A
PROJECT REPORT
ON

Hangman Game

Towards partial fulfillment of the requirement in

2nd Semester BCA 2022-23

Submitted By:-

2205101100152 Dipesh Rauniyar
2205101100103 Raghav Mahajan
2205101100104 Rahul Kumar

Submitted To:-

Parul Institute of Computer Application,
Parul University.

Under the Guidance of
Prof. Hardik Parmar

Acknowledgement

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We respect and thank Dr. Priya Swaminarayan, Dean, FITCS for providing us an opportunity to do the project work in BCA and giving us all support and guidance which made us complete the project duly. We are extremely thankful to Mam for providing her support and guidance, although she had a busy schedule managing the academic affairs.

We would not forget to remember Prof. Hina Chokshi, HOD, BCA department for her encouragement and more over for her timely support and guidance till the completion of our project work.

We owe our deep gratitude to our project guide Prof. Hardik Parmar , who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

We're thankful to and fortunate enough to get constant encouragement, support and guidance from our Parents, all Teaching staff of the BCA Department which helped us in successfully completing our project work. Also, we would like to extend our sincere esteems to all staff in the laboratory for their timely support.

Dipesh Rauniyar 2205101100152

Raghav Mahajan 2205101100103

Rahul Kumar 2205101100104

PARUL INSTITUTE OF COMPUTER APPLICATION

CERTIFICATE

This is to certify that ***Dipesh Rauniyar, Raghav Mahajan and Rahul Kumar*** the student(s) of Parul Institute of Computer Application, has/have satisfactorily completed the project entitled ***" Hangman Game "*** as a part of course curriculum in BCA semester-II for the academic year 2022-2023 under guidance of ***Prof.Hardik Parmar*.**

Dipesh Rauniyar: 2205101100152

Raghav Mahajan: 2205101100103

Rahul Singh: 2205101100104

Quality of work	Grade	Sign of Internal guide
Poor / Average / Good / Excellent	**B /B+ / A / A+**	

Date of submission:

HOD, Principal,

Prof. Hina Chokshi Dr.Priya Swaminarayan

INDEX

ABSTRACT

Hangman is a classic word-guessing game where the player has to guess a hidden word by guessing the individual letters that make up the word. The player is given a limited number of guesses and must guess the word before they run out of guesses or else they lose the game. The game also includes a hangman figure which is slowly built as the player fails to guess correctly.The Hangman game project is an engaging and interactive game that is designed to help users improve their language skills, vocabulary, and spelling. The game is built using the Python programming language and the Pygame, Tkinter, Pillow and Sqlite3 library, which provide a range of features and tools to create an immersive and visually appealing game experience. In addition to its entertainment value, the Hangman game project can also be used as an educational tool for teachers and educators. However, the proposed system does have some limitations, such as a limited word library and potential compatibility issues with older devices or operating systems. Nonetheless, the game's user-friendly interface, and appealing graphics and sound effects make it a valuable and entertaining addition to the gaming community.

Chapter 1

Introduction to Project System

Hangman is a popular word guessing game that is often used in educational settings to help students learn new words and vocabulary. The game is simple: players have to guess the word by guessing one letter at a time. If the players guess wrong, the game adds an element of suspense as a "hangman" is slowly drawn and one more wrong guess will result in the player losing the game. Hangman is a great game for learning new words, helping with spelling, and having some fun with friends and family. In this system we have used various python modules for building this game some of the modules are tkinter, pillow, pygame, sqlite3, threading and random. In this system the user primary have to register himself and only after that the user is allowed to login and start playing game. All the data provided by user is stored through sqlite3 and is stored in a local file. To play the user have to user have to click the character displayed in the game and guess the letter and according to it the system will give score. In this system we have also provided negative scoring for wrong guess made. In this system we have also implemented sound for making the user more enjoyable to play the game again and again. The pillow library is used for adding background image to the start page of the game. The words that are used for guessing are stored in a text file and are later used in the game we can later update the words by simply copying the words to the file. The system has been integrated of two windows tkinter and pygame. Through use of tkinter we created login page , registration page and start page. Where as in pygame we simply created the game page only. We have provided different frame sizes to different pages according to requirement. We have provided 500*550 to login and registration page were as for start page we implemented 605*305 frame sizes for providing better feel to the users. .

Chapter 2

System Requirement Specification

2.1 Introduction to SRS

2.1.1 What is SRS?

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

2.1.2 Need of SRS

In order to fully understand one's project, it is very important that they come up with a SRS listing out their requirements, how are they going to meet it and how will they complete the project. It helps the team to save upon their time as they are able to comprehend how are going to go about the project. Doing this also enables the team to find out about the limitations and risks early on.

2.2 Hardware Requirement

Hardware Components	Specification
Processor	Intel core I3,/I5
RAM	4GB/8GB
Hard disk	512GB/1TB
Monitor	15.6 colour monitor or advance
Device	Keyboard, Mouse

2.3 Software Requirement

Name of component	Specification
Operating System	WindowsXP,windows10
Software development kit	Python Version 3 or above, Python Pycharm
Programming language	Python Programming [with MySql]

2.4 System Users

2.4.1 Player/User

2.5 Description of User Role

2.5.1 Player /User

Player are the only user of this system .They simply register themselves in system and use the entered username and password to start playing the game. To play they use mouse to select the shown option or letters to guess the words.

2.6 System Features

2.6.1 Registration
2.6.2 Login
2.6.3 Background sound
2.6.4 Lives System
2.6.5 Score System
2.6.6 2D animation

2.7 Description of Features

2.7.1 Registration

Data of the player or user is saved from registration page to use it as user authentication while logging to the game.

2.7.2 Login

Only the authenticate user are allowed to login and play the game.

2.7.3 Background Sound

Background sound is implemented in the system for proving the user a thrilling feeling while playing this game and enjoy the game.

2.7.4 Lives System

This system or game has lives system but unlike other games this game lives status is shown in the form of images. There are total 5 lives available in this game.

2.7.5 Score System

Scores are obtained on the basis of right guess and there is also a negative scoring system provided in it.

2.7.6 2D Animation

The game is implemented in a 2d format so that the user won’t be bored as many 1d animation games or console games are very boring for user to play.

2.8 Timeline Chart

Development phase	***75 Days***						**Duration N (days)**
	0to10 days	11to20 days	21to30 days	31to40 days	41to50 days	51to75 days	
Requirement Gathering							07
Analysis							09
Design							10
Development Phase 1							13
Development Phase 2							13
Development Phase 3							13
Documentation							10
Total time (Days)							**75**

Figure: 2.1 Time line chart of Hangman Game

Chapter 3
System Flow Diagram

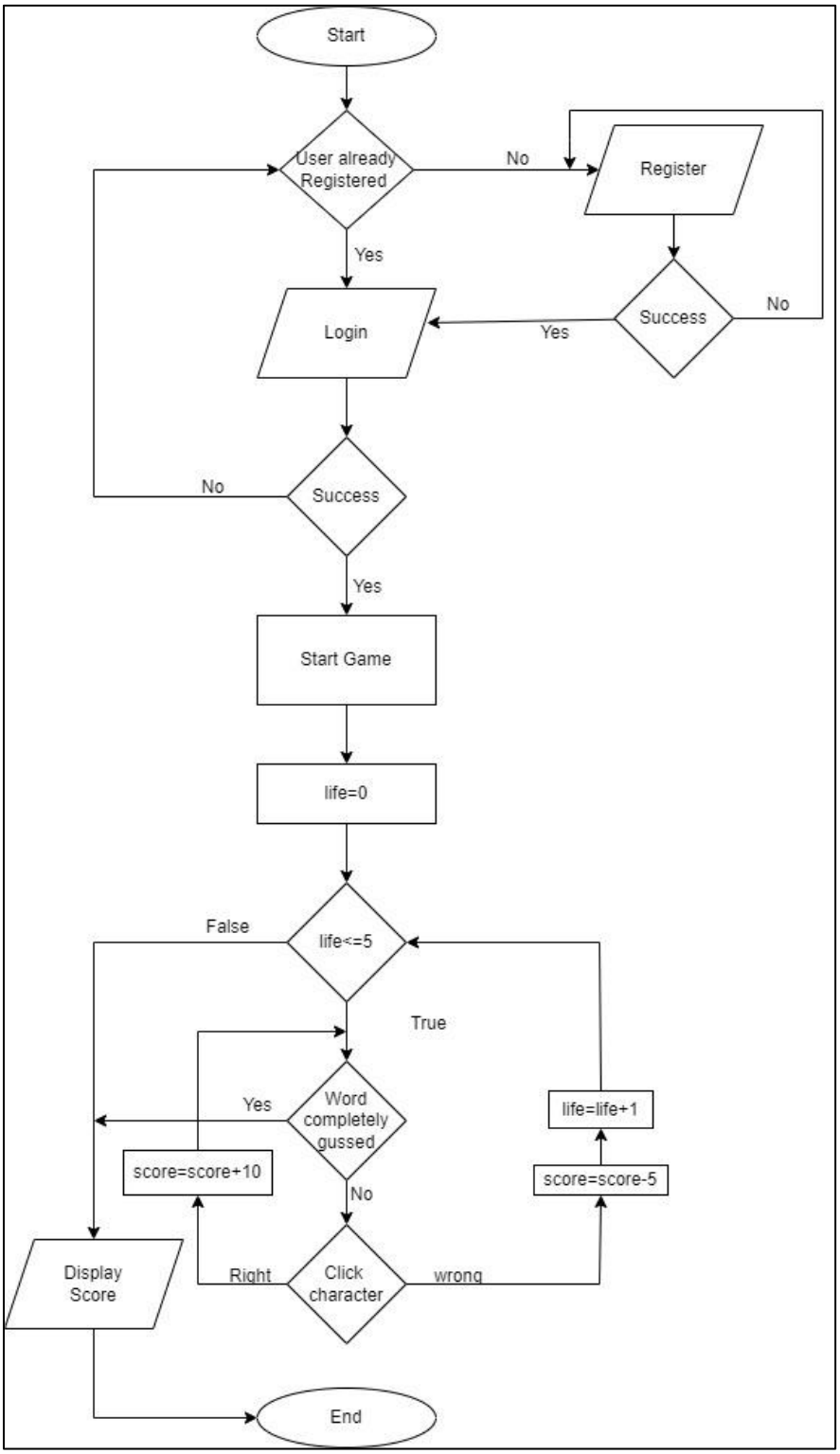

Figure 3.1 System Flow Diagram of Hangman Game

The above figure shows that the applications start and primary it goes for conditional checking It checks it the user is registered or not. If not registered then the user need to register for accessing .If user is registered then they go for login. If the login is successful then it goes to the start game page. If not then they need to check the credentials if wrongly typed. After start page of the game the game runs and the life the player has total would be smaller then and equal to 5. Now according to guessed character the user gain points or losses the pointes. After losing or winning the score is displayed .

Chapter 4

Data Flow Diagram (All Levels)

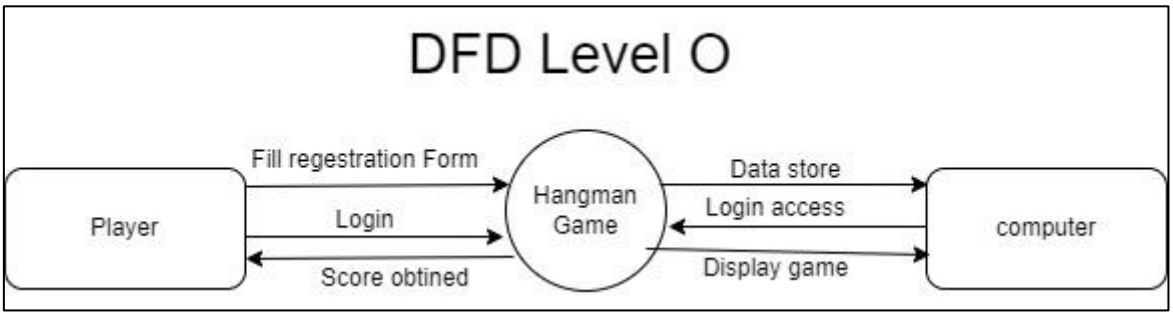

Figure 4.1 Data Flow Diagram level 0 of Hangman Game

The above figure shows the DFD level 0. Where the player can register, login and obtain score from the system hangman game. The data obtained from user or player is stored in the external entity called computer. The login access and display shown is also carried out by it.

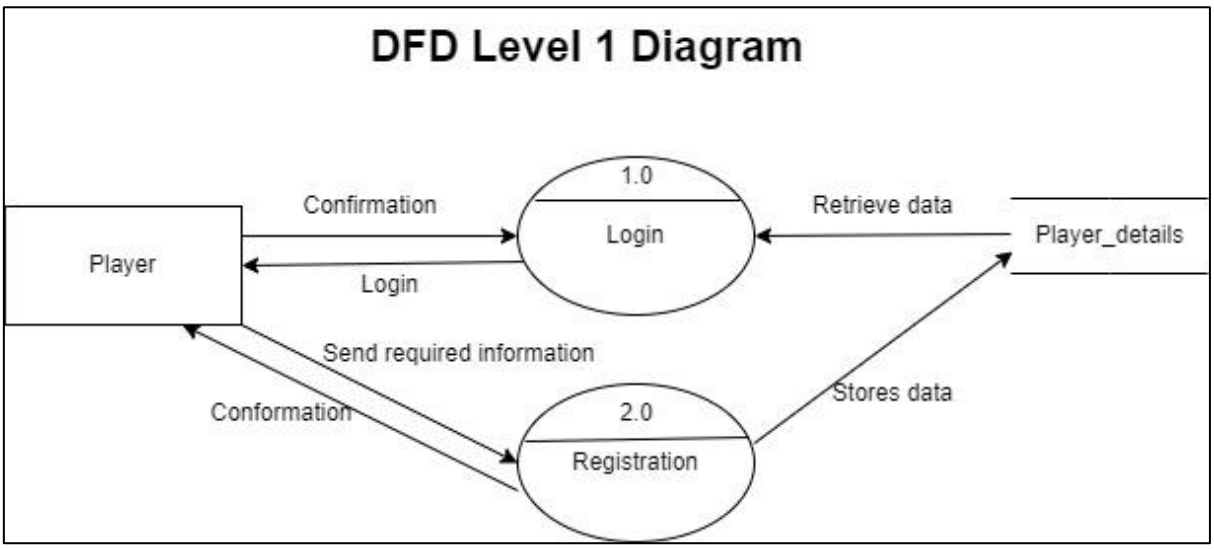

Figure 4.2 Data Flow Diagram level 1 of Hangman Game

The above diagram show the DFD level 1 diagram where the player request for login and the process is carried out by the login process which retrieves the data from database which is stored in Player_details. The player can also register for the system by simply sending required information to the system process and the information is saved in Player_detail database. If every then is correct in this process then the process called registration request for conformation to the player.

DFD Level 2 Diagram

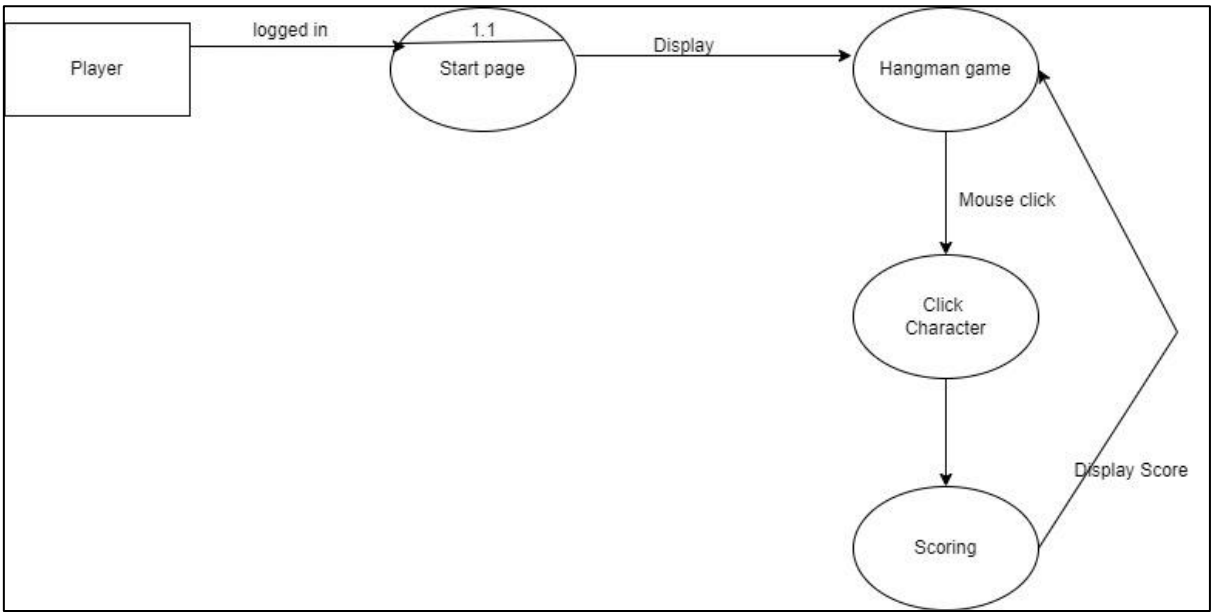

Figure 4.3 Data Flow Diagram level 2 of Hangman Game

The above figure shows DFD level 2 diagram of the hangman game. Here the player is logged in to the system and the hangman game is shown where the player need to click the charater to obtain score the the score is displayed by the hangman game.

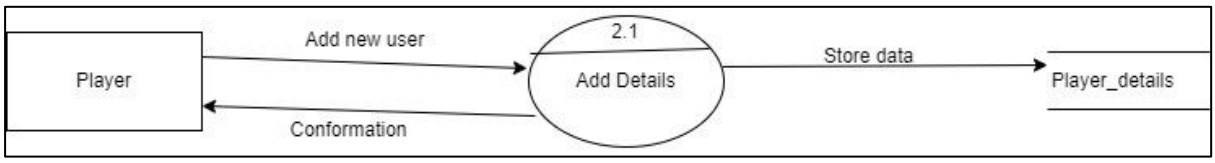

Figure 4.4 Data Flow Diagram level 2 of Hangman Game

The above figure shows us the player entity can add new user and if every information is correct or not . After conformation the data is stored in the database named Player_details.

Chapter 5
Use Case Diagram

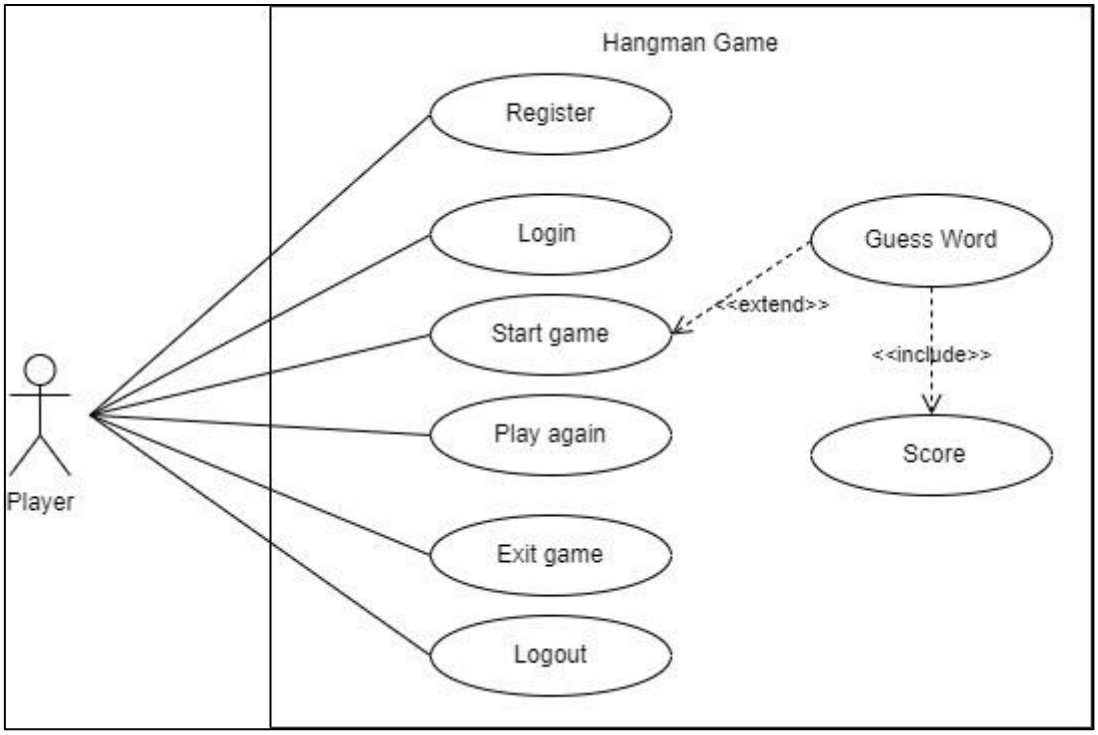

Figure 5.1 Use Case Diagram of Hangman Game

The above figure shows that the player actor can register, login ,start the game, play again, exit the game and logout. And the start game includes guessing of words and the guessed words includes scores.

Chapter 6
Data Dictionary

6.1 Player_details

Stores personal details of the player/user.

Sr.no	Column	Data type	Size	KEY	Constrain	Description	example
1	Player_id	Int	-	PK	Al	Primary key of table	1
2	First_name	Varchar	20		Not Null	First name of user	James
3	Last_name	Varchar	20		Not Null	Last name of user	Bond
4	Contact_no	Varchar	13		Not Null	Contact Number with Country Code	+91 9876543210
5	Email	Varchar	30		Not Null	Email of player	James123@gmail.com
6	User_name	Varchar	20		Not Null	User Name	champion
7	Password	Varchar	10		Not Null	Password	******

Table 6.1 : Player_details

Chapter 7
Screenshot of Development Phase 1

7.1 Login Page Design

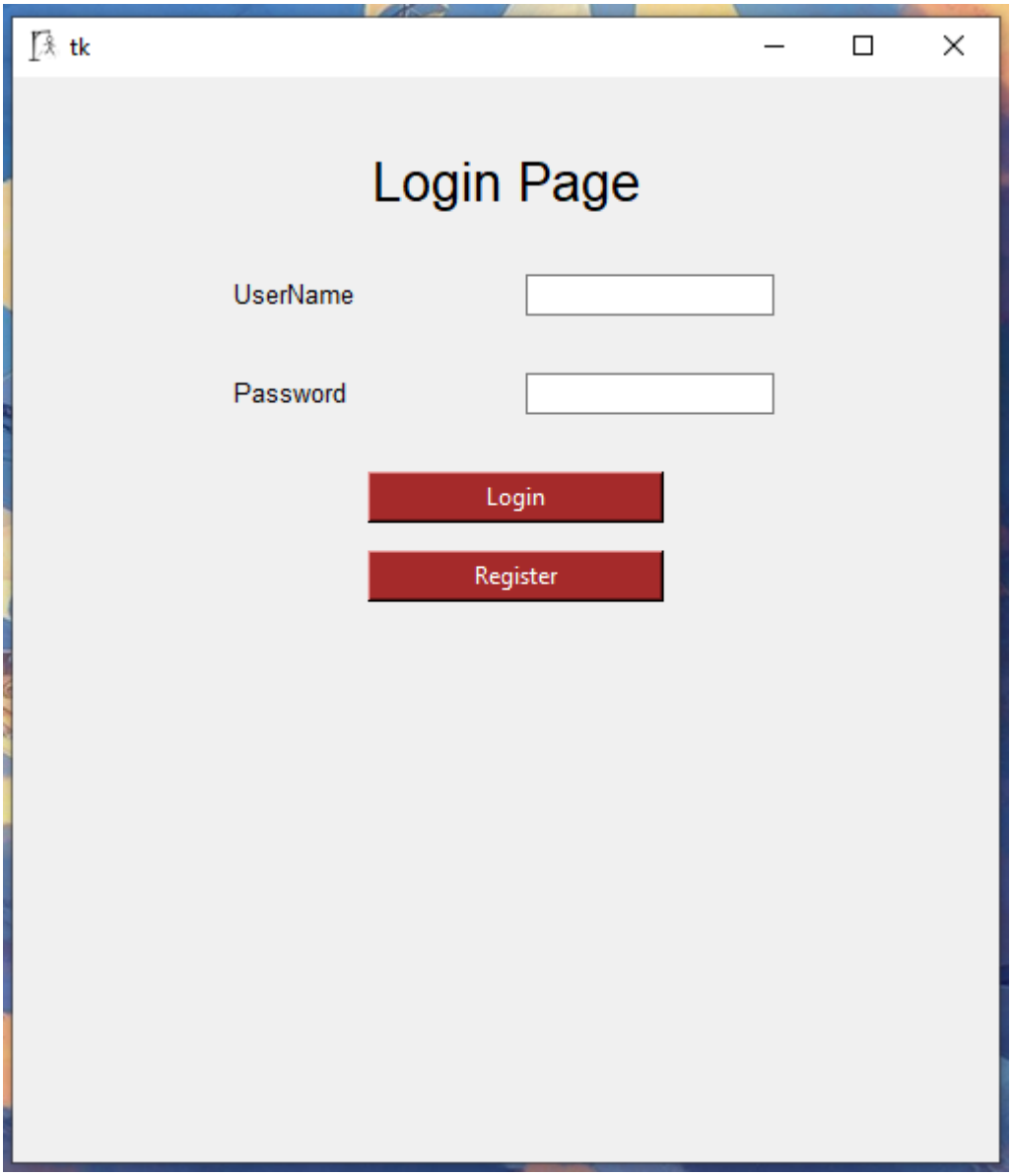

Figure 7.1 Login Page Design

In this Page user can login to their existing account and can also create a new profile for new users.

7.2Code of Login Page

```python
import tkinter as tk
from tkinter import *
from tkinter import ttk
from PIL import Image, ImageTk
import sqlite3
import pygame
import random
import threading
from pygame import mixer

class tkinterApp(tk.Tk):

    # __init__ function for class tkinterApp
    def __init__(self, *args, **kwargs):
        # __init__ function for class Tk
        tk.Tk.__init__(self, *args, **kwargs)

        # creating a container
        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        # initializing frames to an empty array
        self.frames = {}
```

Figure 7.2.1 Code of Login Page

In this section we imported various modules / libraries for different purpose. The module tkinter are used for GUI (Graphical User Interface), were as PIL /Pillow library is used for inserting images into the frames of software, Sqlite3 is used for storing the information obtained from players/users to further use it for authentication process. Pygame modules are simply used to create 2d games .random module is used to choose random items. Threading is used for creating some time based intervals takes to be done. To Switch from different frames in tkinter we created a class called tkinterApp and as argument we provided tk.Tk which is used to create custom commands to create and manipulate GUL widgets. After that in tkinterApp class we created an __inti__ function which is used to take values for the user and store it in identifiers. In it we also created containers and provided different parameters for the container. The frames are we created a empty array so that we can iterate the frames one by one.

```
26
27          # iterating through a tuple consisting
28          # of the different page layouts
29          for F in (Login_Page, Register):
30              frame = F(container, self)
31
32              # initializing frame of that object from
33              # login page, regestration page, game page respectively with
34              # for loop
35              self.frames[F] = frame
36
37              frame.grid(row=0, column=0, sticky="nsew")
38
39          self.show_frame(Login_Page)
40
41      # to display the current frame passed as
42      # parameter
43      def show_frame(self, cont):
44          frame = self.frames[cont]
45          frame.tkraise()
46
```

Figure 7.2.2 Code of Login Page

The frames are iterated using for loop and another function called show_frame was created so that the frames would be raised / displayed according to given commands. The code in line 39 shows that which page to show while first opening the software which is login page.

```
48  class Login_Page(tk.Frame):
49      def __init__(self, parent, controller):
50          tk.Frame.__init__(self, parent)
51
52          def start_game():...
257
258         # for checking username and password
259         username = StringVar()
260         password = StringVar()
261
262         def check_login():
263             # to show error while logging and disappers after some time
264             def error_message():
265                 label_4 = Label(app, text="User doesnot exists", width=20, font=("bold", 10))
266                 label_4.place(x=180, y=280)
267                 start_time = threading.Timer(6, lambda: label_4.place_forget())
268                 start_time.start()
269
270             global user
271             user = username.get()
272             pas = password.get()
273             conn = sqlite3.connect('Regestration.db')
274             c = conn.cursor()
275             c.execute("SELECT * FROM Player_details")
```

Figure 7.2.3 Code of Login Page

For Login Page we create a class called Login_Page for frame. In it we created a function to check the logging authentication so for that we first connected to the existing database called Registration.db and compare it with the existing data from the database for username and password. In the figure we also create function for error message to show error if the user entered wrong credentials and the error message is removed after 6 sec of showing which was implemented with the help of threading library.

```
records = c.fetchall()
d = 0
for i in records:
    if i[4] == user and i[5] == pas:
        d = 1

if d == 1:
    # Start game page
    root = Tk
    img = ImageTk.PhotoImage(Image.open("new.jpg"))
    app.geometry("605x305")
    label = Label(self, image=img)
    label.place(x=0, y=0)
    Button(self, text='Start', width=20, bg='white', fg='black',
           command=start_game).place(x=230, y=250)
    root.mainloop(self)
else:
    error_message()

conn.commit()

conn.close()
```

Figure 7.2.4 Code of Login Page

If the user has entered correct credentials then the program will go to start page.

```python
        # layout of login page
        label_0 = ttk.Label(self, text="Login Page", width=20, font=("bold", 20))
        label_0.place(x=180, y=35)

        label_2 = ttk.Label(self, text="UserName", width=20, font=("bold", 10))
        label_2.place(x=110, y=100)

        entry_2 = ttk.Entry(self, textvar=username)
        entry_2.place(x=260, y=100)

        label_3 = ttk.Label(self, text="Password", width=20, font=("bold", 10))
        label_3.place(x=110, y=150)

        entry_3 = ttk.Entry(self, textvar=password)
        entry_3.place(x=260, y=150)

        # button to login
        Button(self, text='Login', width=20, bg='brown', fg='white', command=check_login).place(x=180, y=200)

        ## button to register
        Button(self, text='Register', width=20, bg='brown', fg='white',
               command=lambda: controller.show_frame(Register)).place(x=180, y=240)
```

Figure 7.2.5 Code of Login Page

This image shows the codes used for creating logging page layout.

```python


app = tkinterApp()
app.iconbitmap('hangman6.ico')

app.geometry("500x550")

app.mainloop()
```

Figure 7.2.6 Code of Login Page

This is the last part for creating a frame and display it simply show that we have implemented icon to the system and it also includes the frame size .

Chapter 8
Screenshot of Development Phase 2

8.1 Registration Page Design

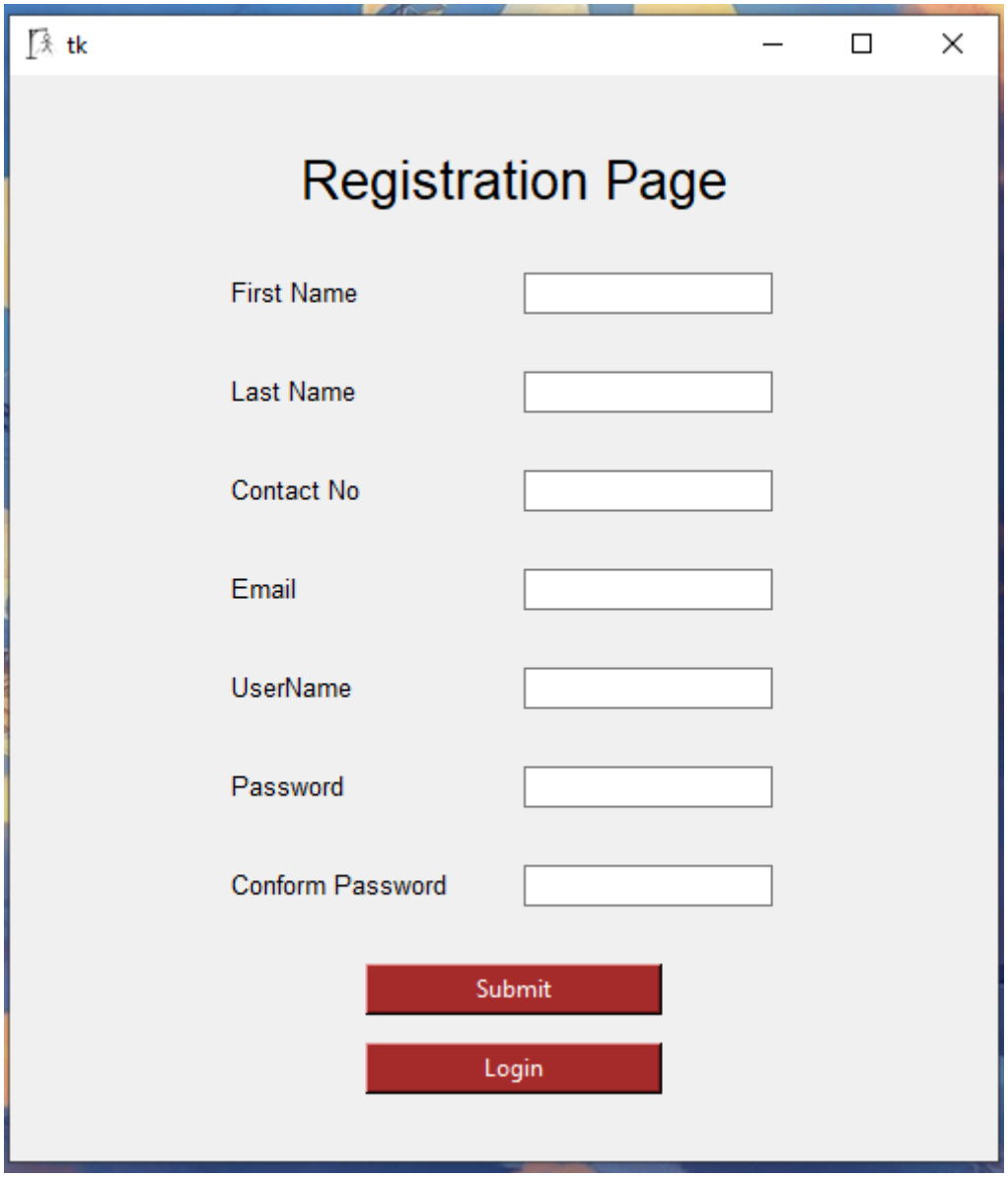

Figure 8.1.1 Registration Page Design

In this page user can register a new profile and can also go to the login page.

8.2Code of Registration Page

```python
# Second window frame Regestration page
class Register(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        # data storage type
        player_id = int()
        first_name = StringVar()
        last_name = StringVar()
        contact_no = StringVar()
        username = StringVar()
        password = StringVar()
        conform_password = StringVar()
        email = StringVar()

        # data storing
        def database():
            # inputs extraction
            f_name = first_name.get()
            l_name = last_name.get()
            contact = contact_no.get()
            user = username.get()
            pas = password.get()
            con_pas = conform_password.get()
            em = email.get()
```

```python
        conn = sqlite3.connect('Regestration.db')
        with conn:
            c = conn.cursor()

        c.execute(
            'CREATE TABLE IF NOT EXISTS Player_details (Player_id INTEGER PRIMARY KEY AUTOINCREMENT,'
            'First_Name TEXT(20),Last_Name TEXT(20),Contact_No INT(13),Username TEXT(20),Password TEXT(10),'
            'Email TEXT(30))')
        c.execute("SELECT * FROM Player_details")
        records = c.fetchall()

        def error_message_regpage():
            d = 0

            # Serching data from sql database and logic
            for i in records:
                if i[4] == user or i[6] == em:
                    d = 1
                    if i[4] == user and i[6] == em:

                        label_9 = Label(self, text="Username and Email Already exists", width=20, font=("bold", 10))
                        label_9.place(x=180, y=520)
                    elif i[4] == user:
                        label_9 = Label(self, text=" Username Already exists", width=20, font=("bold", 10))
                        label_9.place(x=180, y=520)
```

Figure 8.2.1 Code of Registration Page

The above images show that for creating a registration page we created a function called register where we first start with creation of database if doesn't exist. After that we declared the variable data type which will be further be used for storing it in SQL database. To store the data in SQL we created another function called database where we declared another variables that will be used to store the information obtained from entry widget. In this function we connected our database called registration.db and started to create table called Player_details and required table headers if not previously created along with its data type and constrains. If the database and table is already created then if fetches all data for further use. Then we created another function called error_message_repage which simply returns error messages case the user didn't fill a box or the user and email already exist.

```
                elif i[6] == em:
                    label_9 = Label(self, text="Email  Already exists", width=20, font=("bold", 10))
                    label_9.place(x=180, y=520)

            if d != 1:
                if len(user) == 0 or len(em) == 0 or len(pas) == 0 or len(con_pas) == 0 or len(f_name) == 0 or len(
                        l_name) == 0 or len(contact) == 0:
                    label_9 = Label(self, text="Box can,t be empty", width=20, font=("bold", 10))
                    label_9.place(x=180, y=520)
                elif pas == con_pas:
                    c.execute(
                        'INSERT INTO Player_details (First_name,Last_Name,Contact_No,Email,'
                        'Username,Password) VALUES(?,?,?,?,?,?)',
                        (f_name, l_name, contact, em, user, pas))
                    label_9 = Label(self, text="Success", width=20, font=("bold", 10))
                    label_9.place(x=180, y=520)


                elif pas != con_pas:
                    label_9 = Label(self, text="Incorrect Password", width=20, font=("bold", 10))
                    label_9.place(x=180, y=520)

            start_time = threading.Timer(6, lambda: label_9.place_forget())
            start_time.start()
```

Figure 8.2.2 Code of Registration Page

If all fields are correctly typed then the data is now stored in table Player_details and shows message success. The error or success messages are disappears after some interval.

```python
        message_store = error_message_regpage()

        for i in records:
            print(i)

        conn.commit()
        conn.close()

    # Regester Page layout
    label_0 = ttk.Label(self, text="Registration Page", width=20, font=("bold", 20))
    label_0.place(x=145, y=35)

    label_1 = ttk.Label(self, text="First Name", width=20, font=("bold", 10))
    label_1.place(x=110, y=100)
    entry_1 = ttk.Entry(self, textvar=first_name)
    entry_1.place(x=260, y=100)

    label_2 = ttk.Label(self, text="Last Name", width=20, font=("bold", 10))
    label_2.place(x=110, y=150)
    entry_2 = ttk.Entry(self, textvar=last_name)
    entry_2.place(x=260, y=150)

    label_3 = ttk.Label(self, text="Contact No", width=20, font=("bold", 10))
    label_3.place(x=110, y=200)
    entry_3 = ttk.Entry(self, textvar=contact_no)
```

```python
    entry_3.place(x=260, y=200)

    label_4 = ttk.Label(self, text="Email", width=20, font=("bold", 10))
    label_4.place(x=110, y=250)
    entry_4 = ttk.Entry(self, textvar=email)
    entry_4.place(x=260, y=250)

    label_5 = ttk.Label(self, text="UserName", width=20, font=("bold", 10))
    label_5.place(x=110, y=300)
    entry_5 = ttk.Entry(self, textvar=username)
    entry_5.place(x=260, y=300)

    label_6 = ttk.Label(self, text="Password", width=20, font=("bold", 10))
    label_6.place(x=110, y=350)
    entry_6 = ttk.Entry(self, textvar=password)
    entry_6.place(x=260, y=350)

    label_7 = ttk.Label(self, text="Conform Password", width=20, font=("bold", 10))
    label_7.place(x=110, y=400)
    entry_7 = ttk.Entry(self, textvar=conform_password)
    entry_7.place(x=260, y=400)

    # button for submit in reg page
    Button(self, text='Submit', width=20, bg='brown', fg='white', command=database).place(x=180, y=450)
```

```python

    # button for login in reg page
    Button(self, text='Login', width=20, bg='brown', fg='white',
           command=lambda: controller.show_frame(Login_Page)).place(x=180, y=490)
```

Figure 8.2.3 Code of Registration Page

The database if finally closed at last and after that all the codes are about the layout of registration page.

Chapter 9
Screenshot of Development Phase 3

9.1 Game Design

Figure 9.1.1 Game Design

In this page, user can start a new game by clicking on start button.

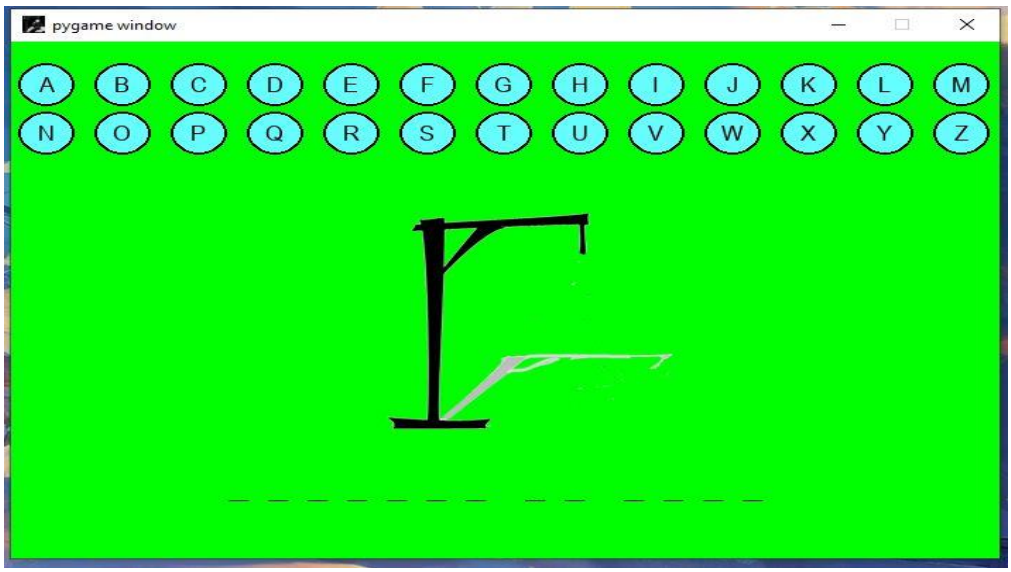

Figure 9.1.2 Game Design

This is the main game screen, here user can click on alphabets displayed on the screen, and start playing the game.

9.2Code of Game

```
def start_game():
    # creating game

    global user
    pygame.init()

    # game background sound
    mixer.init()
    mixer.music.load('sound.mp3')
    mixer.music.set_volume(5)
    mixer.music.play(loops=-1)

    winHeight = 480
    winWidth = 700
    app = pygame.display.set_mode((winWidth, winHeight))
    img = pygame.image.load('hangman6.ico')
    pygame.display.set_icon(img)
    # ---------------------------------------#
    # initialize global variables/constants #
    # ---------------------------------------#
    BLACK = (0, 0, 0)
    WHITE = (255, 255, 255)
    RED = (255, 0, 0)
    GREEN = (0, 255, 0)
    BLUE = (0, 0, 255)
    LIGHT_BLUE = (102, 255, 255)
```

Figure 9.2.1 Code of Game

We have created a function called called start_game so that the game can be called again and again if required. For game windows we have used init function of pygame which simply creates a window for game after that we have implemented background sound for that we used the mixer.music module. And for continuous playing of music we just loop if giving value -1 which does the same work. We then provided the windows display size and icon for the game. After that we provided different color variable to be used in the game.

```python

        btn_font = pygame.font.SysFont("arial", 20)
        guess_font = pygame.font.SysFont("monospace", 24)
        lost_font = pygame.font.SysFont('arial', 45)
        global word
        global buttons
        word = ''
        buttons = []
        global guessed
        global guessed
        global hangmanPics
        global limbs
        guessed = []
        hangmanPics = [pygame.image.load('hangman0.png'), pygame.image.load('hangman1.png'),
                       pygame.image.load('hangman2.png'), pygame.image.load('hangman3.png'),
                       pygame.image.load('hangman4.png'), pygame.image.load('hangman5.png'),
                       pygame.image.load('hangman6.png')]

        limbs = 0

        def redraw_game_window():
            global guessed
            global hangmanPics
            global limbs
            app.fill(GREEN)
```

Figure 9.2.2 Code of Game

After that we gave the font type to the button and font going to be displayed in the game windows. We stored all the hangman image in an array for iterating it according to requirement in logic section. Then we created a function called redraw_game_window so that the window of the game can be updated time to time and set the background color green.

```python
            # Buttons
            for i in range(len(buttons)):
                if buttons[i][4]:
                    pygame.draw.circle(app, BLACK, (buttons[i][1], buttons[i][2]), buttons[i][3])
                    pygame.draw.circle(app, buttons[i][0], (buttons[i][1], buttons[i][2]), buttons[i][3] - 2
                                       )
                    label = btn_font.render(chr(buttons[i][5]), 1, BLACK)
                    app.blit(label,
                             (buttons[i][1] - (label.get_width() / 2), buttons[i][2] - (label.get_height() / 2)))

            spaced = spacedOut(word, guessed)
            label1 = guess_font.render(spaced, 1, BLACK)
            rect = label1.get_rect()
            length = rect[2]

            app.blit(label1, (winWidth / 2 - length / 2, 400))

            pic = hangmanPics[limbs]
            app.blit(pic, (winWidth / 2 - pic.get_width() / 2 + 20, 150))
            pygame.display.update()

        def randomWord():
            file = open('words.txt')
            f = file.readlines()
            i = random.randrange(0, len(f) - 1)
```

Figure 9.2.3 Code of Game

After that we started to create the button for the game for that we used for loop for creation of the button. Bellow the button we created a area for the hangman image to be shown. Then for words guessing we created another function called randomWord which consist of file handling operation where we created a dot txt file called word.txt and store the words we need to be used in the game.

```
        return f[i][:-1]

    def hang(guess):
        global word
        if guess.lower() not in word.lower():
            return True
        else:
            return False

    def spacedOut(word, guessed=[]):
        spacedWord = ''
        guessedLetters = guessed
        for x in range(len(word)):
            if word[x] != ' ':
                spacedWord += '_ '
                for i in range(len(guessedLetters)):
                    if word[x].upper() == guessedLetters[i]:
                        spacedWord = spacedWord[:-2]
                        spacedWord += word[x].upper() + ' '
            elif word[x] == ' ':
                spacedWord += ' '
        return spacedWord

    def buttonHit(x, y):
```

Figure 9.2.4 Code of Game

We created a hang function where we gave an argument called guess where we check if the guessed word matches with the work clicked and returns true if right guess. The another function called spacedOut is created for storing the guessed words for that we gave empty array as argument.

```
def buttonHit(x, y):
    for i in range(len(buttons)):
        if x < buttons[i][1] + 20 and x > buttons[i][1] - 20:
            if y < buttons[i][2] + 20 and y > buttons[i][2] - 20:
                return buttons[i][5]
    return None

def end(winner=False):
    global limbs

    lostTxt = 'You Lost, press any key to play again...'
    winTxt = 'WINNER!, press any key to play again...'
    redraw_game_window()
    pygame.time.delay(1000)
    app.fill(GREEN)

    if winner == True:
        label = lost_font.render(winTxt, 1, BLACK)
    else:
        label = lost_font.render(lostTxt, 1, BLACK)

    wordTxt = lost_font.render(word.upper(), 1, BLACK)
    wordWas = lost_font.render('The phrase was: ', 1, BLACK)

    # show score obtained
    scoreis = lost_font.render(user+' Score is :', 1, BLACK)
    point = lost_font.render(str(score), 1, BLACK)

    app.blit(wordTxt, (winWidth / 2 - wordTxt.get_width() / 2, 295))
    app.blit(wordWas, (winWidth / 2 - wordWas.get_width() / 2, 245))
    app.blit(label, (winWidth / 2 - label.get_width() / 2, 140))
    app.blit(scoreis, (winWidth / 2 - scoreis.get_width() / 2, 40))
    app.blit(point, (winWidth / 2 - point.get_width() / 2, 85))
    pygame.display.update()
    again = True
    while again:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
            if event.type == pygame.KEYDOWN:
                again = False
    reset()

def reset():
    global limbs
    global guessed
    global buttons
    global word
    for i in range(len(buttons)):
        buttons[i][4] = True
```

Figure 9.2.5 Code of Game

Next function is buttonHit which simply tracks which buttons are clicked and return that button. The function end simply is for showing the Winning or loosing text with the scores obtained along with username used for logging in and to play gain the user need to press any key. The reset function simply empties the array or makes the required variables null.

```
            limbs = 0
            guessed = []
            word = randomWord()

        # MAINLINE

        # Setup buttons
        increase = round(winWidth / 13)
        for i in range(26):
            if i < 13:
                y = 40
                x = 25 + (increase * i)
            else:
                x = 25 + (increase * (i - 13))
                y = 85
            buttons.append([LIGHT_BLUE, x, y, 20, True, 65 + i])
            # buttons.append([color, x_pos, y_pos, radius, visible, char])

        word = randomWord()
        inPlay = True
        score = 0
```

Figure 9.2.6 Code of Game

Not only the variable and array are made null but also resets the buttons that were clicked for guess and after all required objects are reset the game can be played again.

```
while inPlay:
    redraw_game_window()
    pygame.time.delay(10)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            inPlay = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                inPlay = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            clickPos = pygame.mouse.get_pos()
            letter = buttonHit(clickPos[0], clickPos[1])
            if letter != None:
                guessed.append(chr(letter))
                buttons[letter - 65][4] = False
                if hang(chr(letter)):
                    if limbs != 5:
                        score = score - 5
                        limbs += 1
                        print(score)
                    else:
                        end()
                else:
                    print(spacedOut(word, guessed))
                    score = score + 10
```

```
                score = score + 10
                print(score)
                if spacedOut(word, guessed).count('_') == 0:
                    end(True)
mixer.music.stop()
pygame.quit()
```

Figure 9.2.7 Code of Game

The while loop is implemented until the game is going on all game events are checked one by one using for loop and at last background music is stopped and game quits.

Chapter 10
Conclusion

The Hangman game project is an engaging and interactive game that is designed to help users improve their language skills, vocabulary, and spelling. The game is built using the Python programming language and the Pygame, Tkinter, Pillow and Sqlite3 library, which provide a range of features and tools to create an immersive and visually appealing game experience. In addition to its entertainment value, the Hangman game project can also be used as an educational tool for teachers and educators. However, the proposed system does have some limitations, such as a limited word library and potential compatibility issues with older devices or operating systems. Nonetheless, the game's user-friendly interface, and appealing graphics and sound effects make it a valuable and entertaining addition to the gaming community, where as for the implementation of codes and development process there are many areas that this system needs to improve in to be called a quality product.

Chapter 11
Future Enhancement

1. Multiplayer Feature
2. Changing 2d graphics to 3d
3. Changing it to an online game
4. Leadership Board for global competition

Chapter 12

References

Website:

1. http://https//www.w3schools.com/python/
2. https://github.com/rmahajan466/PU_Project-01_Sem02
3. https://docs.python.org/3/library/sqlite3.html
4. https://www.geeksforgeeks.org/python-sqlite/

Book:

1. "Python GUI Programming with Tkinter", Alan D.Moore
2. "Morden Tkinter For Busy Python Developers 3rd Ed", Mark Roseman