**Final Report**

1. **Name, the date, and project name.**

Name: Ishav Sohal

Date: March 21st, 2020

Project Name: Hangman Game

1. **A brief description of the game.**

The program that I have created is a Hangman Game. This is a game where the user must try to guess a hidden, unknown word, by choosing letters of the alphabet in order to uncover the letters of this specific word. Once the user has uncovered all of the letters of the specific mystery word, and has thus revealed the word itself, they will have won. However, the user has a limited number of incorrect guesses. Every time the user guesses a letter that is not within the specific mystery word, a body part is added to the “Hangman” image. Once the full body of the hanged man is revealed, the user will have lost. Essentially, the user will be able to guess 6 incorrect letters.

This game involves the use of problem solving skills, as it requires a lot of thinking to determine what specific letters to guess, based on the positions of other letters of the mystery word that have been guessed. It also requires an extensive knowledge of English vocabulary, because once a few letters of the mystery word have been guessed, the word itself can be determined by the user if they have a thorough knowledge of English vocabulary. The knowledge of English vocabulary also includes knowing what letters are most commonly found in English words, as those are the letters that have the highest chances of being in any mystery word.

1. **Results of testing:**
   1. Date of the testing by advisor/contact person.

March 21st, 2020

* 1. Tester's name and occupation.

Name: Kuljeet Sohal

Occupation: Database Administrator

* 1. Summary of features tested.
* “Play Game” button
* “Instructions” button
* “Exit” buttons, which have the function of simply exiting the program
* “Back” buttons, which have the function of going back to the previous window
* All of the letter buttons, from A to Z
* “Clear Records” button, and seeing if the Win/Loss records were successfully cleared whenever the button was clicked.
* The event that occurred when the game was won. I had the mystery word printed out in the output screen for this, so it would be much easier to test.
* The event that occurred when the game was lost.
* Sound effects for each of the buttons, as well as the audio clips that played when the game was won/lost
* The blank dashes that represent the specific mystery word. For this test, I had the specific mystery word printed out on the output screen for the tester. This feature was tested out by having the tester guess letters that were in the specific mystery word, and seeing if the specific letters guessed appeared in the correct letter positions within the blank dashes.
  1. Did the program work as it was supposed to?

Yes it did. All of the features of the program worked successfully, and exactly how they were meant to.

* 1. Did [s]he find the program easy to use?

Yes, the tester did find the program easy to use. He had first decided to read the full set of instructions before proceeding, which provided him with all of the information he needed to know about how to play this game. After that, the tester was able to play the game with no trouble.

* 1. Is the program useful to the target audience?

Yes it is. To recall, the target audience of this program mainly consists of other computer programmers, young students and internet users.

This program is useful to beginner computer programmers because when they have access to this program, they also have access to its source code, and GUI design. With this, they are able to learn exactly how this program works, and see how each line of code contributes towards the various features of this program. These computer programmers are able to then use this program, and the coding methods/strategies found within it, to help them create their own program.

This program is useful to young students because with this program, they can learn to recognize new words, based on their length and the positions of key letters within them. Young students may also discover patterns in words, such as common letters existing amongst a variety of words, and how certain letters are commonly paired together. This would, overall, improve their vocabulary and challenge their minds by making them use their problem solving skills.

This program is useful to regular internet users because it would provide them with a fun, enjoyable game that they can use to simply pass time.

* 1. If the program were made available to the tester, would they use it?

Yes, the tester would use it. The tester enjoyed playing this game, and found it very interesting. They said that they would be willing to play this game in their free time if it was available to them.

* 1. What recommendations for improvement did the tester make?
* The tester believes that the Hangman GUI image could be more lifelike and realistic, as well as have varying facial expressions.
* The tester believes that there should be an option for the users to get a hint about the mystery word (Ex. if the mystery word is a location, the hint could tell the user that the word is a city, country, landmark, etc.)
* The tester believes that if there was theme music that played throughout the game, it would make the games environment more entertaining and relaxing, and it would have made it more fun for them to continue playing.
* The tester believes that it would be a good idea if the game had a multiplayer setting, where two people (or two teams) could play side by side. One person/team could be assigned to choosing the specific mystery word, and the other person/team would try to guess that word. Either person/team would be awarded with a point if the word was guessed successfully or not.
* The tester believes that it would be a good idea to provide the users with helpful tips and strategies, as the tester believes that there are some people who have little to no experience with word-guessing games, and may need some assistance.
* The tester believes that the users should be able to choose how many letters the specific mystery word that they have to guess has.
  1. Are there areas of the program that require upgrading or reworking?

Yes there are. Here is a list of some upgrades and features that could be implemented into the program:

* Making the “hanged man” image more realistic and lifelike
* Giving the users more incorrect guesses before they lose the game.
* Giving the users hints/lifelines, tips and strategies
* Having theme music, with options to turn it on/off and adjust its volume
* Varying difficulty levels for the users to choose from (easy, medium, hard)
* Telling the users what topic their specific mystery word falls under
* Implementation of animations
  1. What would you need to make version two of your application even better than this first version (more time, more programming expertise, etc.)?
* More time. Many of the ideas for the improvement of this program came to me as I was working on this report, as well as when testers gave me feedback. If I had more time to work on this program, I would try to implement these features, which would make my program significantly better. Also, with more time, I would be able to have more people (possible users) test out my program, and provide me with even more feedback/recommendations. This would also allow my program to fit the needs of the possible users of this program more effectively.
* More experience with Java, and more knowledge of Java programming. If I were to learn more about Java programming, and its vast features/functions, I would be able to make this program much more sophisticated. I would be able to add more features that would make this game more professional, and user friendly.

**4.** **Review of the Management of the Project**

* 1. Compare the project plan to actual performance.

Similarities:

* The idea of having multiple screens was always a part of my project plan, and it was implemented into the program successfully.
* The main layouts for each of the different windows went according to my project plan, as they were based on the screen layouts that were created in the Feasibility Study.

* The main rules and logic for this program was always a part of my project plan. As I was creating this program, I continuously referred to the flowchart that I had created in my Feasibility Study. I created my program based on the logic of that flowchart, which indicates what would happen when each of the possible game events had occurred, like guessing a correct/incorrect letter or winning/losing.
* The time estimates that I had made for the development of this program's features were quite accurate. However, some features took longer than I had anticipated to develop, as I needed to do additional research in order to determine how to complete certain features, which I did not account for. Also, at some stages of the program development, I had to take my time to think about how I wanted to proceed, and what additional features I wanted to add.

Differences:

* For the Instructions, I had originally planned on there simply being a pop-up window (JOptionPane) that displayed the instructions. However, in the actual program, I have instead created a separate window for the instructions. I did so because I believed it would make the program more visually appealing, as in the separate window, more effort was put into the design overall.
* In my project plan, I had planned on adding theme music, alongside with the sound effects for each of the buttons. When I had implemented this, I found that the sounds overlapped each other too often, and made it overwhelming to listen to. So, I decided to keep the sound effects for all of the buttons, and instead of having theme music, I implemented short audio clips that play whenever the user wins or loses.
* In the actual program, I have added an XML file, which stores the users Win/Loss records. I also created a button that allows the users to clear these records. This was not a part of my project plan, but as I was working on my program, I realized that it would be a good idea for the users to have a chance to look at all of their Win/Loss records for every time that they used the program.
* In the original project plan, the scoreboard that I had planned on creating only consisted of the number of wins the user had (their score). In the actual program, I included the number of losses as well.
* In the actual program, I decided to add various images on each of my windows. I found that this made the game more visually appealing and less plain, which, I believed, would enhance the experience of the users of this program. This was not a part of my original project plan.
* On the Main Game Screen, I did not plan on adding an “Exit” button. However, as I was creating the program, I realized that it would be a good idea to give the users a chance to go back to the Introduction screen, to either start a new game, or to view the instructions again if they are not entirely sure on how to play.
* The time estimates made for each of the sub-tasks of this assignment (on the Gantt Chart) were fairly accurate. Overall, I had estimated that this assignment would take me 16 work days, with each work day consisting of 3 hours, and in reality, the assignment has taken me 19 days. For the most part, most of the sub-tasks took longer than expected.
  1. Did the completed software application meet all of the original requirements?
* The completed software fulfills all of the original requirements.
* This program contains all of the features that I had planned on adding, such as having multiple windows, instructions provided, sound effects and audio clips that played, a random word chosen from a list, a “hanged man” image that is updated for every incorrect letter guessed, etc.
* Some features of the completed program that were not a part of the original requirements include having various images, having an XML file that stores the users Win/Loss records and a button that is capable of clearing this XML file.
* A small feature that was a part of the original requirements, but was not included in this program was having theme music. I had a hard time trying to implement this, as the theme music seemed to overlap with the other sound effects in the game, and made these sound effects harder to hear. For that reason, I decided to remove this feature from my program.
  1. Outline successes.
* I was able to successfully implement sound effects and audio clips into my program. I was also successful in finding out how to adjust the volumes of these audio clips, as I had briefly searched the internet and quickly found a solution.
* I was able to keep my program very efficient. I was able to create methods that saved me a lot of time, and allowed my code to be shorter. For example, for each of the letter buttons (A to Z), I quickly realized how much more efficient it would be if I were to simply make one method, and just call that method for each of the 26 buttons. I believe that it was a success for me to put forth an effort to keep my code concise and efficient from the start.
* I was able to successfully implement images into my program. I had to search the internet, but I found the solution quite easily, which was to add my saved images as icons to blank JLabels. I also did not have much trouble in editing my images. I simply inserted each image I wanted to use into a blank PowerPoint, and from there, I was able to edit each of my images in many ways.
* I successfully set up all of my windows to have seamless, smooth transitions between each other. I made it so that whenever a new window was opened, the window that was previously open would automatically close, so the users would never have two windows of the program open at the same time. Also, I formatted each of the windows so that they open exactly in the center of the screen.
* I was successful in being able to have different elements overlap each other. After briefly searching the internet, I quickly learned that you could overlap elements by setting the specific JFrames layout to “null layout”. This made it easy for me to set the background image of the Introduction Screen, as well as create the hanged man image with all of the different body parts.
* I was successful in getting all of the buttons to work effectively.
* I was successful in having the random word chosen for the user to guess. I was able to effectively use the “Random '' function to choose the users mystery word from a String array full of random words.
* I was able to successfully create the set of blank dashes to represent the mystery word that the user had to guess. I had created a JLabel, and for each and every letter of the mystery word, I added “\_ ” (An underscore, then a space) to the JLabel to represent that letter.
* I was able to successfully determine if the letter guessed by the user was in the mystery word or not. I used a for loop to access each letter of the specific mystery word, and compare each letter to the letter guessed by the user, in order to determine if the specific guessed letter exists within the mystery word. I was also able to successfully update the set of blank dashes. If the letter guessed by the user exists within the mystery word, then the specific letter position(s) where the guessed letter exists would be used to update the set of dashes with that letter.
  1. Outline difficulties
* I had some trouble creating the XML file itself.
  + I was not sure how exactly to create the XML file, so I had to refer back to my previous programs that involved XML in order to figure out how to create and edit an XML file. I also used my previous programs to determine how to create the root element and how to add child elements.
  + In order to prevent the program from crashing, I had to manually create the XML file for the users beforehand. If I left it up to the users to create the file, and they decided not to, the program would crash.
* I had some difficulty setting up the XML file to store the users Win/Loss records.
  + Initially, my plan was to use an XML file to store the overall high score for all of the times that the game was played. However, I experienced trouble when I was trying to extract the current high score value from the XML file, and compare it with the current number of wins.
  + I was not able to do this, so I instead decided to use the XML file to store the Win/Loss records for every time that the program is used.
* I experienced trouble in determining how to create multiple different JFrames for this program.
  + When I began to work on this program, I searched the internet extensively on how I could create multiple JFrames within the same program, but I could not find what I was particularly looking for.
  + While I was coding in the Java Design section of the program, in the “Palette”, I found an object called “Frame”. I dragged that onto my main JFrame, and it created a new JFrame for me, exactly how I wanted it.
* Another problem that I faced was that whenever I tried to manually close one of the additional windows that I had added, by clicking the “X” button, the window would close, but the program itself would continue running. After doing research online, I found out that for each additional window that I had added, I had to set the “defaultCloseOperation” to “EXIT\_ON\_CLOSE” within the “Design” section of the program. This way, whenever any of my additional windows were manually closed, the program itself would stop running, as well.
* I experienced trouble when I was trying to set a time delay between the Hangman’s last body part (second leg) appearing on the screen, and the Lose screen appearing. I wanted there to be a delay between these two events so that the users could see the full body of the hanged man before the Lose screen appeared. After doing research online, I found out that I was supposed to use a Java Swing Timer. With this, I tried many different methods in order to create this delay, but I was not successful. So, in the end, I decided not to create this delay.
  1. Make recommendations for improvement.
* I could have put more detail into the “hanged man” image. Instead of using a stick figure, I could have designed a more sophisticated, lifelike person. Also, instead of making each body part of the man appear for every incorrect letter guessed, I could have made it more realistic, by having the hanging structure set up around the person, and then having the person hanged. Another feature that could have been added is having various facial expressions for the hanged man, one for every time that the user guessed an incorrect letter.
* I could have given the users more incorrect guesses before they lost. Some of the words that I have put into the list of random words can be quite difficult to guess, so if the users had more incorrect guesses before they lost the game, it would not be as difficult/challenging to win.
* Instead of having the users guess things that were only one word long, I could have the users guess things that were multiple words long. This way, there would be more room for complexity in what the users had to guess, and there would be more of a challenge for the users.
* I could have sorted all of the random words into different topics of words (ex. sports, locations, brands, colors, objects, etc.), and indicate which topic the specific mystery word that the users had to guess was within.
* I could have provided the users with the option to choose between different levels of difficulties, Easy, Medium or Hard. Here, the users can determine what level they believe they are at, and choose that. This makes the game more fair to all users. Some users may be very skilled at guessing words, while others may struggle, so the users having this choice would solve that problem.
* I could have created a “Hints” button, which gives the users some hints about what their specific mystery word is. There could have been a limit of, say, 10 hints for each time the program is used. However, for this to work, I would have to reduce my list of random words to about 15-20 words, as I would need to create multiple hints per word.