SOFTWARE DEVELOPMENT PROJECT

ROXANNE ALBINSSON
IHEANACHO

Company: Swedevelope

February 3, 2020

1. | Revision History

Date	Project Version	Description	Author
February 3, 2020	V1.0	First increment and skeleton code creation for project initiation	Roxanne Iheanacho
February 24, 2020	V1.1	Second increment and implementation of updated code	Roxanne Iheanacho
March 9, 2020	V1.2	Third increment and implementation of test and code validation	Roxanne Iheanacho
March 20, 2020	V 1.3	Fourth increment finalization of project and feature additions	Roxanne Iheanacho

2. | General Information

General Project Summary	
Creation of Hangman game through Javascript code presented to Linnaeus University in order to research code quality solutions. Initiated by company Swedevelop. The Project will be created in four sprints, four iterations with the whole project lasting over a period of 2 months.	
Project Title	**Project ID**
Hangman Project	Project ´No.1
Project Manager	**Client**
Roxanne Iheanacho	Linnaeus University
Executive Summary	

Roxanne Iheanacho will serve as project manager and developer of the Hangman game creation that will be initiated in Uppsala Sweden using a Dell Laptop as well as Visual Studio Code in order to develop the hangman game. The goal of the project for the client is to write the best quality code that is understandable to anyone who may use the code or develop it further at Linnaeus University.

3. | Vision

Vision as of 24.February.2020

The vision Swedevelop and Linnaeus University share is to create good quality code in increments using SCRUM and managing the project via a Project Manager who will set up each sprint run. Swedevelop aims to provide the best, most readable code in which Linnaeus University will be able to research in order to develop new ways for Javascript games and beyond. This will be done through Javascript code with each module created and will be run asynchronously. There will be a module for a timer, a high score list, a module to set up pieces of hangman creation, and the module for the actual game. It will be created with a template of an html.index page and have css implemented in order to present the game as beautifully and simply as possible. The HTML and CSS will use LNU theme colors (Black, Yellow, and White).

Revised Vision as of 20.March.2020

The vision of SweDevelop and 1DV600 of LNU is to create a Hangman game using plain javascript in node.js console in order to test user input for faults and errors in code. The vision further sees being able to watch the project grow and create UML diagrams that represent the components and units of the potential code, test the code and then revise the UML diagrams and code based on the tests. This will allow a healthy project between stakeholders in order to better understand faulty code in order to mitigate risks and understand how to communicate those risks better in the future to enhance the ability to create better quality code and exercise better javascript programs in the future.

4. | Project Plan

Project Plan as of 24.February.2020

In order to write the Hangman Application, there are some guidelines that will be followed according to Linnaeus University, Course 1DV600 on the Moodle Site with simple instructions.

That is that Hangman should have several pieces, between 8-10 in which a user inputs a right or wrong answer resulting in the construction of the hangman. If the user inputs too many wrong answers it results in the man getting hanged - that all 8-10 pieces are constructed. The project plan is to write the skeleton of the code, or the outline with all modules in order to show, in sprints, what code is created and to be presented. These sprints will be done accordingly with the skeleton code done on 03.02.2020, with an update after feedback on 24.02.2020 that will also include more detailed javascript code of states, with an array of words to be used in the game of hangman that will have new functions such as NumberofTries that will work with whether or not to terminate a game as well as nickname and high score list option for use cases diagram, and will include a class diagram showing the classes and children of the newgame class for the Hangman Game, with a third update and overall substance complete on the code on 09.03.2020, and then the finalized project and application presented to Linnaeus University on 20.03.2020. As for the current reflection of SweDevelop, it is believed that the application should be able to have a code that is generated on 03.02.2020 and will be able to use the feedback after each sprint with code developed for review in order to develop code. As there currently has not been any feedback given to SweDevelopes first and initial sprint, it is not possible to have a fully developed reflection.

Project Plan Update and Reflection 20.03.2020

The project plan stands as previously stated however with a few revisions. The code will not include a picture of a man getting hanged within a code however will be represented by underscores with the user starting off with 10 tries (previously called NumberOfTries). If the user runs out of tries, the nolives function is called and the game quits. As the feedback times varied, in the final iteration all feedback from assignment 2 and 3 will be implemented only then when UML Diagrams will be restructured, new features such as the QuitGame and Guessword option are included and re implemented in diagrams. The overall code was completed in iteration 3 with final implementations in iteration 4.

Reflection: The project as a whole was not so much about creating a game with many features but rather being able to understand sprints, iteration, and coding processes in order to develop an understanding of how to test for coding failures, to expect risks and failures and how to be resilient and create new goals each sprint. As it can be represented in this document via the different reflections and posts depending on the sprint, the main outcome was the ability to adapt and understand how to visualize code in UML Diagrams in order to truly understand the needs. This project helped to fully understand requirements and why UML diagrams and testing is important. In the beginning, it seemed the best thing would be to develop an amazing game, however an amazing game is never that amazing if it will keep failing. It therefor helped

SweDevelop to streamline its future capabilities to develop UML diagrams and code tests to represent the requirements of LNU better in order to work more effectively each sprint.

4.1 Introduction

The Hangman Project works with the Linnaeus University Course in order to develop a Hangman Game for the 1DV600 using the 1DV600 course guidelines. The code in the hangman game will be used in order to find faults and errors within the code and more specifically focusing on the input functions and methods to develop a better understanding of what could go wrong within the code. This will help SweDevelop and Linnaeus University further visualize what faulty and error prone code is in order to mitigate future risks of system failure of future game programming.

4.2 Justification

The application should be made in order to create innovative code while being able to implement strict guidelines given by world-renowned researchers and teachers at Linnaeus University. From a self-oriented point of view, Swedevelop builds a future client and portfolio for free . From an interactive point, Swedevelop uses its developer knowledge to help Linnaeus University act as a client. This also helps to develop a better view of code, how it is communicated and visualized and what could possibly go wrong and will go wrong with the code.

4.3 Stakeholders

Stakeholders in the project creation are Roxanne Iheanacho, a student and future developer who acts in regards to “SweDevelop” a future company concept. Linnaeus University, and the course 1DV600 moderators are stakeholders - Namely Tobias Andersson Gidlund, Daniel Toll, and Tobias Olsson who will be reviewing the quality of the code and project.

4.4 Resources

Resources used are the Moodle software of LNU, Researchers and teachers of the 1DV600 course, Visual Studio code for development, Node.js, Gitlab, Slack for communication, Google Chrome for the console when visual studio and node.js does not work, Dell laptop of SweDevelop, Swedevelops lone developer and program manager- Roxanne Iheaenacho, and SweDevelops home office in Uppsala, Sweden.

4.5 Hard and Software Requirements

Hardware requirements are an up to date laptop that has 5GB or more in storage in order to hold software. Software requirements are Visual Studio Code and Google Chrome, Node.js, Gitlab, Adobe editor for pdfs, Google drive, and Draw.io for diagram creations.

4.6 Overall Project Schedule

The overall project schedule will work on a weekly (Monday through Friday) 8am-12pm basis. In order to develop the code and project as swiftly and effectively as possible, 4 hours per day will be allocated in order to build the best code possible from Monday through Friday. That is from 22 January 2020 to 20 March 2020 on weekdays, 8-12pm SweDevelop will work to implement the best version of Hangman possible through Javascript, HTML and CSS.

Note: It was later revised for the fourth increment that HTML and CSS will not be used.

4.7 Scope, Constraints and Assumptions

The Scope of the project will rely on 1DV600 knowledge, Swedevelops previous knowledge obtained from LNU coding guidelines obtained via courses 1DV022, and 1DV021. The scope will rely on ‘use strict’ and NPM standards of Javascript, and HTML5 and CSS standard coding and be used in a Chrome format. Therefore all other materials inspiring code, for example youtube materials, Stockholm tech events and conferences which SweDevelop attends, and other SweDevelop peer networks from LNU and game inspiration found elsewhere are outside of the scope of game creation. The code will be generated and tested in the node.js console in pure Javascript. The time constraint of a 2 month period did not allow for more advanced code. It is assumed that there is a limited time for grading, feedback in order to implement revisions and perfect the code and therefor a basic game with minimal variations, transitions and use cases will create an ideal system for basic testing and diagram creation in order for all stakeholders to visualize code and be on the same page.

5 | Iterations

Each iteration will be referred to as a version or sprint. Iteration 1 will also be Version 1 and Sprint 1. As each iteration will have different expectations, the following will be updated after the feedback from the client for the previous iteration is integrated so that there can be a whole and clear view of what each iteration will entail in its entirety.

5.1 Iteration 1

The instructions for Iteration 1 is to create a skeleton code for Hangman. For the Javascript portion, there will be three modules, one with the Hangman game class, another with the pieces which will integrate what piece is what in the actual “man” illustration, then there will be a timer. These modules will all be called by App.js. It is assumed that in order to play the game there will need to be an interface for the website, although it is not specified in the clients moodle requirements. It will therefore use an index.html to call the app.js as well as a style.css in order to create a beautiful website to host the javascript game creation interface. This will be uploaded on Gitlab on February 3, 2020 as a skeleton code to be reviewed.

5.2 Iteration 2

Iteration 2 will update the code that will create the foundation for the whole system to be tested for iteration 3. This code will be created in completion to allow for 3 use cases, several states for state machine diagram creation, and classes in order to represent different facets of the code. This also allows for visibility of the stakeholders of Linnaeus University and SweDevelop to plan for the next iteration of testing as diagrams have been created. The focus will be on implementing hangman with 4 words as cases. These will be test cases to be run and to test with UML State Diagram, Class Diagram and Use Case diagram for presentation. Diagrams and Use Case Scenarios will be included in a separate document labeled “Diagrams.PDF”

5.3 Iteration 3

Iteration 3 will have a full and structured quality code that will be validated. The game implementations, components and units will be ready for testing via static tests such as code review and then for more dynamic tests such as manual tests. These tests will be orthogonal and test input features such as name and word input. The automated unit tests will be run with a TDD, specifically Chai and Mocha for Javascript on the Visual Studio code via the Node.js npm framework. 5 automated unit tests will be run to test input and output and with one feature implementation that will be run for the first time in order to test for possible future implementation.

5.4 Iteration 4

Iteration 4 will create revisions to the code, diagrams and complete the project as a whole. This will allow for conclusions and final feature implementation. Specifically the guessword() function will be created as a special feature which was previously tested in iteration 3. A new quitgame() function was created to connect to wingame() and nolives() function as an extension. This was created as in iteration 2 it was found that the game was not able to quit mid game which

was specified in grading of Linnaeus University stakeholders (teachers) as a requirement for the state machine diagram which was enhanced and recreated within the fourth iteration. The fourth iteration also explained and went through all previous work of iterations to explain the final result. Prior to the 3rd iteration, it was found that the code worked, but in order to create automated unit tests, the inputletter() function removed certain portions and included error messages. This allowed us to catch bugs for input, however rendered the game without function. Although the game does not work, it caught errors of input functions which allowed for the goal of the project of Linnaeus University and SweDevelop to come to a whole in order to visualize and find faults and bugs.

6 | Risk Analysis

There are several categories of risks and their possible effects on the process of the outcome of the project. Some risks are minimal, possible, and very possible. Then there are effects such as inconsequential, serious, and catastrophic. Depending on the stakeholders, software and hardware, time, company, and client risk will be evaluated in this chapter. In reflection as of Sprint 1, risks should be calculable and the ability of all risks to be mitigated should be relatively easy to do so long as the project plan is followed.

6.1 List of risks

1. Organizational risks such as LNU's inability as a client to give feedback in time as well as SweDevelop to not follow through on coding on time is possible. This risk is unavoidable as usually coding does not happen on schedule.

2. Then there is an issue that timing does not happen as followed from the side of SweDevelop. This is an avoidable risk however with catastrophic effects of developing to the LNU client side.

3. Then there is a hardware risk of Swedevelops hardware and or software crashing, which is mostly avoidable with minimal chance however with catastrophic effects.

4. It was found within the third iteration where faults and errors were found via Chai and Mocha testing in the Node.js console that the actual testing would show the bugs of the code. The deliberate production of errors allowed SweDevelop and Linnaeus University to visualize the risk of the hangman game that showed that various forms of input would not allow the code/game to function as it should. Another risk was that finding the bugs and errors in the code would result in system failure without producing a solution to the

system failure due to the time constraint of two months. The fourth iteration found and was able to visualize these errors, bugs, and create diagrams to effectively map code which was the overall goal of the project, however the code that was produced within the development period was proven to be not as useful or effective for the game in doing so.

6.2 Strategies

In order for SweDevelop to mitigate organizational risk, time logging and effective time planning with strict regimen will be implemented. SweDevelop will properly communicate with LNU in order to mitigate risk.SweDevelop will keep an extra laptop handy with software so that way there is backup hardware and software should there be a catastrophic crash. Swedevelop will also mitigate any loss of work by frequently commiting work to Gitlab and saving on a cloud service in order to not lose any work and create a backlog in time.

In order to develop clear communication, slack will be implemented and a timely feedback to feedback and questions will be posed via Slack and gitlab to teachers. Slack will be monitored on a daily basis to keep up with possible failures within the overall project and be able to clearly understand and update the code and project as needed.

7 | Time log

Time logged/Date	Work Description	Project version	Work done by
2hrs/22.01.20	1DV600 Hangman Format introduction	V1.0	Roxanne Iheanacho
2hrs/25.01.20	1DV600 Project Materials Familiarity	V1.0	Roxanne Iheanacho
1hr/28.01.20	1DV600 Project Management with book	V1.0	Roxanne Iheanacho
6hrs/31.01.20	1DV600 course book and code construction/ familiarization	V1.0	Roxanne Iheanacho
4hrs/02.02.20	Sprint 1 completion	V1.0	Roxanne Iheanacho
4hrs/03.02.20	Sprint 1 completion code and pdf	V1.0	Roxanne Iheanacho
4 hrs/ 07.02.20	Course literature/diagram building	V1.1	Roxanne Iheanacho
3 hrs/ 08.02.20	Course literature/function / diagram building	V1.1	Roxanne Iheanacho

5 hrs/ 13.02.20	Code implementation/class diagram/literature and lecture review	V1.1	Roxanne Iheanacho
2 hrs/ 18.02.20	Diagram building	V1.1	Roxanne Iheanacho
4 hrs/ 22.02.20	Studying lecture materials, implementation of code, preparation for sumission to LNU	V1.1	Roxanne Iheanacho
10hrs/variable days	Reading instructions and understanding readings	V1.2	Roxanne Iheanacho
7.5hrs/variable days	Coding updates	V1.2	Roxanne Iheanacho
5hrs/variable days	Reviewing code	V1.2	Roxanne Iheanacho
2hrs/variable days	Manual test	V1.2	Roxanne Iheanacho
20hrs/Variable days	Restructuring/redoing and creatin methods	V1.2	Roxanne Iheanacho
8hrs/variable days	Automating Unit Code with TDD, Chai and Mocha test setup and config	V1.2	Roxanne Iheanacho
8hrs/variable days	Feedback from iteration 2 reviews restudied and reapplied	V1.3	Roxanne Iheanacho
8hrs/variable days	Overall program and structure review from assignment 3 reevaluated in order to implement changes within code	V1.3	Roxanne Iheanacho
8hrs/variable days	Literature and other source (stackoverflow) reviewed in order to re implement code structure	V1.3	Roxanne Iheanacho
8hrs/variable days	Slack review as well as last diagrams, test, and overall code review worked on in order to try to resolve code, visualize what is needed for further implementation	V1.3	Roxanne Iheanacho

8 | Submission Details

Submission for each sprint for the client will be handed in on GitLab. Below is a table with submission details outlining what is submitted for which version and date.

Submission Date	**File**	**File Description**	**Version**
03.02.2020	"Sprint 1" File Roxanne Iheanacho Software Development Project PDF	Folder with Css, HTML, and JS modules for skeleton code labeled "Sprint 1", and PDF with Project Management Project	V1.0
24.02.2020	"Sprint 2" files, PDF, ReadME, JS, HTML CSS file	HighScore.js, App.js, Game.js, index.html,style.css, ReadME, PDF link	V1.1
09.03.2020	"Sprint 3"files, Test and ProjectPDF, README, Hangmantest.js, Hangmantest.html, test script and json files for Chai and Mocha test implementation	Read.ME, Hangmantest.html,Hangmantest.js, Test.PDF, ROXANNEIHEANACHOSOFTWAREDEVELOPMENTPROJECT9.3.pdf	V1.2
20.03.2020	Sprint 4 files include Test.Pdf, README, Project.PDF, Hangmantest.js, and Diagrams.Pdf	Test.PDF = Testing completed Diagrams.PDF = All diagrams Project.PDF = Project overview	V1.3