

HANGMAN GAME

A Mini Project Report
Submitted in Partial fulfillment for the award of
Bachelor of Technology in Artificial Intelligence and Data Science

Submitted to
RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA
BHOPAL (M.P)



MINI PROJECT REPORT

Submitted by
Smriti Vishwakarma [70]

Under the supervision of
Devendra Singh Rathore
Assistant Professor



Department of Artificial Intelligence and Data Science
Lakshmi Narain College of Technology, Bhopal (M.P.)
Session 2022-23



LAKSHMI NARAIN COLLEGE OF TECHNOLOGY, BHOPAL

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA
SCIENCE**

CERTIFICATE

This is to certify that the work embodied in this project work entitled ”**HANGMAN GAME**” has been satisfactorily completed by the **SMRITI VISHWAKARMA**[70].It is a bonfire piece of work, carried out under the guidance in **Department of Artificial Intelligence and Data Science, Lakshmi Narain College of Technology, Bhopal** for the partial fulfillment of the **Bachelor of Technology** during the academic year 2022-23.

Guided By

Devendra Singh Rathore
Assistant Professor

Approved By

Dr. Bhupesh Gour
Prof. & Head
Department of Artificial Intelligence and Data Science



LAKSHMI NARAIN COLLEGE OF TECHNOLOGY, BHOPAL

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA
SCIENCE**

ACKNOWLEDGEMENT

We express our deep sense of gratitude to Prof. Devendra Singh Rathore (Guide) department of Artificial Intelligence and Data Science, L.N.C.T., Bhopal. Whose kindness valuable guidance and timely help encouraged me to complete this project.

A special thank goes to Dr. Bhupesh Gour (HOD) who helped me in completing this project work. She exchanged her interesting ideas & thoughts which made this project work successful.

We would also thank our institution and all the faculty members without whom this project work would have been a distant reality.

Smriti Vishwakarma [70]

S.NO.	TOPIC	PAGES
1.	Introduction	1-2
2.	Literature Survey	3-5
3.	Mini objective & scope of project.	6-13
4.	Problem Analysis and requirement specification	14-15
5.	Detailed Design (Modeling and ERD/DFD)	17-18
6.	Hardware/Software platform environment	19
7.	Snapshots of Input & Output	20-21
8.	Coding	22-27
9.	Project limitation and Future scope	28
10.	References	29

CHAPTER 1

INTRODUCTION

ABOUT THE GAMES

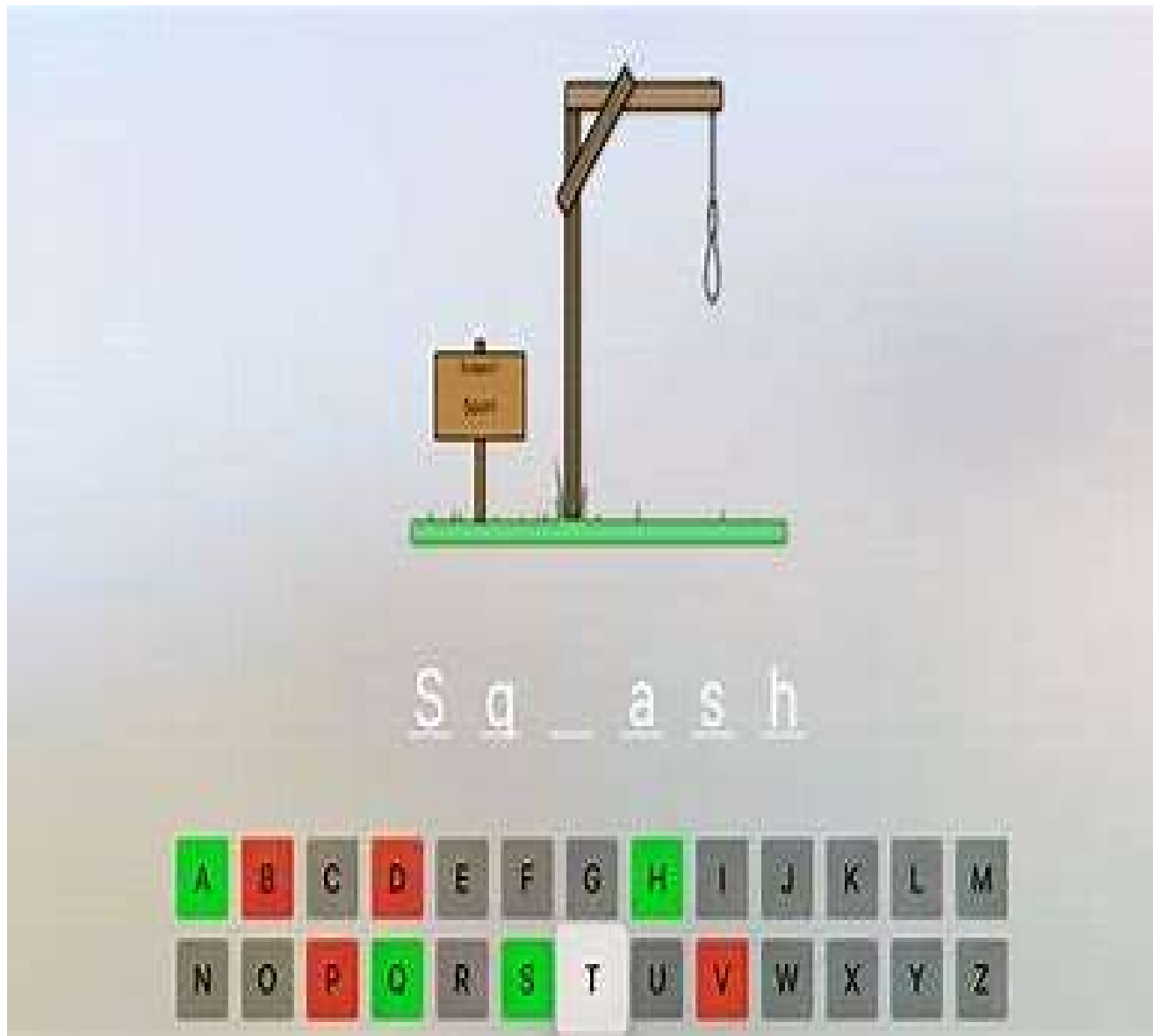
Simplified hangman game .in this version of the game, the computer program randomly chooses a target string and asks the user to suggest letters that occur in the string and asks the user to suggest letters that occur in the string. after each guess, the program shows the user a version

Of the target string that replaces letters that have not been guessed with stars (*)

Along with a count of the number of wrong guesses made so far.

The game is over either when the user wins by correctly guessing the string or losses by making more than assigned incorrect guesses. Note that the space character in the “hello World” string is included in the initial guess string, because the user only needs to guess alphabetic characters.

also, notice that the game matched a guess of the lower-case letter ‘h’ With the upper-case letter ‘H’, and preserved the case of the letter from the original string.



CHAPTER 2

LITERATURE SUEVEY

INTRODUCTION

Hangman is a [guessing game](#) for two or more players. One player thinks of a [word](#), [phrase](#) or [sentence](#) and the other(s) tries to guess it by suggesting [letters](#) within a certain number of guesses. Originally a [Paper-and-pencil game](#), there are now electronic versions.

Hangman is a paper and pen word game where you have to guess the right term. It's a perfect game to get to know new or uncommon words. You can either play it with your friends or your family. Hangman is also ideal if you're a teacher or want to teach children or adults a particular language. Check our different lists out: We have collected easy, funny, hard as well as themed hangman word

HISRORY

Though the origins of the game are unknown, a variant is mentioned in a book of children's games assembled by [Alice Gomme](#) in 1894 called Birds, Beasts, and Fishes. This version lacks the image of a hanged man, instead relying on keeping score as to the number of attempts it took each player to fill in the blanks.

A version which incorporated hanging imagery was described in a 1902 [Philadelphia Inquirer](#) article, which stated that it was popular at "White Cap" parties hosted by [Vigilance Committees](#) where guests would wear "white peaked caps with masks".

VARIANTS

As the name of the game suggests, the [diagram](#) is designed to look like a [hanging](#) man. Although debates have arisen about the game,^[5] it is still in use

today. A common alternative for teachers is to draw an [apple tree](#) with ten apples, erasing or crossing out the [apples](#) as the guesses are used up.

Some modifications to game play ([house rules](#)) to increase the difficulty level are sometimes implemented, such as limiting guesses on high-frequency [consonants](#) and [vowels](#). Another alternative is to give the definition of the word; this can be used to facilitate the learning of a [foreign language](#).

STRATEGY

The fact that the twelve most commonly occurring letters in the [English language](#) are [e-t-a-o-i-n-s-h-r-d-l-u](#) (from most to least), along with other [letter-frequency](#) lists, are used by the guessing player to increase the odds when it is their turn to guess. On the other hand, the same lists can be used by the puzzle setter to stump their opponent by choosing a word that deliberately avoids common letters (e.g. *rhythm* or *zephyr*) or one that contains rare letters (e.g. *jazz*).

Another common strategy is to guess vowels first, as [English](#) only has five vowels ([a](#), [e](#), [i](#), [o](#), and [u](#), while [y](#) may sometimes, but rarely, be used as a vowel) and [almost every word has at least one](#)

DERIVATIONS

The American game show [Wheel of Fortune](#) was inspired by hangman. [Merv Griffin](#) conceived of the show after recalling long car trips as a child, on which he and his sister played the game.

[Brazil](#) also had a show in the 1960s and again from 2012–2013 called 'Let's Play Hangman', hosted by [Silvio Santos](#). Brazil would later get its own version of Wheel of Fortune, running from 1980 to 1993, again from 2003 to 2012 (during which the new Let's Play Hangman aired), and again since 2013 to the present. These shows were also hosted by Santos.

In July 2017, the [BBC](#) introduced a game show of its own called [Letterbox](#), which is also based on hangman.

Objectives

Completing this project will solidify your understanding of character array and cell array. You will also work with a text file and create graphics.

3.1 The Game “Hangman”

This is an old game that sadly has a rather unpleasant name (and matching graphic). Another name for the game is “Gallows”, which is also of questionable taste . . . This game often is used for learning the vocabulary of a language. If you don’t know the game, play a game or two on <http://www.manythings.org/hmf/8997.html>. Next, check out the demo video, available on the course website, of our “Hangman” game.

Our version of “Hangman” uses words chosen from a list of commonly used English words. A round of the game begins with a randomly chosen word (hidden from the player except for the length). The player keeps guessing until seven wrong guesses have been made or until the word is completed, whichever happens first. At the end of a round, the user is prompted to indicate whether he/she wants to play again. The game ends when the player indicates that he/she wants to quit. You will build this program in a modular fashion by writing four functions:

- **hangman**: the main function that starts the game, handles user interaction, and creates and updates the figure. This function calls the remaining three functions.
- **get Words**: read the data file and return the words in a cell array
- **new Word**: randomly select a word that hasn’t been used before from the cell array

- `findLetterInWord`: locate a letter in a word You may write subfunctions as appropriate.

You can use the general built-in functions that we have used in the past such as `length`, `rand`, `plot`, . . ., etc., but only the following file and string handling functions are allowed: `fopen`, `fclose`, `feof`, `fgetl`, `char`, `str2double`, `strcmp`, `lower`, `upper`, `isletter`. Additionally, there are two general built-in functions introduced in Insight that you may want to use: `all` and `any`. Use `help` in MATLAB if you want to learn more about them. Also, be sure to read the 1-page document `File i/o example`, posted along with the notes of Lecture 18; it will help you with this project. Do not use functions `find`, `strfind`, or `findstr`.

3.2.1 GETTING THE DATA

The file `popularWords.txt`, available on the course website, contains more than 5000 popular words in the English language. Here're the first few lines in the file: Most popular words in the English Language From Englishclub.com
columns 1-5 is the word number columns 9 and on contains the word
00001 the
00002 be
00003 and
00004 of
1 The words of interest appear on the “numbered” lines (starting with the line numbered ‘00001’ that contains the word ‘the’ in this file); the other non-numbered lines are considered the file header and should not be used as word data for the game. You can see the whole file by using any plain text editor (such as Notepad) or even MATLAB (just double-click on the filename in MATLAB). In our game of Hangman, we use only the words that do not contain any punctuation marks (hyphen, apostrophe, etc.) and that are at least five letters long. Do not change the case of any letter from upper to lower or vice versa. Implement the following function as specified:

3.2.2 Selecting a word

Implement the following function as specified:

function [word, updatedUsedWords] = newWord (C, usedWords)

% word is randomly chosen from cell array C but is not in cell array
usedWords.

% C is a 1-d cell array of strings; C is not empty.

% usedWords is a (possibly empty) 1-d cell array of strings.

% word is a string randomly selected from C; word is not in cell array
usedWords.

% updatedUsedWords is a cell array of strings; it is usedWords with one extra
cell

% containing word.

% Assume C contains many more different strings than used Words does.

Recall that C may contain the same word in multiple cells; you need to make
sure that the returned word is not identical to any string in usedWords.

3.2.3 Searching for a letter in a word

Implement the following function as specified:

```
function [found, tf] = findLetterInWord (let, word)
```

% Locate a letter (let) in a word (word), regardless of case.

% let is a char scalar. word is a 1-d char array (string).

% found is 1 if the letter is found; otherwise found is 0.

% tf is a vector that has the same length as word; for each valid index k,

% tf(k) is 1 if word(k) is let, regardless of case; otherwise tf(k) is 0.

Consider this example: let stores the scalar 'a' and word stores the string
'American'. Then the function returns in found the value 1 (true) and in tf the
vector [1 0 0 0 0 0 1 0].

3.2.4 Putting together the game By

making effective use of the above functions, implement function hangman as
specified:

```
function hangman(frame)
```

% Run the game hangman using words that are chosen from the file named by

% The string in frame.

% The game begins with the gallows drawn and dashes are displayed above the

% gallows to indicate the number of characters in the word. Each time the
 % user guesses an incorrect letter; a body part is added to the figure. A
 % round of the game ends when the user has made seven incorrect guesses or
 % When the user has guessed the word correctly, whichever occurs first.
 % Throughout the game, display the word status in the title area of the
 % figure window and update the hangman figure as appropriate.
 % When a round ends, display in the title area of the figure window a
 % message indicating the word and whether the round is won or lost. In the
 % Command Window, prompt the user about whether to play again.
 % Words should not be repeated in a game; you can assume that the user
 % chooses to play again far fewer times than there are available wo

3.3. SCOPE

This game can have varied applications in the context of *word formations* and **puzzles**. Its knowledge can be valuable to many other games like **CROSSWORD PUZZLES**, **WHEEL OF FORTUNE**, **SCRABBLE**. We can also have an **investigation of very popular and commonly used letters** in most of the words. Make a frequency distribution in graph out of it. The underlying mathematical concepts are **Data Collection and Analysis, Presentation and Interpretation** which can have lot of implications in language processing and study of graphs and testing conjectures. Also, we can find out that the most popular letter in the English language is **"e"**. The letter frequency of all letters in the English language is: **e t a o i n s r h l d c u m f p g w y b v k x j q z**.

CHAPTER 4

PROBLEM ANALYSIS & REQUIREMENT

PROBLEM STATEMENT

Hangman is a paper and pencil guessing game for two or more players. One player thinks of a word and the other tries to guess it by suggesting the letters. The word to guess is represented by a row of dashes, giving the number of letters. If the guessing player suggests a letter which occurs in the word, the program writes it in all its correct positions. If the suggested letter does not occur in the word, the other player draws one element of the hangman diagram as a tally mark. The game is over when

SOLUTION DESIGN

The gaming code will mainly contain the **class Hangman** which will provide the list of good letters as well as the no. of chances given to a user. I am planning to use tinker toolkit for GUI. Incorporation of some widgets will better the result screen , so that , the user can proceed in the game with no confusion. The overall architecture can be thought of having *four* main parts which consist of the following functionalities: -

- 1) Formulating a word list (with or without a hint) and store them in a data structure with the list of all 26 alphabets of English Language.
- 2) The actual method which does the logical reasoning, whether the

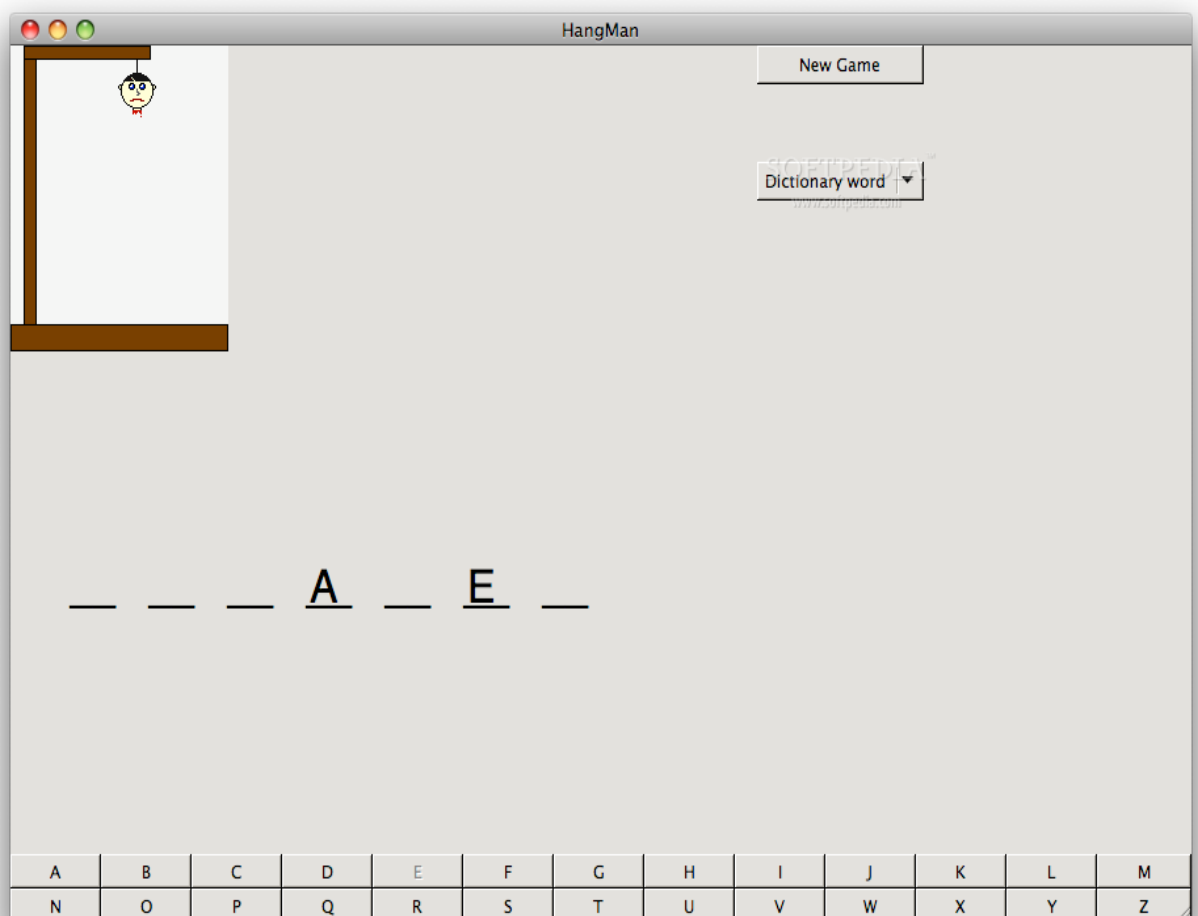
letter exists or not , if yes , write it down at all the places else strike off a lifeline. This forms the main part of the code.

3) Final word to be displayed if guessed wrongly else, interactive message saying that "The Player is the winner"

4) Finally, the GUI coding , user interactive screen which will mainly prevail during the code output.

I am using the **glib** and **gtk+** libraries which are already available. Also planning to utilize the **built in classes** and **submodules** of tinker toolkit as and when required.

Implementation Plan

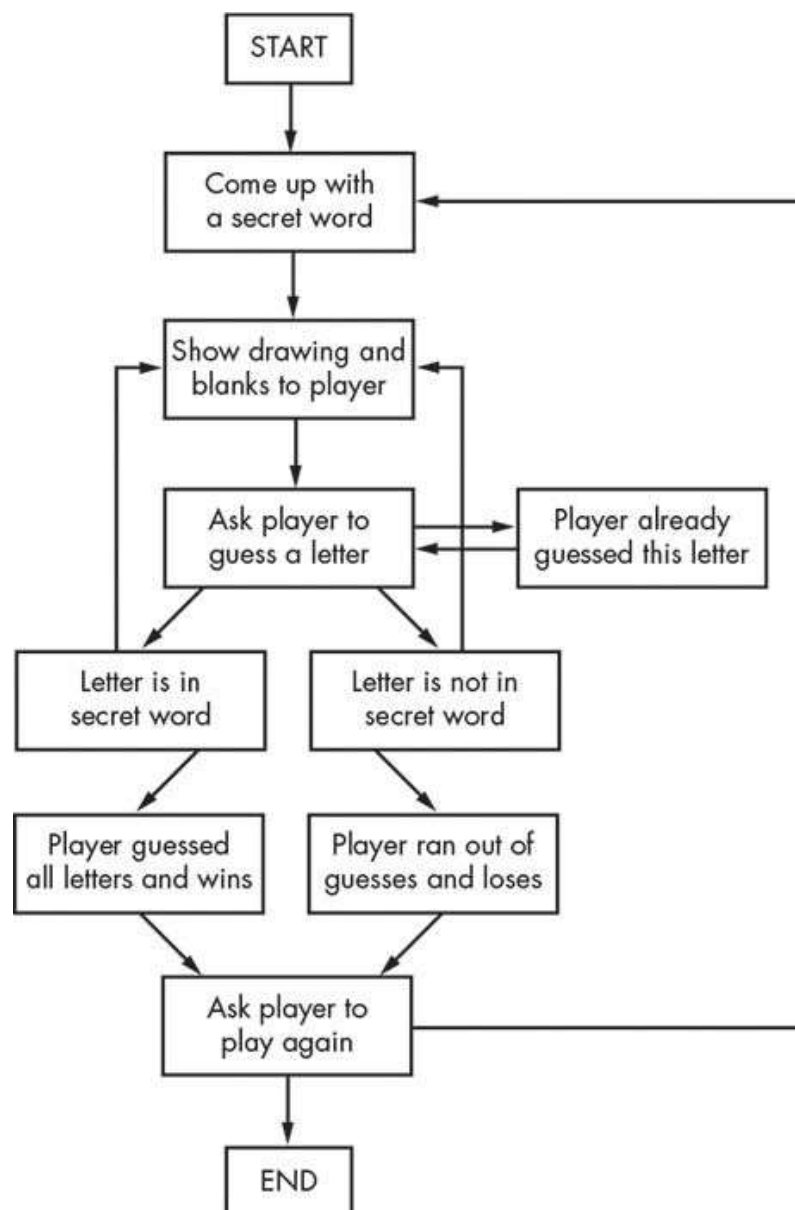


I have decided to use **PYTHON** language and specifically **Tkinter** toolkit for the GUI package. Initially I was confused between **Tkinter** and **wxpython**, as python itself is very new to me. But then, thought of utilizing the classes and modules of this package for my gaming environment. This game will mainly play with the 26 letters of the English language, so my input data will be just letters and as output the player comes to know whether he has won, or else if lost, what was the word which he missed to guess correctly. Thus, there will be a list of words in store with the program from which a player will be asked to guess a word. My estimated lines of code are **200** but it may be little more depending upon how much well I can program. I need 2 weeks' time for the project as I am new to python and specially GUI code. Currently I have only done some homework about what my project will do , exactly what it will accomplish ,but I am yet to start with coding. I hope and will try my best to complete it before the scheduled deadline.

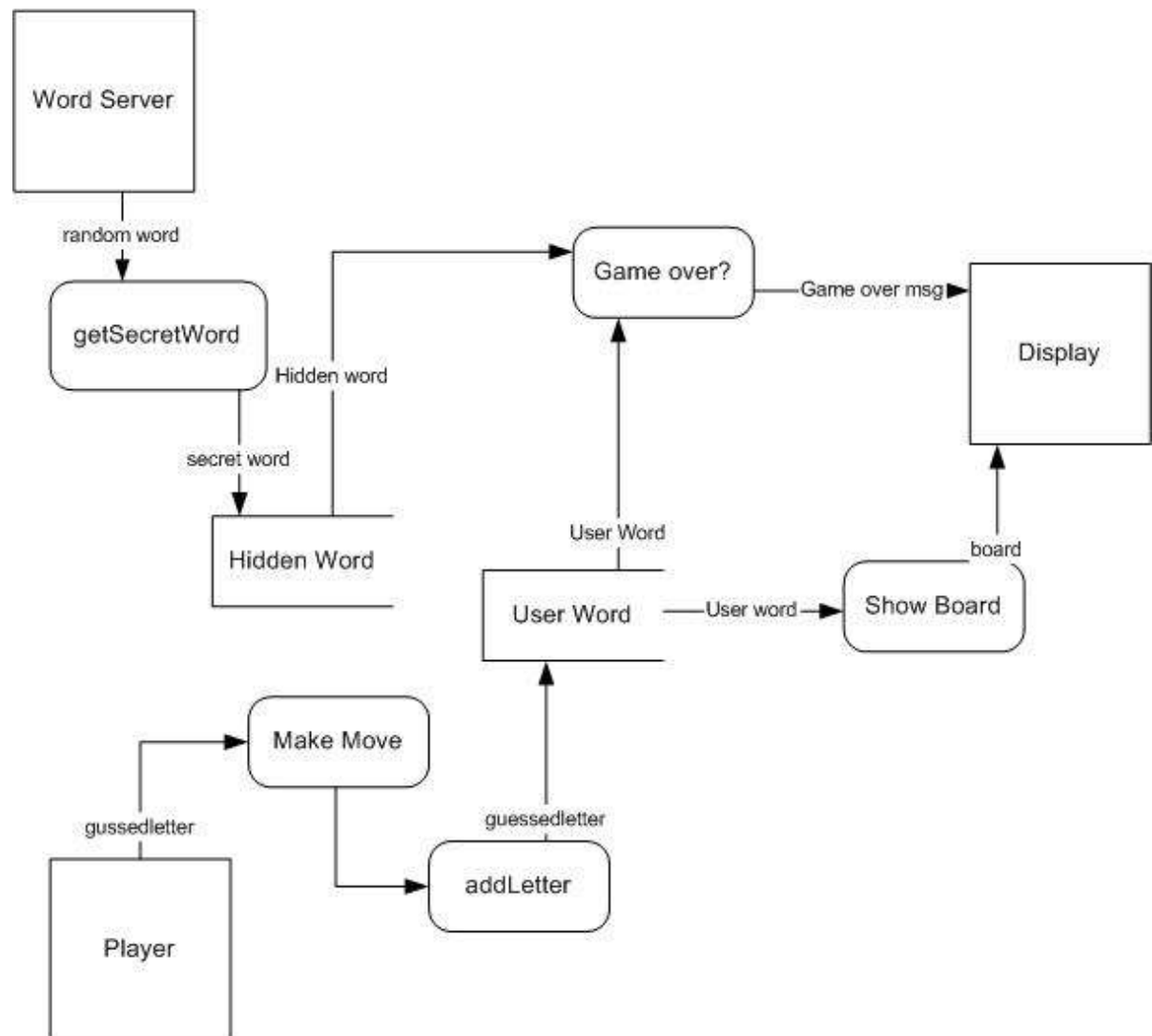
CHAPTER 5

DETAILED DESIGN (MODELING AND ERD/DFD)

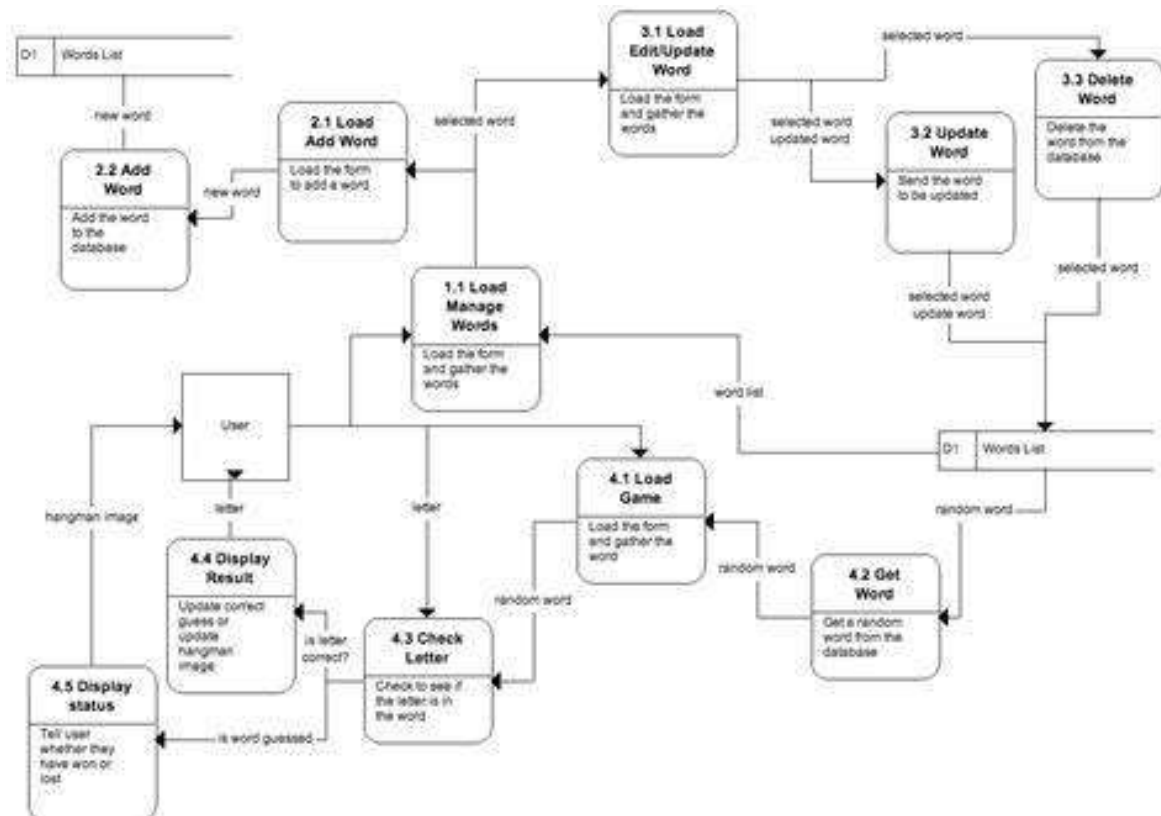
FLOW CHART



SRS



DFD (DATA FLOW DIAGRAM)



CHAPTER 6

HARDWARE /SOFTWARE PLATFORM ENVIRONMENT

HARDWARE REQUIREMENT

RAM: 256MB OR MORE

VRAM: 256 OR MORE

HARD DISK: SATA 40 GB OR ABOVE

GRAFICAL CARD: INTEL HD GRAFPHCS

SOFTWARE REQUIREMENT

WINDOW OPERATING SYSTEM

PYTHON

TKINTER-PYTHON INTERFACE

CHAPTER 7

SNAP SHORT OF INPUT & OUTPUT

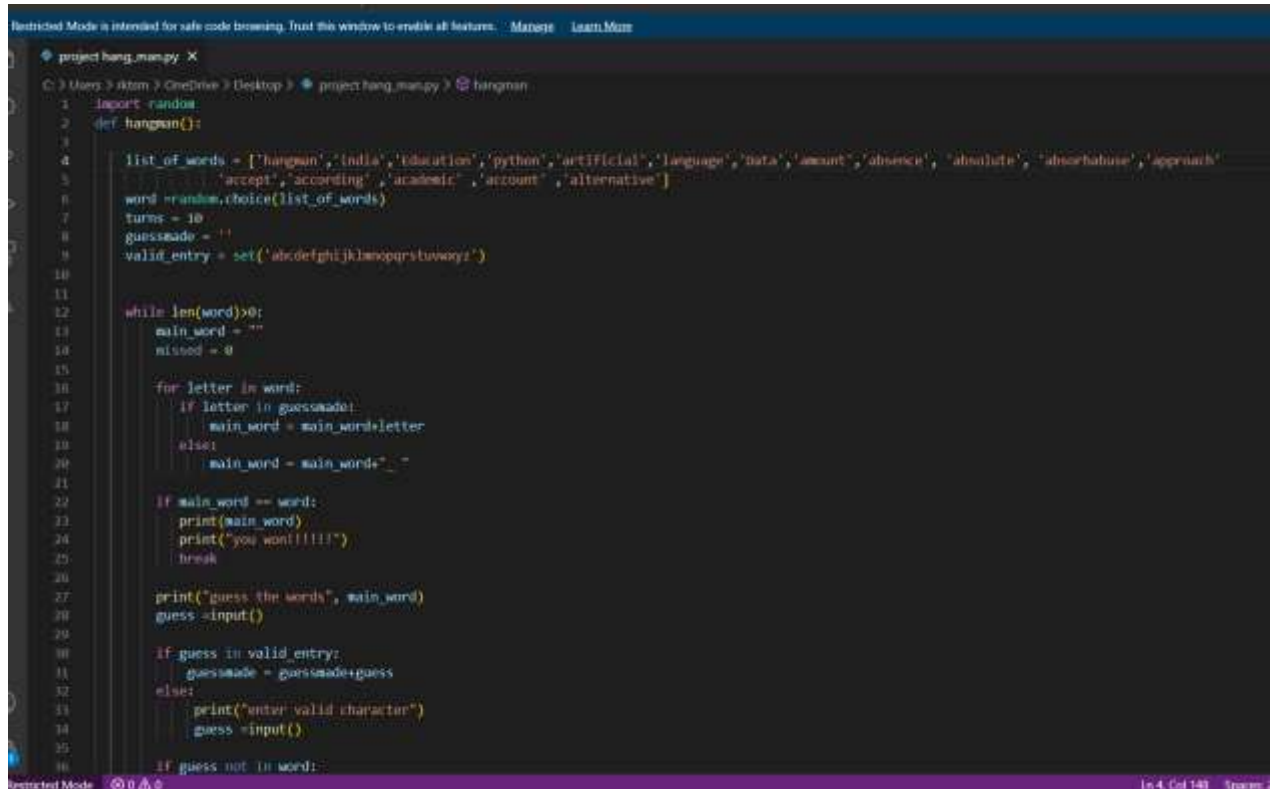
INPUT AND OUTPUT

```
enter your name smriti
welcome  smriti !
-----
try to geuess the word in less than 10 attempts
guess the words _ _ _ _ _
i
guess the words i _ _ i _
d
guess the words i _ di _
n
guess the words indi _
a
india
you won!!!!!!
>>
```



 A D O I D

PROGRAM



```
Restricted Mode is intended for safe code browsing. Trust this website to enable all features. Manage Learn More

project hangman.py X
C:\Users> cd C:\Users> Desktop> project hangman.py> hangman
1 import random
2 def hangman():
3
4     list_of_words = ['hangman','india','education','python','artificial','language','data','account','absence','absolute','absorbent','approach',
5                     'accept','according','academic','account','alternative']
6     word = random.choice(list_of_words)
7     turns = 10
8     guessmade = ''
9     valid_entry = set('abcdefghijklmnopqrstuvwxyz')
10
11
12     while len(word)>0:
13         main_word = ""
14         missed = 0
15
16         for letter in word:
17             if letter in guessmade:
18                 main_word = main_word+letter
19             else:
20                 main_word = main_word+"_"
21
22         if main_word == word:
23             print(main_word)
24             print("you won!!!!")
25             break
26
27         print("guess the words", main_word)
28         guess = input()
29
30         if guess in valid_entry:
31             guessmade = guessmade+guess
32         else:
33             print("enter valid character")
34             guess = input()
35
36         if guess not in word:
```

CODE EXPLANATION

1. The code starts by importing the random module.
2. This module provides a way to generate random numbers.
3. Next, the code creates some Words, which is a list of fruit names.
4. The list is split into spaces using the string ' ', and then each space is replaced with a letter.
5. Next, the code randomly selects a secret word from our some Words list.
6. This word will be used as the input for the game later on.

7. The next part of the code checks to see if the user has entered an alpha character (a letter that appears in front of other letters).
8. If not, then they are asked to enter only a letter.
9. If they enter an alpha character, then it's assumed that they to guess at another letter in word.
10. So, this part of the code checks to see if guess matches One of the letter word.
11. If it does, then chances is updated and flag is set to 1.
12. Otherwise, chances is decreased by 1 and flag remains at 0.
13. The next part of the code tries to guess at another letter in word.
14. If guess isn't valid (i.e., it doesn't match any of the letters in word), then `print ()` prints out all empty spaces for letters in word, and
15. The code starts by importing the random module.
16. This module provides us with a number of useful functions, one of which is the choice function.
17. This function allows us to randomly choose a secret word from our list of words.
18. Next, we create some variables which will be used throughout the program.
19. These include some Words , word and letter Guessed .
20. letter Guessed will store the letter guessed by the player, while chances will store the number of times that the player has correctly guessed the word so far.
21. correct will keep track of how many letters have been guessed so far and flag will indicate whether or not the player has guessed the word correctly.
22. We then start looping through our list of words and randomly choosing a secret word from it.


```
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

project hang_man.py X
C:\Users> rktm > OneDrive > Desktop > project hang_man.py > ...
65     print("      / \      ")
66     if turns==4:
67         print("4 turns are left!!!!")
68         print("-----")
69         print("      \ o      ")
70         print("      |      ")
71         print("      / \      ")
72     if turns==3:
73         print("3 turns are left!!!!")
74         print("-----")
75         print("      \ o /      ")
76         print("      |      ")
77         print("      / \      ")
78     if turns==2:
79         print("2 turns are left!!!!")
80         print("-----")
81         print("      \ o / |      ")
82         print("      |      ")
83         print("      / \      ")
84     if turns==1:
85         print("1 turns are left!!!! hangman on his last breath")
86         print("-----")
87         print("      \ o / _|      ")
88         print("      |      ")
89         print("      / \      ")
90     if turns==0:
91         print("you loose")
92         print("you let a good man die")
93
94
95     name =input("enter your name")
96     print("welcome",name,"!")
97     print("-----")
98     print("try to geuess the word in less than 10 attempts")
99     hangman()
100
```

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. [Manage](#) [Learn More](#)

project hang_man.py X

C:\Users> rktsm > OneDrive > Desktop > project hang_man.py > hangman

```
35
36     if guess not in word:
37         turns = turns-1
38
39     if turns==9:
40         print("9 turns are left!!!!")
41         print("-----")
42
43     if turns==8:
44         print("8 turns are left!!!!")
45         print("-----")
46         print("      o      ")
47
48     if turns==7:
49         print("7 turns are left!!!!")
50         print("-----")
51         print("      o      ")
52         print("      |      ")
53
54     if turns==6:
55         print("6 turns are left!!!!")
56         print("-----")
57         print("      o      ")
58         print("      |      ")
59         print("      /      ")
60
61     if turns==5:
62         print("5 turns are left!!!!")
63         print("-----")
64         print("      o      ")
65         print("      |      ")
66         print("      / \     ")
67
68     if turns==4:
69         print("4 turns are left!!!!")
70         print("-----")
71         print("      \ o     ")
72         print("      |      ")
```

Restricted Mode 0 0 0

30°C Clear

Search

CHAPTER 9

PROJECT LIMITATION & FUTURE GOAL

Future Scopes and Limitation of this project:-

This game can have varied applications in the context of *word formations* and **puzzles**. Its knowledge can be valuable to many other games like **CROSSWORD PUZZLES, WHEEL OF FORTUNE, SCRABBLE**. We can also have an **investigation of very popular and commonly used letters** in most of the words. Make a frequency distribution in graph out of it. The underlying mathematical concepts are **Data Collection and Analysis, Presentation and Interpretation** which can have lot of implications in language processing and study of graphs and testing conjectures. Also, we can find out that the most popular letter in the English language is **"e"**. The letter frequency of all letters in the English language is: **e t a o i n s r h l d c u m f p g w y b v k x j q z**.

www.google.com

<https://www.gamestolearnenglish.com/hangman/>

<https://linuxize.org/hangman-game-project-report>

[https://en.wikipedia.org/wiki/Hangman_\(game\)](https://en.wikipedia.org/wiki/Hangman_(game))

<https://www.youtube.com/watch?v=Yt4l98jFlu>

THANK YOU