features. A core list of essential features must therefore be determined and prioritized during development.

7.3 Reflection

I felt that performing the risk analysis was quite hard. For my specific situation, developing a quite basic application and having a very limited amount of time, it feels like the risk will always be related to time. If no direct costs are involved I guess the risk is almost always to complete the project in time (assuming that the project is possible to complete at all). This is very true here as well, especially considering the fact that the application is quite limited (in this sense it is not a risk that the application cannot be developed). I feel that risk analysis will be quite different in a large project, considering management, customers, budget etc. But this is still a healthy exercise I believe.