**Base-3, 5-bit, 6-word encoding**

**for 5 Character Corpora**

**(Obfuscation with Wordle)**

Vijay Karki

2153300

Project Dissertation submitted to Swansea University in Partial Fulfilment

for the Degree of Master of Science.

Logo

Description automatically generated

Department of Computer Science

Swansea University

December 15, 2022

# **ABSTRACT**

# **DECLARATION**

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed By: Vijay Karki

Date: 15/12/2022

# **STATEMENT 1**

This dissertation is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by giving explicit references. A bibliography is appended.

Signed By: Vijay Karki

Date: 15/12/2022

# **STATEMENT 2**

I hereby give consent for my dissertation, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed By: Vijay Karki

Date: 15/12/2022

# **ACKNOWLEDGEMENT**

Table of Contents

[**ABSTRACT** i](#_Toc121925385)

[**DECLARATION** ii](#_Toc121925386)

[**STATEMENT 1** ii](#_Toc121925387)

[**STATEMENT 2** ii](#_Toc121925388)

[**ACKNOWLEDGEMENT** iii](#_Toc121925389)

[**1.** **INTRODUCTION** 1](#_Toc121925390)

[**1.1.** **Motivation** 1](#_Toc121925391)

[**1.2.** **Aim** 2](#_Toc121925392)

[**1.3.** **Objectives** 2](#_Toc121925393)

[**1.4.** **Contributions / Achievements** 2](#_Toc121925394)

[**2.** **BACKGROUND** 3](#_Toc121925395)

[**2.1.** **Wordle** 3](#_Toc121925396)

[**2.2.** **Obfuscation and Hashing** 4](#_Toc121925397)

[**2.3.** **Project Relation with Obfuscation and Hashing** 5](#_Toc121925398)

[**3.** **METHODOLOGY** 7](#_Toc121925399)

[**3.1.** **Software development life cycle** 7](#_Toc121925400)

[**3.2.** **Theoretical methods** 7](#_Toc121925401)

[**3.3.** **Project Challenges** 9](#_Toc121925402)

[**4.** **IMPLEMENTATION** 10](#_Toc121925403)

[**4.1.** **Front End Implementation** 10](#_Toc121925404)

[**4.1.1.** **User Input** 10](#_Toc121925405)

[**4.1.2.** **Random Selection** 11](#_Toc121925406)

[**4.2.** **Backend Implementation** 13](#_Toc121925407)

[**5.** **EVALUATIONS And RESULT** 17](#_Toc121925408)

[**6.** **Conclusion and Recommendation** 21](#_Toc121925409)

[**7.** **References** 22](#_Toc121925410)

**Table of Figures**

[Figure 1: An example of the game Wordle [1] 3](#_Toc121925370)

[Figure 2: An example of obfuscation in software development [4] 4](#_Toc121925371)

[Figure 3: Hashing operation from plain text to hash [5] 5](#_Toc121925372)

[Figure 4: Obfuscated matrix shown to users 6](#_Toc121925373)

[Figure 5: Waterfall model of SDLC (Tutorials Point, 2019) [10] 7](#_Toc121925374)

[Figure 6: Words that matches the most 8](#_Toc121925375)

[Figure 7: Words with no matching words 9](#_Toc121925376)

[Figure 8: Landing page of project 10](#_Toc121925377)

[Figure 9: Matrix generated by user input 11](#_Toc121925378)

[Figure 10: Matrix generated by random selection 12](#_Toc121925379)

[Figure 11: Message return with correct guess 12](#_Toc121925380)

[Figure 12: Message return with wrong guess 13](#_Toc121925381)

[Figure 13: Information provided to users on what to input in the textbox 17](#_Toc121925382)

[Figure 14: Error message is displayed when user tries to enter word that is not in the wordlist. This message is displayed in the terminal not the web page 18](#_Toc121925383)

[Figure 15: Distribution of colours in a way that there are few greens and more yellows 19](#_Toc121925384)

# **INTRODUCTION**

This project’s main focal point is to experiment with the Wordle and creating something different that is more fun and interesting. There are two types of gameplays included in the project which users can play in a nice-looking web-based user interface. This project utilizes obfuscation techniques and merges it with Wordle to achieve its purpose. This project inverses the game Wordle and presents users with obscure 6x5 color-coded matrix based on user input. User then are required to trace back the original word by looking at the matrix. By the start this project hoped to answer questions such as A) Can we create a hash using base 3 encodings inn a way that users are still able to relate to the original five-letter word? B) Can we optimize it more to increase the number of yellow and reduce greens? C) Can we use it as a way of message sharing or as a fun game between two persons?

There are two main gameplay options available for users which are user input based and random word selection. If the user wants to input a word, a color-coded matrix will generate. Users can then use the matrix to try and identify the original word they input.

The other gameplay included is random word selection. Instead of inputting word user can select random word. If this option is selected, system selects a single five letter word and generates a color-coded matrix which is presented to user. Along with matrix, users are presented with an obscure matrix of words which are generated based on word which users can use to identify single original word.

The project is built using python programming language and HTML. The program utilizes library which are already present in python such as Flask, NumPy

## **Motivation**

There have been many variants of Wordle since it came out. According to (Haripriya, 2022) [9] there are more than 75 wordle variants available to play right now. With every variant the main logic is the same, there is a word and users are given n-tries to identify the word like wordle. However, there is no variant which reverses the wordle. So, a thought came what if we reverse the wordle and present user with a colour coded matrix and asks if they can reverse the matrix into single five letter word.

## **Aim**

The main aim of this project is to take a five-letter word, obfuscate it and create a hash of base-3 encoding and present it to user in a game like user interface.

## **Objectives**

The objectives of this project are

* Design a system which takes in input of five letter word and create an obfuscated matrix
* Create a web-based user interface which interacts with users to take input and provide output
* Optimize the code to make it harder to crack word by increasing count of yellow letters and decreasing greens

## **Contributions / Achievements**

After the completion of project and all the testing and evaluations it is safe to assume that all the objectives were met. A complete application has been developed with front end to user interaction and backend to core processing which takes in input a single five letter word or from random selection and gives output a colour coded matrix along with an obscured word matrix. The code was optimized in order to increase the count of yellow and decrease the number of greens shown to users.

# **BACKGROUND**

## **Wordle**

Wordle [1] became an internet sensation back in early 2022 gaining massive user base. At its peak there were around 2 million daily users in January 2022 according to the Guardian [2]. Wordle is a word puzzle game which can be played once every day. It was designed by former Reddit employee and Software Engineer Josh Wardle and was launched in October 2021. User are given six chances to find word of the day. Words are five letter English word and are marked by colours in which green represents the correct letter in correct position, yellow represents correct in wrong position and black represents wrong letter.

During the play, users tend to focus on increasing the count of green as it gives information about the position of letters. The higher the count of greens the more and fast chances of finding the word. An example of Wordle is given in Figure 1 below.

Graphical user interface

Description automatically generated

Figure 1: An example of the game Wordle [1]

Guessing the correct word in the first try is extremely hard and the odds are 1 out of 2,315 or 0.043% according to (Levin, 2022) [6]. After entering the first letter wordle returns one of the 243 patterns. According to the (Zaiontz, 2022) [7] if the initial guess is all black, the chances of landing correct guess in second try is 1/246 meaning that of no letter is matched in first guess there are 246 possible target that matches the pattern. Moreover, if the pattern return by wordle after first guess is “\*\*YGG” (\* for black, Y for Yellow and G for Green) then, the probability is 1/3.

## **Obfuscation and Hashing**

Obfuscation is the technique which makes it difficult to understand in order to provide security. Obfuscation are being used in natural and computer languages. In computer language, obfuscation is achieved by making the code very difficult to read. This is done in order to protect intellectual property or trade secrets (Lutkevich, 2021) [3]. In case of this project, obfuscation is achieved by creating an obscure matrix of words based on the single five letter word. An example of obfuscation is given in Figure 2 below.

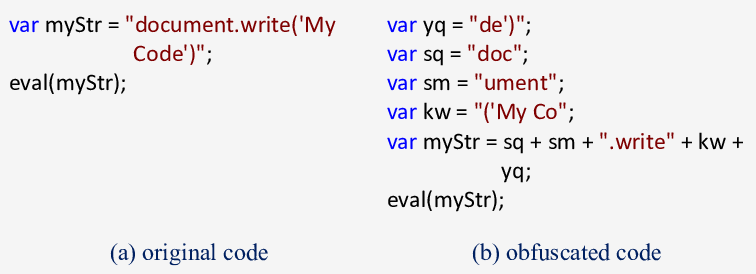


Figure 2: An example of obfuscation in software development [4]

In natural language, obfuscation is being done for many centuries. If we take ‘Pig Latin’ as an example, it is believed that this was invented by American children sometime in 1800s. In Pig Latin, beginning consonant letter is shifted to the end and added ‘ay’ vowel sound so the Pig Latin itself becomes ‘igpay atinlay’ [8].

Hashing on the other hand is a technique of converting any length of input into single fix length output based on the selected hashing algorithm. There are different types of hash length in use for example, 32-bit, 64-bit, 128-bit, 256-bit and 512-bit. The strongest hash is 512-bit hash which is the strongest one and almost impossible to crack. An example of hashing operation is given in Figure 3.

Diagram

Description automatically generated

Figure 3: Hashing operation from plain text to hash [5]

## **Project Relation with Obfuscation and Hashing**

This project implements obfuscation as well as hashing to achieve its goal. Obfuscation is achieved an obscured colour coded matrix which is displayed to user based on their input. The goal is to make it hard for user to identify the original word and not make it impossible. With the matrix users gets confused as to what the word is but certainly can reverse it back with a bit of patient and multiple tries. Following figure shows the obfuscated matrix which is shown to users.

**Graphical user interface

Description automatically generated**

Figure 4: Obfuscated matrix shown to users

Similarly, hashing is used in a form of base-3 encoding in this project. Unlike actual hashing functions which makes it impossible to crack the original values, our target is to make it hard as possible but still reversible. So, in this project the hash uses three characters 0, 1, and 2. Here, 0 represents letter is not matching, 1 represents matching letters in wrong place and 2 represents correct letter in correct place. The encoding generated in this project is “[[1, 1, 2, 0, 0], [1, 1, 1, 0, 0], [1, 0, 2, 0, 1], [1, 2, 1, 0, 0], [1, 2, 0, 1, 0], [1, 2, 0, 0, 1]]” which is in an array form.

# **METHODOLOGY**

## **Software development life cycle**

For this project Waterfall Software Development Life Cycle was chosen as the best model. There are other models which are also good such as V model and agile model, but this project does not need to be dynamic and parallel working in many phases of application. In Waterfall SDLC, each step comes after previous phase is complete, so this suits best for this project. The waterfall model has six phases which are shown in Figure 5 below.

Diagram

Description automatically generated

Figure 5: Waterfall model of SDLC (Tutorials Point, 2019) [10]

## **Theoretical methods**

The main methodology behind this project is that when user executes a program, they get an URL ‘http://127.0.0.1:5000/’. The server used in this project is pythons default server. When users enter the URL in browser, they get an index page and users can interact with the program from there.

When users enter the word, there are limited number of results that can be shown. For example, there are finite numbers of five-character English letter words available. Moreover, most of the word that are available is not heard by most people so, the list is capped to 2,315 words which are most popular and frequently used.

A test was performed to determine how many words in the list are matched with other words and how many words does not match at all. For this purpose, python’s library matplotlib was used to plot the result. In the case of words that are most matching to other words are shown in Figure 6 below. Since, each word are distinct in the list, the test was performed with words that has four characters matching. For example, ‘water’ and ‘cater’ have four matching letters which are included in this test. The test was performed with the help of loop which compares each word in the list against whole list. As result shows, some words have more than 15 matches and others has none.

Chart, histogram

Description automatically generated

Figure 6: Words that matches the most

Another test that was performed was to identify how many words has no matching characters at all. The result was a bit unexpected. It has been found that some words in the list have up to 1350 words with no matching characters. The plot created in this test is shown in the Figure 7 below.

Chart, line chart

Description automatically generated

Figure 7: Words with no matching words

## **Project Challenges**

This project utilizes from simple html codes to optimizations in order to work properly. The challenges began with researching how wordle works. At the beginning of project, Wordle has just hit the market and there were not many resources available to research. After the research, then next challenge to overcome was how to implement the theoretical approach into practical form. For the implementation, python was the best suited programming languages but, I was not very familiar with python. For this project to work, I had to learn python form scratch which was a big burden. Apart from these, there were minor challenges which were overcome easily. However, one major challenge came in a form of COVID. My wife and I both got affected by the covid and had to isolate ourselves for about a week. Having covid gave me a real pressure of finishing it on time.

# **IMPLEMENTATION**

This project is implemented using two programming languages Python and HTML. Python is used for the core working of this application which is generating encodings, optimizations, comparisons and everything else that is being processed in the background. Moreover, Flask which is a web framework based on python is used in development of web applications which interacts with users. Along with Flask, HTML is used to render the web pages and CSS is used to make it look beautiful and organized.

## **Front End Implementation**

When users execute the application, they land on a home page Figure 8 which is a basic and simple looking page that gives information about how it is played, how colour represents and what to do next. There is an input box and two buttons which executed further processes. This can be played in two ways a) a user can enter a five letter English word or b) they can select random word.

Graphical user interface, text, application

Description automatically generated

Figure 8: Landing page of project

### **User Input**

When users decide to input a word by themselves, they can input it in a text box provided and click Play button. For it to work there are certain conditions that needs to meet in order to proceed further. The conditions are: -

* The word must be a valid English letter word
* The word must not be less than five characters
* The word must not be more than five characters
* The word must also be present within the word list of the system

If the conditions are met, then the button executes further process and gives users a coloured matrix with which user can try to reverse back the original word. Since the word is provided by user, there is no further input to compare the original word with users guess. The matrix which is generated by user input is given in Figure 9. If users decide to input new word, then a new matrix will be generated based on the user input.

**Chart, treemap chart

Description automatically generated**

Figure 9: Matrix generated by user input

Should the conditions be not met, the programs give error saying some conditions are not met.

### **Random Selection**

If user decides to select random word, then the program will select one random word out of 2315 words inside the wordlist. The program then generates a colour matrix as well as an obscure words matrix which are displayed for the user. The matrix is given in Figure 10.

**Graphical user interface

Description automatically generated**

Figure 10: Matrix generated by random selection

Along with the matrix, users are provided with another text input box which takes in same types of input as previous with same conditions and returns message by comparing the original word with user inputted word. If the word is correct, it displays message which is given in Figure 11 otherwise it displays a message given on Figure 12.

Graphical user interface, application

Description automatically generated

Figure 11: Message return with correct guess

**Graphical user interface, application

Description automatically generated**

Figure 12: Message return with wrong guess

## **Backend Implementation**

Implementing backend is the real challenge in this project. The backend utilizes main library NumPy which is used in optimization. There are altogether seven functions which is divided according to different jobs. The main function to call is ‘run’ in case of user input and ‘runRandom’ in case of random selection. The main wordlist is stored in a form of text file which is rendered at the beginning of program which holds 2315 English five letter words, same as Wordle.

import sys

import numpy as np

from random import \*

green\_sq = '\U0001F7E9' # unicode for green square

yellow\_sq = '\U0001F7E8'

black\_sq = '\U00002B1B'

list\_of\_words = {}

with open("wordlist.txt") as wordlist:

    list\_of\_words = [line.rstrip() for line in wordlist.readlines()]

When the button is clicked by user, the flask framework renders input and pass it to another file called ‘Wordle.py’ which holds all the functions. The ‘run’ or ‘runRandom’ function takes one argument which is a single five letter word. The two functions ‘run’ and ‘runRandom’ works basically the same, but one has user input and other selects random word from the list.

def run(argument):

    single\_word = getWord(argument)

    result = generateEncoding(single\_word, list\_of\_words)

    newResult = optimizationFunction(result)

    newEncoding = removeDuplicate(newResult)[:6]

    print(newEncoding)

    finalString = compareEncodings(result, list\_of\_words, newEncoding)

    return (finalString, newEncoding)

single\_word = np.random.choice(list\_of\_words)

print(single\_word)

def runRandom(argument):

    global single\_word

    if argument == 'true':

        single\_word = np.random.choice(list\_of\_words)

    else:

        single\_word = single\_word

    result = generateEncoding(single\_word, list\_of\_words)

    newResult = optimizationFunction(result)

    newEncoding = removeDuplicate(newResult)[:6]

    finalString = compareEncodings(result, list\_of\_words, newEncoding)

    return (finalString, newEncoding, single\_word)

The function then calls another function ‘getWord’. Function getWord’s main job is to get the word and compare it against the list of words to check if it’s a valid English letter word and if its five characters in length. The function then either gives error or returns the word.

def getWord(argument):

    single\_word = argument

    assert not len(single\_word) != 5, "Please provide five letter word"

    if single\_word not in list\_of\_words:

        print ("Please only use valid english letter word")

        sys.exit(0)

    return single\_word

The function run then calls another function ‘generateEncoding’ which takes in two arguments single word and list of words. This function’s main purpose is the generate base-3 encoding. This function runs through loops and compares letter of single word with every single letter of words available in the lists and creates a new list of encoding which has either 0, 1 or 2. The function then returns the new created list.

def generateEncoding(single\_string, string\_list):

    result\_list = []

    for i\_word in range(len(string\_list)):

        result\_list.append([])

        current\_word = string\_list[i\_word]

        for char\_word in range(len(current\_word)):

            result = 0

            for ch in range(len(single\_string)):

                if current\_word[char\_word] == single\_string[ch]:

                    result = 1

            result\_list[-1].append(result)

    for i\_word in range(len(string\_list)):

        current\_word = string\_list[i\_word]

        for char\_word in range(len(current\_word)):

            if (current\_word[char\_word] == single\_string[char\_word]):

                result\_list[i\_word][char\_word] = 2

    return result\_list

The function ‘run’ then calls the most important function ‘optimizationFunction’ which takes in single argument. This is the main function which sorts and optimizes the encoding based on different conditions. The function checks the total value of encoding of each word and how many ones, twos and zeros are there. The functions then return a new list based on the conditions.

def optimizationFunction(argument):

    newResult = []

    sum\_of\_encoding = np.sum(argument,axis=1)

    count\_of\_zeros = np.sum(np.asarray(argument) == 0, axis = 1)

    count\_of\_two = np.sum(np.asarray(argument) == 2, axis = 1)

    for i in range(len(argument)):

        if(sum\_of\_encoding[i] <= 7):

            if count\_of\_zeros[i] <= 2:

                if (count\_of\_two[i] <= 1):

                    newResult.append(argument[i])

    return newResult

The function ‘run’ then calls a function ‘removeDuplicate’. This function’s main purpose is to remove duplicate values from the previous list generated by ‘optimizationFunctions’. This is done so the matrix provided to user does not have duplicate words.

def removeDuplicate(argument):

    result = []

    [result.append(i) for i in argument if i not in result]

    return result

Finally, the last function to call is ‘compareEncoding’ which compares encoding with list of words and provides a word matrix of 6x5 to display along with colour matrix. This function only applies in case of random selection.

def compareEncodings(result\_list, string\_list, finalEncoding):

    finalString = []

    for i\_word in range(len(finalEncoding)):

        for i in range(len(result\_list)):

            if(finalEncoding[i\_word] == result\_list[i]):

                word = string\_list[i]

            else:

                continue

        finalString.append(word)

    return finalString

# **EVALUATIONS And RESULT**

Evaluation in this project has been done mainly in two ways. One is system testing and other is user evaluation. System testing of this system has been done in each phase of development be it simply taking input from users or the finished program. Testing has been done to ensure that program functions as it is expected and there are no error or flaws that can disturbs user experience.

Testing of each button has been done to check if it functions as, it should, and the result is that its functionality is exactly as expected. Moreover, all the input boxes where user can input word have been tested to check if it taking values as required. As expected, the text box takes only a five-character English letter word and when user hovers mouse over the box it provides information as to what user should insert in the textbox (Figure 13).

Graphical user interface

Description automatically generated

Figure 13: Information provided to users on what to input in the textbox

During the testing, words that are made of random letters or not enough letters were given and the result shows that when user input words with not enough characters, it shows a message that words are not the correct one. However, when user inserts a word which is exactly five characters but is made of random letters such as ‘aaaaa’ then the programs terminate and says page is not working. This problem is mentioned in future work section and needs to be fixed.

In addition, with front end, back-end codes were also tested and evaluated along the development. All the function performs exactly as they are supposed to work. Testing has been done by providing wrong input and found that system shows error messages when there is something unexpected (Figure 14).

Text

Description automatically generated

Figure 14: Error message is displayed when user tries to enter word that is not in the wordlist. This message is displayed in the terminal not the web page

Evaluations has also been done in optimization as well. At the first stage of development the output produced was the same as user input word. For example, if user inputs ‘water’ the output is water six time. To improve result optimization was done which showed some other words as well but with maximum number of greens which was not our target. To refine the result even more, current conditions and loops were added so the output shown to users are what we aimed to show. In the current output, there are maximum number of yellow, very minimum number of green and good amount of black. The letters are distributed among colours in a way that users can trace back the original word but not so easily (Figure 15).

A picture containing calendar

Description automatically generated

Figure 15: Distribution of colours in a way that there are few greens and more yellows

In addition, with system testing, user evaluations have also been done. For this evaluation a total of six people’s feedback were taken. Since this is a small project, few feedbacks were taken just for second opinions. To get users feedback they were asked the following questions.

1. Is the system user-friendly?
2. What improvements can be made to ensure the system is more user-friendly?
3. How hard is it to identify the word?
4. Is it fun using the system?

For the evaluations, first they were given a chance to play the game and were asked the questions to get their honest opinions. The average result is the following: -

1. **Is the system user-friendly?**

Overall, the system looks neat and informative. The information provided are enough to understand the playthrough. However, there can be more added things such as balanced colours, styles and aesthetics. Although application is good it is not the most eye pleasing so, that can be improved.

1. **What improvements can be made to ensure the system is more user-friendly?**

Better colours for background and buttons would help users. In addition, using a good style would help system look more better hence, more user friendly.

1. **How hard is it to identify the word?**

It is relatively hard to identify the original word. Anywhere from two to five tries were taken to identify the word. It is just the right amount of information needed to identify the word so changing it would make it a lot harder.

1. **Is it fun using the system?**

Yes, it’s fun using the system however, it could be better if there is sharing options.

Based on user feedback, the application was updated as to make it lot better than before and user friendly.

# **Conclusion and Recommendation**

Overall, it is safe to say that the program works as expected and without any issues. The main aim of this project is to reverse the game wordle and provide users with different types of experience, which this project can provide. Instead of guessing words six times, users can now see the matrix and reverse back to the original with the help of colours and letters. Initially, the goal was to take just user input and generate a matrix however, that has been improvised to two versions which are user input and random word selections. Users have now choice to play however they like based on their preference.

There are certainly some limitations to the project. It is safe to assume that no application is perfect which applies to this as well. The main problem to solve is (as mentioned in evaluation and result section) when users input words that are not in wordlist, the program terminates, and website does not work. This needs to be fixed to show message which says the word is not present. Moreover, the program can have some added features as well such as sharing between friends. Currently, this program is limited to single user but, if there is sharing feature added to it them multiple users can play it together, something like challenges or maybe a message sharing be it just a five-letter word. Last but not lease, the program can be extended to not only a five-letter word but a whole sentence. The base-3 encoding can be used in sharing a whole message which are not so confidential.

# **References**

[1] Wardle, J. Wordle - A daily word game. [online] www.nytimes.com. Available at: https://www.nytimes.com/games/wordle/index.html [Accessed 08 Dec. 2022].

[2] Hall, R. (2022). *Wordle creator overwhelmed by global success of hit puzzle*. [online] The Guardian. Available at: <https://www.theguardian.com/games/2022/jan/11/wordle-creator-overwhelmed-by-global-success-of-hit-puzzle>.

[3] Lutkevich, B. (2021). What is obfuscation and how does it work? [online] Tech Target. Available at: https://www.techtarget.com/searchsecurity/definition/obfuscation#:~:text=Obfuscation%20mean s%20to%20make%20something [Accessed 25 Nov. 2022]

[4] JSObfusDetector: A binary PSO-based one-class classifier ensemble to detect obfuscated JavaScript code - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/An-example-of-data-obfuscation\_fig1\_279861980 [accessed 25 Nov. 2022]

[5] computersciencewiki.org. (n.d.). File:Hashing.png - Computer Science Wiki. [online] Available at: https://computersciencewiki.org/index.php/File:Hashing.png [Accessed 25 Nov. 2022].

[6] Levin, B. (2022). What Are The Odds Of Winning Wordle On The First Try? [online] Rigorous Themes. Available at: https://rigorousthemes.com/blog/what-are-the-odds-of-winning-wordle-on-the-first-try/#:~:text=The%20chances%20of%20landing%20on [Accessed 13 Dec. 2022].

[7] Zaiontz, C. (2022). *Winning Wordle in Two Tries | Real Statistics Using Excel*. [online] www.real-statistics.com. Available at: https://www.real-statistics.com/wordle-strategy/winning-wordle-in-two-tries/ [Accessed 13 Dec. 2022].

[8] Grammarist (2016). Pig Latin. [online] https://grammarist.com/. Available at: https://grammarist.com/interesting-words/pig-latin/#:~:text=Pig%20Latin%20seems%20to%20have [Accessed 13 Dec. 2022].

[9] Haripriya (2022). *Wordle Variants and Spinoffs: 76 Games Like Wordle to Play*. [online] Nerds Chalk. Available at: https://nerdschalk.com/wordle-variants-27-different-types-of-wordle-games-you-can-play/ [Accessed 13 Dec. 2022].

[10] Tutorials Point (2019). SDLC Waterfall Model. [online] www.tutorialspoint.com. Available at: https://www.tutorialspoint.com/sdlc/sdlc\_waterfall\_model.htm.