**Original Design and Specification:**

In order to start the hangman game, an object of the class *hangman* is created. The object, *start*, then calls its method *menu* and begins the game. The menu welcomes the player to the game and asks to either continue to play the game or quit. We felt like having an opening menu function would give the program an authentic game feel. Once the user chooses to continue to the game, the main list of 50 possible words is imported, and the *start* object is used to start the *play* function. When *play* is first called, a random word is chosen using *random\_word*. *Random\_word* picks a random number between 1 and 50, and the word at that location within the main vector of words is chosen to be the secret word.

After the secret word is chosen, the user enters their guess and it enters a series of loops to determine what to do with the entered character. First, the character is checked against the vector of previously guessed letters to see if the letter had been repeated. If it is a duplicate, the user is asked to choose a different letter, but if it passes, the vector of used letters is updated and then goes to check if the letter is in the secret word. If true, the function *update\_word* is called, which finds the position of the guessed character within the secret word, and replaces the ‘\_’ at that position with the letter and returns the updated string. The updated string is then checked against the secret word to see if they have won. If the letter is not in the secret word, the user is told and then asked for another guess. This guessing process repeats until the number of turns exceeds 10, or the secret word is solved.

**What I learned:**

Throughout the process of planning, coding, and executing the hangman game, I have learned a lot about the structure and order that games are programmed. The original idea that we had was to do an “Evil Hangman”. We started planning, which includes writing down the components that we need in the game, work out the logic that the game will have to follow, and even writing down pseudocode in a notebook. As Jess and I moved on to coding, there were more and more problems that arose within the logic we had for “Evil Hangman”, and there were more functions used for checking the “secret word” than there were for actually playing the game. One of the major problems we had was that the vector of strings we were using to choose the new secret word was being created based on the current letter being guessed, and not any of the past guessed letters. While this could have been fixed using three or more loops and another two functions, we chose to reevaluate our project and take into account the time we had left. We both came to the decision that it would be better to code a normal game of hangman because that would still use many of the techniques we learned in class and be within the project requirements.

During this process, I learned the value of taking a step back and looking at a half-finished project and deciding whether or not it is worth it to continue, considering the due date, labor time, and how much this project matters to the final grade. From a technical standpoint, I also learned a great deal about how to import text files, write “check” functions for a variety of specifications, and manipulate vectors of strings. After a lot thinking and trial and error I am proud that I figured out how to look at individual characters inside a specific word inside a vector of strings. That process was a critical part of playing the game and checking the guessed letter against the secret word.