# SOFTWARE DEVELOPMENT PROJECT

Christian Fagerholm, CF222JF

August 20, 2019

# Contents

# 1 Revision History

| Date | Revision | Description | Author |
|---|---|---|---|
| 2019-07-18 | First | Vision document and started project plan | CF222JF |
| 2019-07-25 | Second | Finished the majority of the project plan | CF222JF |
| 2019-08-16 | Third | Finished the testing. | CF222JF |
| 2019-08-18 | Fourth | Added additional functionality | CF222JF |

# 2 General Information

| **Project Plan** | |
|---|---|
| **Project Name** | **Project ID** |
| Hangman | 1337 |
| **Project Manager** | **Main Client** |
| CF222JF | Tobias Gidlund |
| **Key Stakeholders** | |
| Client, Developer, Tester, Project Manager | |
| **Executive Summary** | |
| - | |

# 3 Vision

The vision for this application is to have a simple game of "hangman" that can be played in the console of any major operating system. The game should show the players progress by drawing a hanged man in unicode characters. The word should be written below the gallows if a player succeeds with guessing the correct characters, otherwise it will just show a line/underscore. The player gets 8 attempts to guess right, otherwise the player will lose the game. There will also be a scoreboard where the players can keep track of their score. The score will be calculated as how many words the player has guessed correctly until the player loses or exits the game. If a player exits the game while having a higher score than someone else on the highscore scoreboard, the player will enter his name and the scoreboard will be updated. The players should also be able to add new words to the application.

## 3.1 Reflection

Creating a vision document feels somewhat unnecessary for a project where I am the only developer and especially for an application this small, however, if there were more than one developer for this application or if the application was a larger project, it would make a lot more sense as the view on what the game should include or not is highly subjective. The larger the project, the more important the vision document is to ensure that we all share the same view on what should be produced.

# 4 Project Plan

Write a project plan for the project. This project plan should show the way to the complete and finished application, something that you should be able to follow. Write as much as possible in the project plan, use the material available on mymoodle (deadlines etc.), and update the document throughout the course when you know more in the later assignments. Again, as an addition, write down your reflections on creating a project plan. This reflection should be about 100 words.

## 4.1 Introduction

This project goals is to create a working game of "hangman" that can be played in any console. The application should be able run on any system that has Java installed and ha access to a console with a keyboard connected. The game is fairly simple. The game will start with a randomly selected english word, then as a player, you guess either a letter or the whole word if you wish. If your guess is correct, all the instances of the guessed letter should be revealed, and if all the letters are revealed, you win. If you guess incorrectly, A gallows should be shown. For every incorrect guess, the gallows will add another part until the image shows a hanged man. If you guess incorrectly 8 times, you lose.

To make the game more interesting, we will have a scoring system based on the number of guesses needed to win. We will also have a scoreboard where we can see the top scores and the names of the players who made that score. If a player scores better than another player on the scoreboard, the player will be asked to enter his player name and his score will be shown on the scoreboard.

## 4.2 Justification

Because it is a fairly simple and fun game that everyone can learn easily and it has been around for ages. Making a software that automates the drawing and selection of words enable us to play more times in the same time frame as playing the game on a whiteboard. We can skip all the tedious stuff and get right into playing.

## 4.3 Stakeholders

1. Project Manager
2. Developer
3. End-user

## 4.4 Resources

The resources available for this project is a windows 10 laptop with a Java 11 developers toolkit installed and Eclipse as compiler. We also have access to one developer, which is also the project manager for this project.

## 4.5 Hard- and Software Requirements

**Used**

- Windows 10
- Eclipse
- Java 11 developer toolkit

**Required**

- Any Java installation
- Access to a console
- Keyboard

## 4.6 Overall Project Schedule

Since this project started in the middle of the summer, I do not really have the same time constraints as others, but I expect this project to be done in 4 weeks. Starting from the 18th of July, there will be one iteration per week.

25th of July - Vision document, Project plan and application skeleton 1st of August - Application with extra features done. 8th of August - Testing and documentation 15th of August - Finished project report.

## 4.7 Scope, Constraints and Assumptions

Detail what is part of the project and what is outside – specify the scope of the project. The scope of the project is to create a functional hangman game in a console environment. This game is going to be created in Java due to familiarity with the Java language.

Since it is stated in the requirements that it should be a console game, any other graphical interface is outside of this scope. Even though there is a possibility of multiplayer, this project will focus on the single player functionality.

## 4.8 Reflection

Creating a project plan for a project this small is pretty easy, however, this would be a lot harder if the scope increases or there are requests for new functionality. The project plan feels like a very live document, as there might be architectural decisions affecting the project and therefore the previous plan must be modified. The system needs to be adaptable to change.

# 5 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

For this iteration, I expect to have most of the basic design finished as well as the skeleton of the application. The design for the application will be provided in a separate document for this specific iteration. Since I have decided to go with a Model-View-Controller architecture, the view and the model part of the application should be done as well since they are very basic. The application will not be functional yet as it is intended to be a software skeleton, so a basic overview of needed methods for the application.

## 5.2 Iteration 2

All the design decisions and reflections are found in the design document for this iteration.

## 5.3 Iteration 3

Major testing, bug testing and fixing some of the issues/concerns. Details can be found in the test plan.

## 5.4 Iteration 4

The last iteration will be covered in its own document.

# 6 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.

## 6.1 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:** Insufficient details in the requirements.
**Description:**
Since there was not a lot of information to use for the application, we cannot be sure that the end user will be satisfied with the product since the requirement specification was not very detailed, which in turn means that the developers and designers might have to make assumptions regarding some design decisions. Assumptions might lead to bad or incorrect decisions which either requires the developers to change the application or the end-user might not be satisfied with the product.
**Probability:** Medium
**Impact:** High

**Risk:** Rushed product or non-finished product
**Description:**
Due to strict deadlines and the low number of developers, there is a risk that the software will be rushed as there is a lot to do and very few people who are working on it, which means that there is a lot of responsibility on each developer. This might lead to the developers rush the project in order to finish the product on time. The issue with rushing the project is that there might not be enough time to do thorough testing and the end product might have a lot of issues that are not handled. Worst case, we deliver a product that is unfinished and needs patching.
**Probability:** Medium
**Impact:** High

**Risk:** Limitations to or incorrect choice of programming language
**Description:**
The choice of programming language for the development of the software can play a big part in the other design choices. If you intend to develop a web platform, you would probably use a language that is better suited and has more support for web programming, such as JavaScript. An incorrect choice of programming language can usually be worked around, although it takes a lot more time. Worst case, parts of the software needs to be reprogrammed in another language since the functionality we need might not be supported or available in the current programming language.
**Probability:** Low
**Impact:** Medium

## 6.2 Strategies

First of all, lets address the programming language. Since we are doing a local application, Java will be great as it is effective, simple and has good support. It is also very forgiving, as it prevents you or makes it harder for you to change variable types, or call insecure functions. There is also many external libraries for java if needed, and it has been around for a long time which in turn means that there is a lot of documentation and guides if we should run into trouble. I am also most familiar with Java, so therefore this will be the perfect fit.

The second risk we will address is the insufficient details in the requirements specification. This software we intend to design is not that big, and the requirements specify the requirements pretty clear, but not too in-depth. This mean that there might be assumptions on my part. I haven't got any possibility to ask the end-user to clear out any misunderstandings, but have to try to follow the specification that was provided. To minimize this risk, I will document my assumptions for the application and I will present them on every iteration so that they may be addressed as early as possible.

Third, the rushed and/or unfinished product risk will be mitigated by having an open schedule. At the time of this project, I have no other projects that I am working on concurrently, which means that I can spend all of my focus on this project. Also, by having a strict schedule, it is easier to keep track of the progress and to assess where I might have to put more time. If I can follow the schedule I should be able to finish this application on time.

## 6.3 Reflection

It is very hard to find plausible risks for a project of this size, but the risks I found are very general risks that are plausible for all sizes of projects. There are always risks with software engineering, but for a project this small, there aren't many risks. On bigger projects, risk assessment is a lot more important since the problems impact more people and failure to address and mitigate risks can have really dire consequences.