College of Applied Business

(Tribhuvan University)

A REPORT ON PYTHON'S PROJECT

HANGMAN

Submitted by:

Himani Bhattarai
Bikash Giri
Biplaw Chaudhary
Sandesh Shrestha
BSc.CSIT (2nd Semester)
Faculty: Science and Technology

Submitted to:

Kamal Acharya
Date of Submission: 2021/04/15

Acknowledgement

"It's not possible to prepare a project report without the assistance and encouragement of other people. This one is certainly no exception."

On the very outset of this report, we would like to extend our sincere and heartfelt obligation towards all the personages who have helped us in this endeavor. Without their active guidance, help, cooperation and encouragement, we would not have made headway in this project.

We are ineffably indebted to Mr. Kamal Acharya, our python instructor, for conscientious guidance and encouragement to accomplish this project.

We extend our gratitude towards College of Applied Business for giving us this opportunity.

We also acknowledge a deep sense of reverence, our gratitude towards our parents and family members of our family, who have helped us morally.

At last but least gratitude goes to all of our friends who directly or indirectly helped us to complete this project report.

Any omission in this brief acknowledgement does not mean lack of gratitude.

ABSTRACT

The project proposal was given to create an application using python. As we have seen people like being creative and guessing things, we have created a word guessing game called "Hangman".

The purpose of this game is to expose the user and players to the python implementation of word guessing game to increase their vocabulary and mind's power. People have to guess the word on the basis of given hints. People have also forgotten about basic vocabulary learnt. So we have tried to put such words that revise about basic vocabularies.

Table of Contents

Chapter One

Project Introduction

Hangman is a word guessing game. The word to guess is represented by a row of dashes, representing each letter of the word. In most variants, proper nouns, such as names, places, and brands, are not allowed. Slang words, sometimes referred to as informal or shortened words, are also not allowed. If the guessing player suggests a letter which occurs in the word, the other player writes it in all its correct positions. If the suggested letter does not occur in the word, the other player draws one element of a hanged man stick figure as a tally mark.

The player guessing the word may, at any time, attempt to guess the whole word. If the word is correct, the game is over and the guesser wins. Otherwise, the other player may choose to penalize the guesser by adding an element to the diagram. On the other hand, if the other player makes enough incorrect guesses to allow his opponent to complete the diagram, the game is also over, this time with the guesser losing. However, the guesser can also win by guessing all the letters that appear in the word, thereby completing the word, before the diagram is completed.

Tools used

We selected **python** as our programming language for this game as we were studying python and this is a part of our python's project. Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. We selected python because of its simplified syntax and with an emphasis on natural language. Also, for every task it has a build in module which made our task a lot easier.

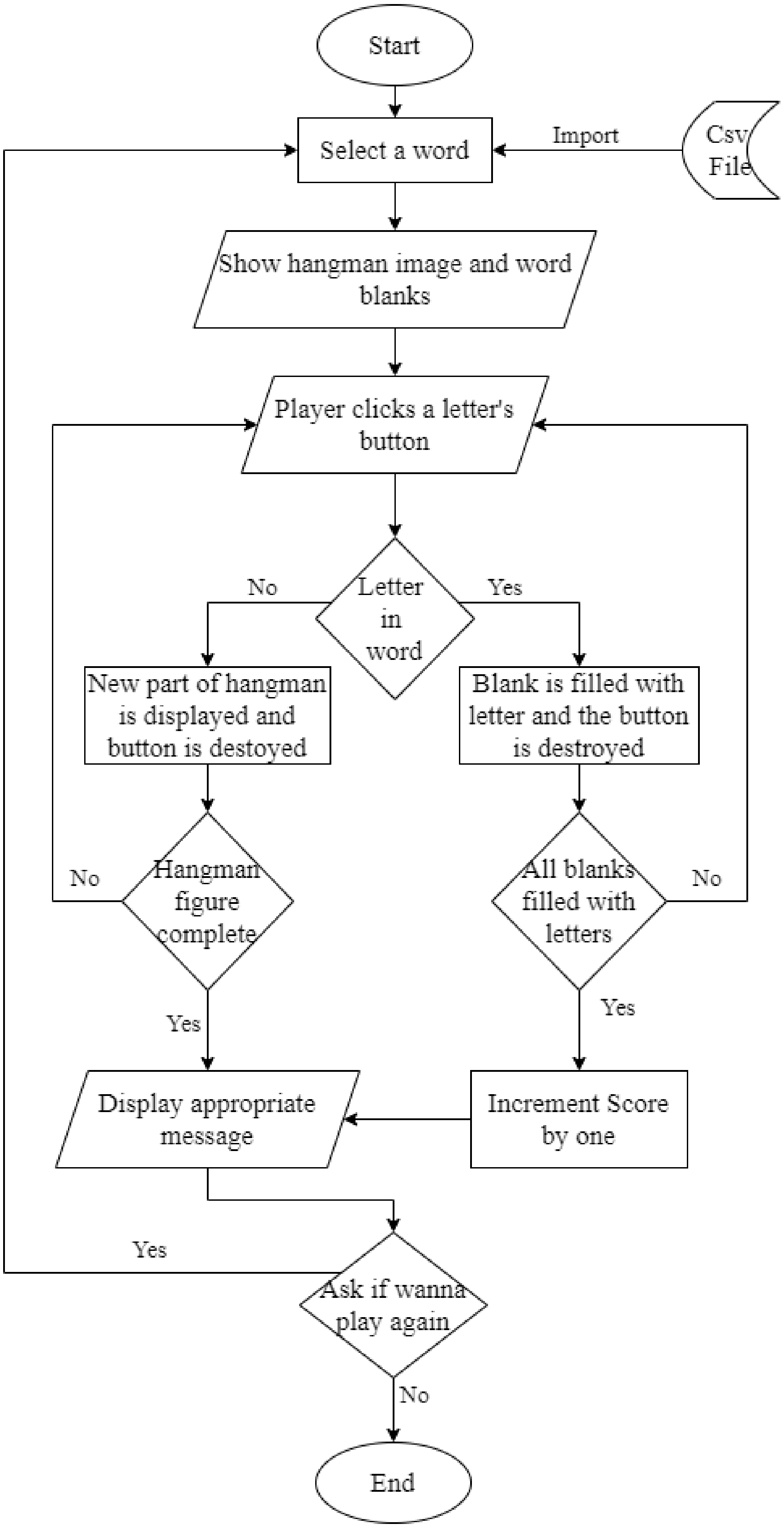
For the words list we selected a **.csv** (comma separated values) in which the word along with its hint is separated by comma.

For the GUI part, we used **tkinter**. Tkinter is a python binding to the GUI toolkit. It is the standard python interface to GUI toolkit. The name tkinter comes from Tk interface. As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands, which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

For our IDE, we used **atom** as our editor. Atom has a good syntax highlighting and interactive support for a number of programming languages through its available plugins. Atom has an amazing plugin library that is easy to use and integrates seamlessly. Also, it has git integration. So, we chose atom as our IDE.

Chapter Two

Flowchart


Start
Select a word
Import
Csv File
Show hangman image and word blanks
Player clicks a letter's button
Letter in word
No
Yes
New part of hangman is displayed and button is destoyed
Blank is filled with letter and the button is destroyed
Hangman figure complete
No
Yes
All blanks filled with letters
No
Yes
Display appropriate message
Increment Score by one
Yes
Ask if wanna play again
No
End

First of all the data in .csv file are imported to the program which then stored in a list. A word is selected at random from that list. Then, the first hangman image and the blanks which is equivalent to word length is shown which is to be guessed. Letter buttons are also displayed at this time.

Player selects a button after reading the hint of the word. If the selected letter is in the word then the blank is filled with the letter. Else, next hangman image is displayed. If the blanks are completely filled with letters the score is incremented by one and "You won" is displayed. Else if the complete hangman image is displayed then appropriate message "You lost" is displayed. If not then the player again have to select a word button till either one of the above two conditions is fulfilled.

After above condition is fulfilled, the program asks the user if he or she wants to play again. If the user selects yes then again the program selects another word randomly and the game begins again. If the player selects no then the game terminates.

Chapter Three

Implementation

In our implementation of Hangman, the computer will take on the role of the "chooser" and the human player will be the "guesser." The computer will secretly choose a word from a list which is in .csv format which will be imported to program and show the player how many letters are in the word by displaying a sequence of blanks (underscores) and the hint related to that word. Then, the player will select the words form the buttons. If the player clicks a button of letter that is in the secret word, all blanks representing an instance of that letter should be replaced by the letter and the letter button will be destroyed. If the guessed letter is not in the word at all, the player should lose a chance and a new part of the Hangman figure should appear. If the player guesses all letters in the word, he or she wins. If the Hangman figure is completed, the player loses, then the program asks if he or she wants to play again. If user selects yes then the game restarts else quits.

If the user guesses the word correctly (wins) then the game asks user if he or she wants to play again. If the player selects yes then the root window resets and the score is incremented by one and the game begins again. If the user selects no then the game terminates.

<u>Code:</u>

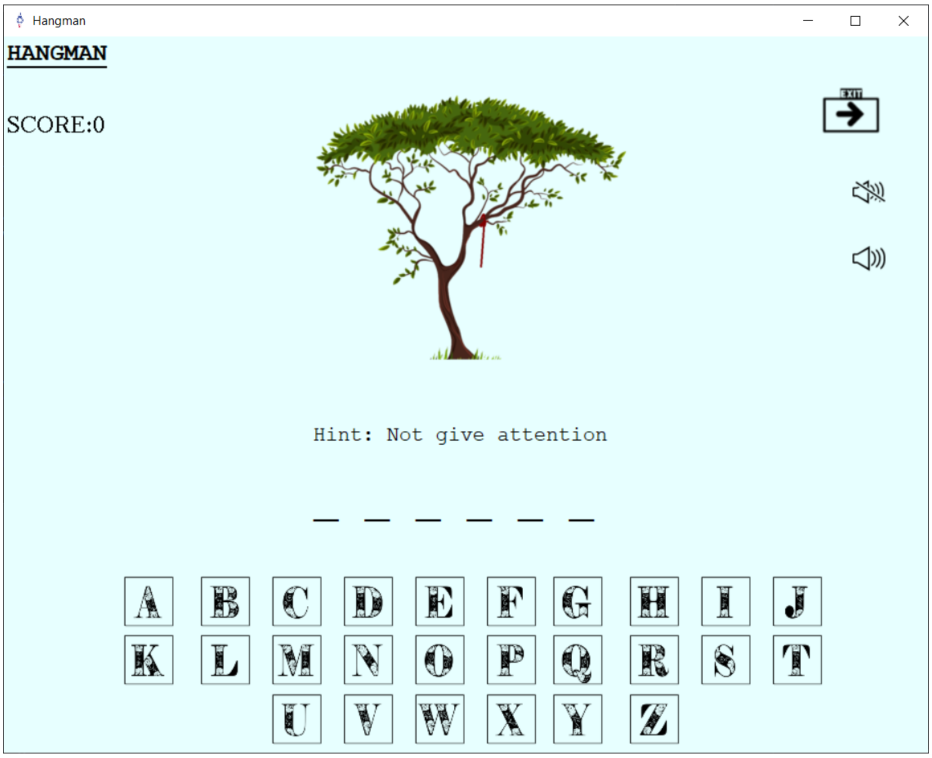
```
#---------IMPORTS-----------------
import random
from tkinter import *
from tkinter import messagebox
import csv
from pygame import mixer

mixer.init()

score = 0
run = True
```

```
#----Importing data from external file-----
f_name = "data.csv"
items = list()
with open(f_name, 'r') as csvfile:  # opens, reads and closes csv file
    handle = csv.reader(csvfile)  # csv file handler
    header = next(handle)
    for rows in handle:  # going in each row in that file
        items.append(rows)

no_of_items = len(items)  # Counts the number of items

#Generates a list of index
choices=list(range(no_of_items))
#shuffles the index
random.shuffle(choices)

#Counter to check if all index are complete
index_run=0

# main loop
while run:
    root = Tk()
    root.geometry('905x700+300+100')
    root.title('Hangman')
    root.iconbitmap(r'img/hangman.ico')
    root.config(bg='#E7FFFF')

    # Playing sound
    mixer.music.load("main.mp3")
    mixer.music.set_volume(0.1)
    mixer.music.play()

    run=False  #Destroys the root on clicking the cross
    count = 0
    win_count = 0

    #Checking if all words are complete.
    if index_run==no_of_items:
        run = False
        messagebox.showinfo(title="Completed",message="All
            complete.\n Thank you for playing.")
        root.destroy()
        exit()

    # choosing word
    index = choices.pop()
    index_run+=1
    selected_word = items[index][0]

    # Displaying Hints
    hint = 'Hint: ' + str(items[index][1])
    hint = Label(root, text=hint,bg="#E7FFFF", font=("courier", 15))
    hint.place(x=300, y=375)

    # Display's the name of the game
    title_game = 'HANGMAN'
    title_game_l = Label(root, text=title_game, bg="#E7FFFF" ,font=("courier",
18,'underline bold'))
    title_game_l.place(x=0, y=0)

    # creation of word dashes variables
    x = 250
    for i in range(0, len(selected_word)):
        x += 50

exec('d{}=Label(root,text="_",bg="#E7FFFF",font=("verdana",30))'.format(i)
)
```

```
        exec('d{}.place(x={},y={})'.format(i, x, 425))

    # letters icon
    al = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
          'w', 'x', 'y', 'z']
    for let in al:
        exec('{}=PhotoImage(file="img/{}.png")'.format(let, let))

    # hangman images
    h123 = ['h1', 'h2', 'h3', 'h4', 'h5', 'h6', 'h7']
    for hangman in h123:
        exec('{}=PhotoImage(file="img/{}.png")'.format(hangman, hangman))

    # letters placement
    button = [['b1', 'a', 105, 520], ['b2', 'b', 180, 520], ['b3', 'c', 250, 520], ['b4', 'd',
320, 520], ['b5', 'e', 390, 520], ['b6', 'f', 460, 520], ['b7', 'g', 525, 520], ['b8', 'h',
600, 520], ['b9', 'i', 670, 520], ['b10', 'j', 740, 520], ['b11', 'k', 105, 577], ['b12',
'l', 180, 577],['b13', 'm', 250, 577], ['b14', 'n', 320, 577], ['b15', 'o', 390, 577],
['b16', 'p', 460, 577], ['b17', 'q', 525, 577], ['b18', 'r', 600, 577], ['b19', 's', 670,
577], ['b20', 't', 740, 577],['b21', 'u', 250, 635], ['b22', 'v', 320, 635], ['b23', 'w',
390, 635], ['b24', 'x', 460, 635],['b25', 'y', 525, 635], ['b26', 'z', 600, 635]]

    for q1 in button:
            exec('{}=Button(root,bd=0,command=lambda:check("{}","{}"),bg="#
E7FFFF",activebackground="gainsboro",font=10,image={})'.format(q1[0],
q1[1], q1[0], q1[1]))
        exec('{}.place(x={},y={})'.format(q1[0], q1[2], q1[3]))

    # hangman placement
    han = [['c1', 'h1'], ['c2', 'h2'], ['c3', 'h3'], ['c4', 'h4'], ['c5', 'h5'], ['c6', 'h6'], ['c7',
'h7']]
    for p1 in han:
        exec('{}=Label(root,bg="#E7FFFF",image={})'.format(p1[0], p1[1]))
    # placement of first hangman image
    c1.place(x=300, y=40)

    # exit buton
    def close():
        global run
        answer = messagebox.askyesno('ALERT', 'YOU WANT TO EXIT THE
GAME?')
        if answer == True:
            run = False
            messagebox.showinfo(title="Thanks", message="Thank you for
playing.")
            root.destroy()

    def sound():
        mixer.music.pause()

    def play():
        mixer.music.unpause()

    # Displays the exit button
    e1 = PhotoImage(file='img/exit_1.png')
    ex = Button(root, bd=0, command=close,bg="#E7FFFF"
,activebackground="red", font=10, image=e1)
    ex.place(x=760, y=0)

    #The mute button
    mute = PhotoImage(file='img/mute.png')
    mute_b = Button(root, bd=0, command=sound, bg="#E7FFFF",
activebackground="light gray", font=5, image=mute)
    mute_b.place(x=830, y=135)
    #The play button
    unpause = PhotoImage(file='img/play.png')
    unpause_p = Button(root, bd=0, command=play, bg="#E7FFFF", activebackground="light gray", font=10,
                        image=unpause)
    unpause_p.place(x=830, y=200)

    # Displays the score
    s2 = 'SCORE:' + str(score)
    s1 = Label(root, text=s2, bg="#E7FFFF", font=("ms serif", 18))
    s1.place(x=0, y=70)

    # button press check function
    def check(letter, button):
        global count, win_count, run, score
        exec('{}.destroy()'.format(button))
        if letter in selected_word:
            for i in range(0, len(selected_word)):
                if selected_word[i] == letter:
                    win_count += 1
                    exec('d{}.config(text="{}")'.format(i, letter.upper()))
            if win_count == len(selected_word):
                score += 1
                answer = messagebox.askyesno('Winner', 'YOU WON!\nWANT TO PLAY AGAIN?')
                if answer == True:
                    run = True
                    root.destroy()


                else:
                    run = False
                    messagebox.showinfo(title="Thanks", message="Thank you for playing.")
                    root.destroy()
        else:
            count += 1
            exec('c{}.destroy()'.format(count))
            exec('c{}.place(x={},y={})'.format(count + 1, 300, 40))
            if count == 6:
                answer = messagebox.askretrycancel('GAME OVER', 'YOU LOST!\nWANT TO PLAY AGAIN?')
                if answer == True:
                    run =True
                    score = 0
                    root.destroy()
                else:
                    run = False
                    messagebox.showinfo(title="Thanks", message="Thank you for playing.")
                    root.destroy()



    root.mainloop()
```

Output:

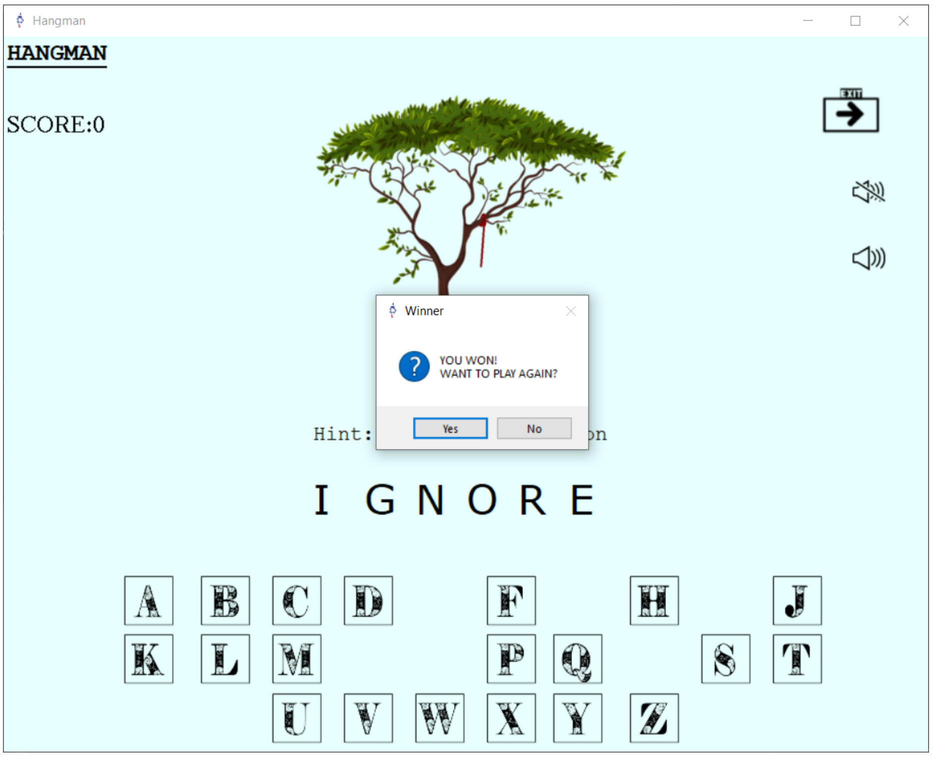
a. The program at startup:



Hint: Not give attention

_ _ _ _ _ _

b. If the selected word is true:

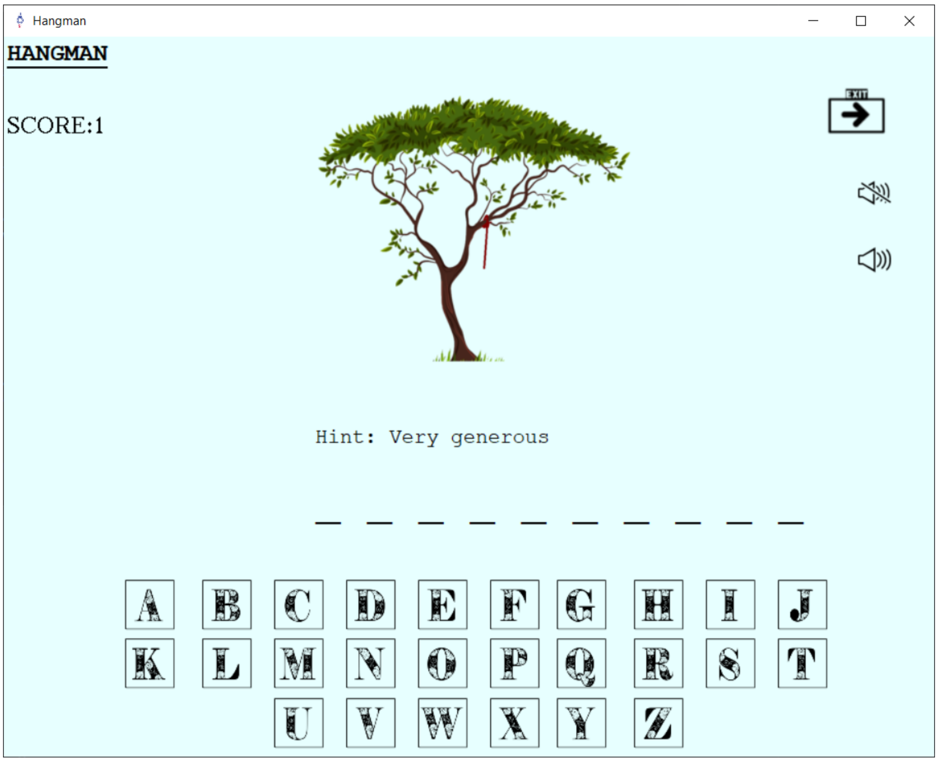


YOU WON!
WANT TO PLAY AGAIN?

Hint: ...on

I G N O R E

c. After selecting yes on play again:


Hangman

HANGMAN

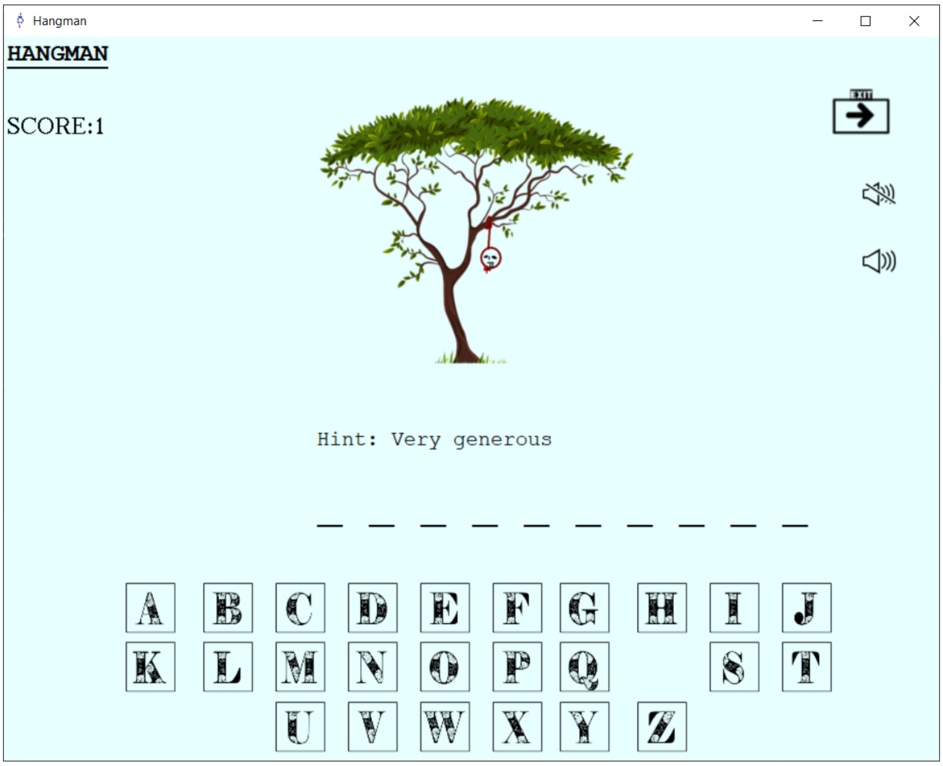
SCORE:1

Hint: Very generous

_ _ _ _ _ _ _ _ _ _

d. If chosen word is incorrect then, the parts of image is displayed as:

i) After first mistake:


Hangman

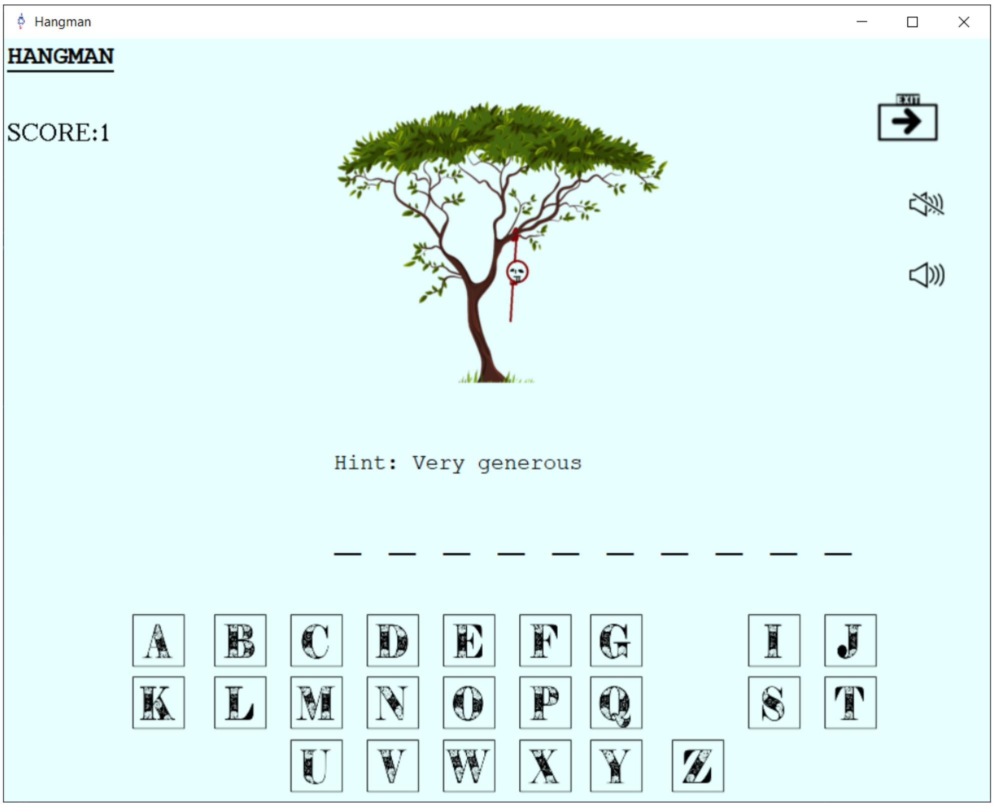
HANGMAN

SCORE:1

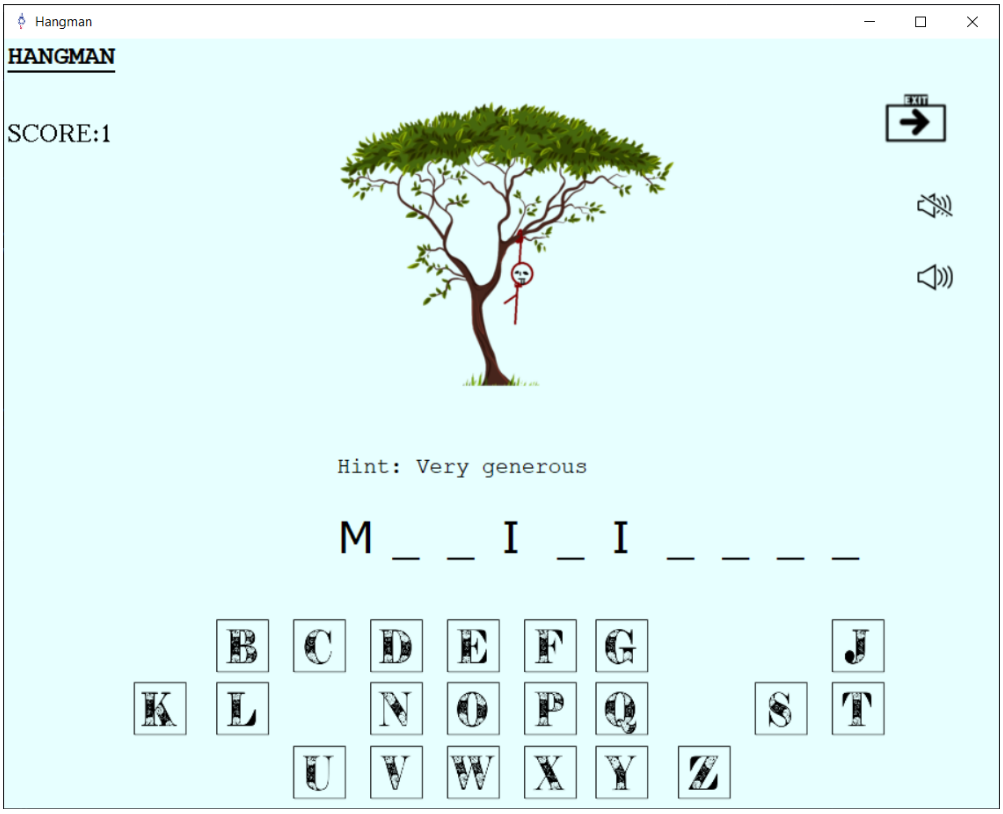
Hint: Very generous

_ _ _ _ _ _ _ _ _ _

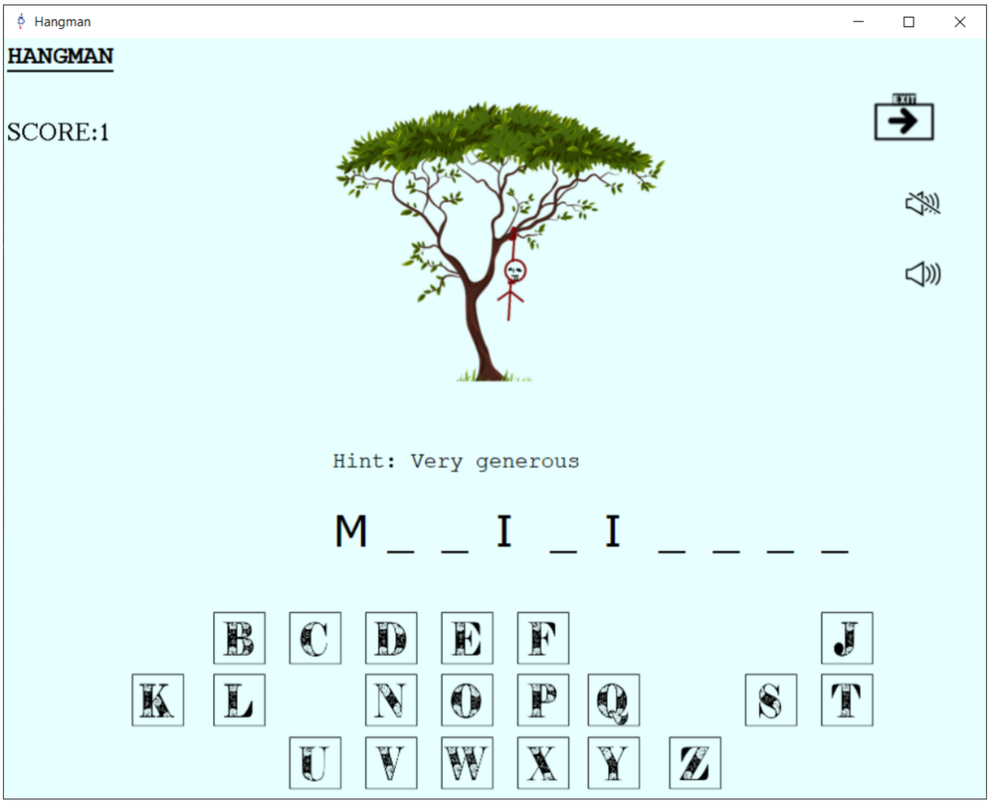
ii) After second mistake:



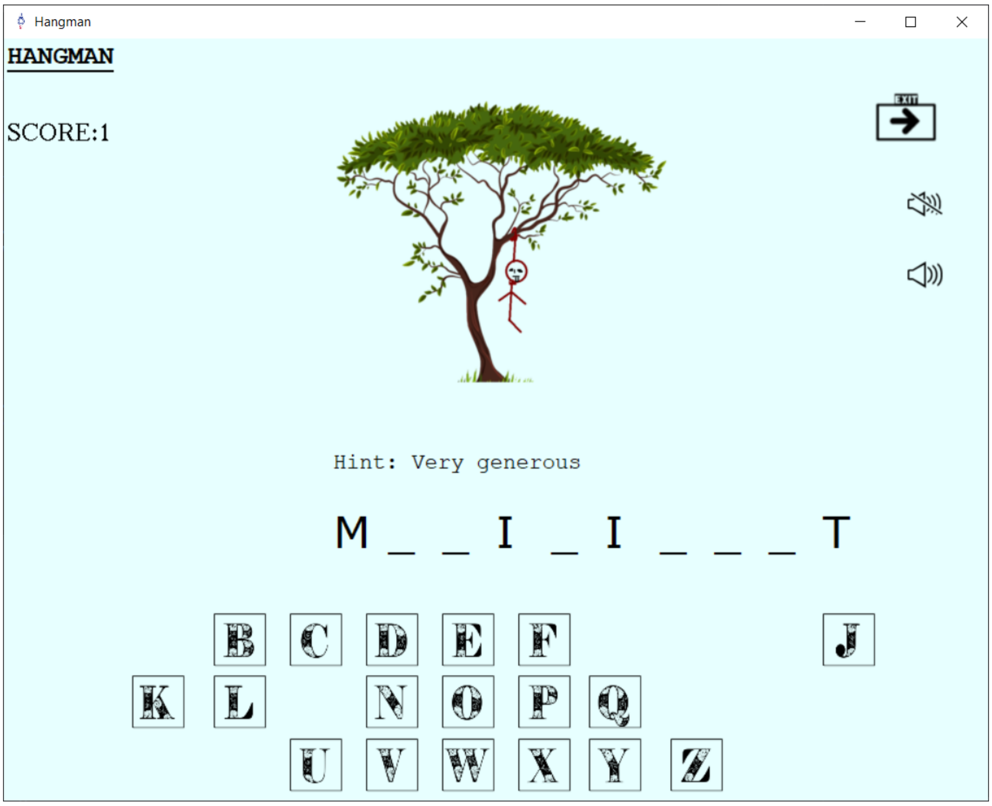
iii) After third mistake:



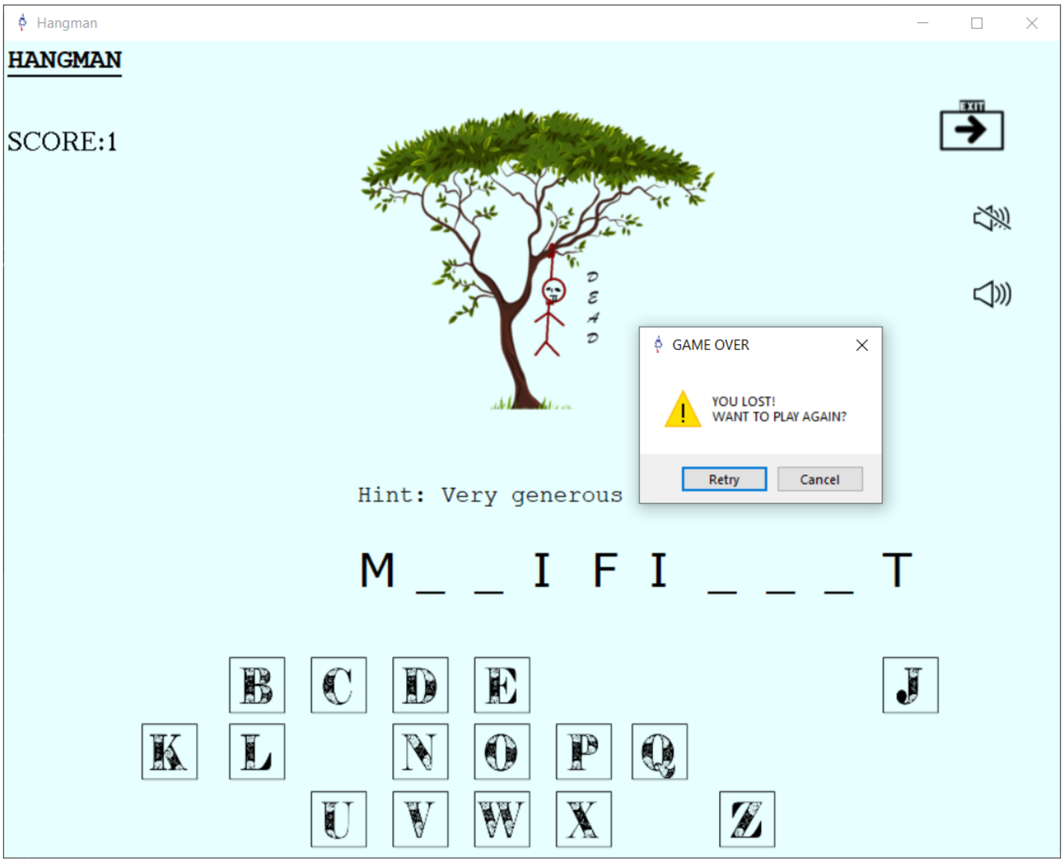
iv) After fourth mistake:



v) After fifth mistake:



vi) After sixth mistake:



YOU LOST!
WANT TO PLAY AGAIN?

Hint: Very generous

M _ _ I F I _ _ _ T

Conclusion

We learned about python and tkinter a lot while we were working on this project. We learned to manipulate data by importing then form external source, we learned about tkinter, we learned to build a GUI application. We also learned about the team effort. Overall we can say that this project has given us the idea on how to create a GUI application using python and tkinter.

Different problems arose during this project. The main problem was making sure every widget, image and hints are displayed correctly again after the player selects to play again. We'd to make sure the score is incremented by one and the game begins again with a new word resetting every image and destroyed button.

To overcome this problem, we created the root window inside a loop along with all the widgets. So when the player selects to play again the root is destroyed and another root is created with all the elements in it.

This game is not a perfect game. Since it is a prototype a lot of features we wanted to add didn't make it to final version like difficulty levels. There are many places where improvements can be made. For instance, the root is destroyed and then created again if play again is selected. A better way could be the root to be stable, and only the widgets inside the root are reset.

This project taught us a lot about application development using python and tkinter and pushed all the skills we learnt in college to test. It was nice to create a project that put everything we learned to the test so that we can prepare ourselves for the future.