

CATCHER:THE **INTERACTIVE GAME**

PROJECT REPORT

COMPUTER SCIENCE

MADE BY-ARIB ANSARI

BOARD SCHOOL NO.-

SCHOOL-DPS INDIRAPURAM

SESSION-2021-2022

Certificate

This is to certify that ARIB ANSARI of class XII-F has prepared the project on “Catcher: The Interactive Game”. The project is result of his efforts endeavours. This project is found worthy of acceptance as the final project report for the subject computer science of class XII.

He has prepared this project under my guidance.

Ms. Rinkoo Gupta
(Computer Science Teacher)
(DPS Indirapuram)

Acknowledgement

I would like to express a deep sense of gratitude towards my computer science teacher Rinkoo Gupta for always guiding me.

My sincere thanks goes to Ms. Sangeeta Hajela, our Principal.

I would like to thank all those who have helped directly or indirectly in the completion of the project.

Arib Ansari

XII-F

INDEX

S.No.	Topic	Remarks
1	Introduction	
2	Tools used	
3	Hardware and Software requirements	
4	MySQL tables used and their structure	
5	Coding	
6	Outputs	
7	Conclusion	
8	Future Enhancements	
9	Bibliography	

Introduction

The project's aim is to build a player controlled 2-D defence game called 'CATCHER : THE INTERACTIVE GAME'.

About the game:

On starting the program we go to a user-menu wherein we can take the role of a 'player' or an 'admin'.

In this menu, the data is stored into tables in MySQL using Python's MySQL connector. This data consists of names and scores of users. This is the part where everything related to the school's curriculum is used.

After choosing play, we can begin the game. A window pops up and we can move the character towards the newly spawned enemies who move towards the player in hopes of getting past and break through.

After the game ends the player is shown a window where he/she/them can input their name and their score to be entered into the leader boards.

On the main menu players can access the leader boards and remove their records or update it as they see fit.

The game can be exited either by selecting quit on the main menu or by pressing x(close) on the game window.

WORKING DESCRIPTION:

The build up of the user menu section is as follows :

1.Start: Executes Game

2.Quit

3.Admin

3.1 Add record

3.2 Remove all Records

All the points stated above have been made to incorporate the basic functions of SQL.

The program consists of 3 separate files of code.
These are:

- 1.CreateDb- makes the database in SQL through python.
- 2.CreateTable-makes the table in the database in SQL.
- 3.Main-The code that executes the menu and game.

Tools Used

Python:

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many Unix variants including Linux and macOS, and on Windows.

MYSQL

MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL).

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or a place to hold the vast amounts of information in a corporate network. In particular, a relational database is a digital store collecting data and organizing it according to the relational model. In this model, tables consist of rows and columns, and relationships between data elements all follow a strict logical structure. An RDBMS is simply the set of software tools used to actually implement, manage, and query such a database.

MySQL is integral to many of the most popular software stacks for building and maintaining everything from customer-facing web applications to powerful, data-driven B2B services. Its open-source nature, stability, and rich feature set, paired with ongoing development and support from Oracle, have meant that internet-critical organizations such as Facebook, Flickr, Twitter, Wikipedia, and YouTube all employ MySQL backends.

PYGAME:

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. Pygame is suitable to create client-side applications that can be potentially wrapped in a standalone executable.

Tkinter:

Tkinter is an open source, portable graphical user interface (GUI) library designed for use in Python scripts.

Tkinter relies on the Tk library, the GUI library used by Tcl/Tk and Perl, which is in turn implemented in C. Therefore, Tkinter can be said to be implemented using multiple layers.

Hardware and Software Requirements

Hardware: Ryzen 5 4600H

Software: Windows* 10

Install each compulsorily:

1.Python

2.MySQL 8.0 command line

3.MySQL Connector/Python 8

4.Python/Pygame

SQL TABLES USED AND THEIR STRUCTURE

Used databases:

```
mysql> show databases;
+-----+
| Database |
+-----+
| a        |
| b        |
| d        |
| information_schema |
| mysql    |
| performance_schema |
| recordb  |
| sys      |
+-----+
8 rows in set (1.67 sec)
```

Admin table structure:

```
mysql> use recordb;
Database changed
mysql> desc record;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name  | varchar(200)  | YES  |     | NULL    |       |
| score | varchar(200)  | YES  |     | NULL    |       |
| id    | int           | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.70 sec)
```

CODING

#create databases(db):

```
import mysql.connector as con
mycon = con.connect(host='localhost',user='root',passwd='hello')
cur = mycon.cursor()
cur.execute('SHOW DATABASES')
flag = False
for dbname in cur:
    if('recordb' in dbname):
        flag= True
        break
if not flag:
    cur.execute("Create Database recordb")
    print('Database created')
else:
    print('Database Already Exists')
mycon.commit()
```

#create table:

```
import mysql.connector as con

mycon =
con.connect(host='localhost',user='root',passwd='hello',database='recordb')

cur = mycon.cursor()

stry = 'CREATE TABLE record(name varchar(200), score varchar(200) , id int)'

cur.execute(stry)

mycon.commit()
```

#Main File:

```
import pygame

from pygame.locals import *

import os

from pygame import mixer

from tkinter import *

import random

# Game Initialization

pygame.init()

from tkinter import *

from tkinter import ttk

import mysql.connector as mycon

con=mycon.connect(host='localhost',user='root',passwd='hello',database='r
ecordb')
```

```
cur = con.cursor()
```

```
def popup():
```

```
    root1 = Tk()
```

```
    root1.title("idk2")
```

```
    root1.geometry("500x600")
```

```
    count1=0
```

```
def add_record1():
```

```
    A=[]
```

```
    a=name_box1.get()
```

```
    b=score1.get()
```

```
    c = random.randint(0,100000)
```

```
    Z=[a]
```

```
    if str(b).isalpha()==True:
```

```
        root2 = Tk()
```

```
        root2.title("idk23")
```

```
        root2.geometry("400x200")
```

```
label1 = Label(root2, text = "CANNOT ENTER ALPHABETS",bd = 100,
font = "Castellar")

label1.pack()

root2.mainloop()

pass

if str(b)==" " or str(b).isspace()==True:

    root34 = Tk()

    root34.title("idk23")

    root34.geometry("400x200")

    label121 = Label(root34, text = "SCORE CANNOT BE EMPTY",bd = 100,
font = "Castellar")

    label121.pack()

    root34.mainloop()

    pass

if int(b) > 20:

    root42 = Tk()

    root42.title("idk23")

    root42.geometry("400x200")

    label13 = Label(root42, text = "CANNOT BE MORE THAN 20",bd = 100,
font = "Castellar")

    label13.pack()
```



```
root42.mainloop()
```

```
pass
```

```
if str(a)==" " or str(a).isspace()==True:
```

```
    root3 = Tk()
```

```
    root3.title("idk23")
```

```
    root3.geometry("400x200")
```

```
    label12 = Label(root3, text = "NAME CANNOT BE EMPTY",bd = 100,  
font = "Castellar")
```

```
    label12.pack()
```

```
    root3.mainloop()
```

```
else:
```

```
    for i in range(0,1):
```

```
        A.append(a)
```

```
        A.append(b)
```

```
        A.append(c)
```

```
#try the above

my_tree1.insert(parent="",index='end', text="",
values=(name_box1.get(),score1.get(),c))

t1 = tuple(A)

entr = 'Insert into record(name,score,id) values(%s,%s,%s)'
cur.execute(entr,t1)

#to clear the boxes

name_box1.delete(0,END)

score1.delete(0,END)

con.commit()
```

```
my_tree1 = ttk.Treeview(root1)

add_frame1=Frame(root1)

add_frame1.pack(pady=20)


name_box1 = Entry(add_frame1)

name_box1.grid(row=1,column=0)
```

```
score1 = Entry(add_frame1)
score1.grid(row=1,column=1)
```

```
nl1 = Label(add_frame1, text="Name")
nl1.grid(row=0,column=0)
```

```
il1 = Label(add_frame1, text="Score")
il1.grid(row=0, column=1)
```

```
add_record1 = Button(root1, text="Add Record", command=add_record1)
add_record1.pack(pady=20)
```

```
def update_name_window():  
    A1=[]  
    root1 = Tk()  
    root1.title("idk2")  
    root1.geometry("200x300")  
    add_frame1=Frame(root1)  
    add_frame1.pack(pady=20)  
    prompt0= Label(add_frame1, text='Enter game id')  
    prompt0.grid(row=200, column=0)  
  
    user_id=Entry(add_frame1)  
    user_id.grid(row=250, column=0)  
    c=user_id.get()  
  
    prompt= Label(add_frame1, text='Enter the new username')  
    prompt.grid(row=100,column=0)  
    new_name = Entry(add_frame1)  
    new_name.grid(row=150,column=0)  
    user_new=new_name.get()  
  
    def update_name(user_new,c):
```

```
A1=[user_new, c]
```

```
up = 'Update record set name = %s where id = %s'
```

```
t11 = tuple(A1)
```

```
cur.execute(up,t11)
```

```
con.commit()
```

```
update_record = Button(root1, text="update record",  
command=update_name(user_new,c))
```

```
update_record.pack(pady=20)
```

```
def admin():
```

```
root = Tk()
```

```
root.title("idk")
```

```
root.geometry("500x600")
```

```
my_tree = ttk.Treeview(root)

#define our columns
my_tree['columns'] = ("Name", "GAME SCORE", "ID")

#Format our columns
my_tree.column("#0",width=0,minwidth=2)
my_tree.column("Name",anchor=W,width=100)
my_tree.column("GAME SCORE",anchor=CENTER,width=100)
my_tree.column("ID",anchor=E,width=140)


add_frame=Frame(root)
add_frame.pack(pady=20)

nl = Label(add_frame, text="Name")
nl.grid(row=0,column=0)


tl = Label(add_frame, text="Game Score")
```

```
tl.grid(row=0,column=1)
```

```
def add_record():
```

```
    global count
```

```
    #to clear the boxes
```

```
    A=[]
```

```
    a=name_box.get()
```

```
    b=game_score.get()
```

```
    c=random.randint(0,888888)
```

```
    if str(b).isalpha() == True:
```

```
        root2 = Tk()
```

```
        root2.title("idk23")
```

```
        root2.geometry("400x200")
```

```
        label1 = Label(root2, text = "CANNOT ENTER ALPHABETS",bd = 100,  
font = "Castellar")
```

```
label1.pack()

root2.mainloop()

pass

if str(b)==" " or str(b).isspace()==True:

    root34 = Tk()

    root34.title("idk23")

    root34.geometry("400x200")

    label121 = Label(root34, text = "SCORE CANNOT BE EMPTY",bd = 100,
font = "Castellar")

    label121.pack()

    root34.mainloop()

    pass

if int(b) >20:

    root42 = Tk()

    root42.title("idk23")

    root42.geometry("400x200")

    label13 = Label(root42, text = "SCORE CANNOT BE MORE THAN
20",bd = 100, font = "Castellar")

    label13.pack()

    root42.mainloop()
```


pass

if str(a)==" " or str(a).isspace()==True: #not working

root3 = Tk()

root3.title("idk23")

root3.geometry("400x200")

label12 = Label(root3, text = "NAME CANNOT BE EMPTY",bd = 100,
font = "Castellar")

label12.pack()

root3.mainloop()

else:

for i in range(0,1):

A.append(a)

A.append(b)

A.append(c)

#try the above

my_tree.insert(parent="",index='end', text="",
values=(name_box.get(),game_score.get(),c))

t1 = tuple(A)

entr = 'Insert into record(name,score,id) values(%s,%s,%s)'

```
        cur.execute(entr,t1)

        #to clear the boxes

        name_box.delete(0,END)

        game_score.delete(0,END)

        con.commit()

        name_box.delete(0,END)


        game_score.delete(0,END)


def remove_all():

    cur.execute('delete from record')


    for record in my_tree.get_children():

        my_tree.delete(record)

    con.commit()


update_record = Button(root, text="Update Record",
command=update_name_window)

update_record.pack(pady=20)
```

```
name_box = Entry(add_frame)
name_box.grid(row=1,column=0)
```

```
game_score = Entry(add_frame)
game_score.grid(row=1,column=1)
```

```
add_record = Button(root, text="Add Record", command=add_record)
add_record.pack(pady=20)
```

```
#create headings
my_tree.heading("#0",text="",)
my_tree.heading("Name",text="Name",anchor=W)
my_tree.heading("GAME SCORE",text="GAME SCORE",anchor=CENTER)
my_tree.heading("ID",text="ID",anchor=E)
```



```
root.mainloop()
```

```
# Center the Game Application
```

```
os.environ['SDL_VIDEO_CENTERED'] = '1'
```

```
def game():
```

```
    import math
```

```
    import pygame
```

```
    import sys
```

```
    import os
```

```
    import random
```

```
    pygame.init()
```

```
    popup()
```

```
screen = pygame.display.set_mode((800,600))
```

```
pygame.display.set_caption('Hungry Bird')
```

```
icon = pygame.image.load('in.png')
```

```
pygame.display.set_icon(icon)
```

```
bg = pygame.image.load('rainforest.png')
```

```
bg_sound = mixer.music.load("ambience.wav")
```

```
mixer.music.play(-1)

score_value=0

live=3

font = pygame.font.Font("Retro.ttf",30)

textx=10

texty=10

textx1 = 600

texty1 = 10

def scores(x,y):

    score = font.render("Score:" +str(score_value),True,(255,255,255))

    screen.blit(score,(x,y))

def gameover():

    overfont=pygame.font.Font("Retro.ttf",100)

    over_text=overfont.render("GAME OVER",True,(255,0,0))

    screen.blit(over_text, (250, 250))

def gamewon():

    wonfont=pygame.font.Font("Retro.ttf",100)

    won_text=wonfont.render("YOU WON!!", True,(255,0,0))

    screen.blit(won_text, (250, 250))

def lives(x,y):
```

```
    life = font.render('Lives Left :'+ str(live),True,(255,255,255))

    screen.blit(life,(x,y))

playerimg = pygame.image.load('doveR.png')
playerx = 370
playery = 480
x_change = 0
y_change = 0
def player(x,y):

    screen.blit(playerimg,(x,y))
enemyimg = []
enemyx = []
enemyy = []
x_z = []
y_z = []
numenemy = random.randint(3,4)
for i in range(numenemy):
    enemyimg.append(pygame.image.load('ladybugs.png'))
    enemyx.append(random.randint(10,730))
    enemyy.append(random.randint(20,100))
    x_z.append(random.randint(1,2))
    y_z.append(random.randint(1,2))
```

```
def enemy(x,y,i):
    screen.blit(enemyimg[i],(x,y))

def isCollision(enemyx,enemyy,playerx,playery,i):
    distance = math.sqrt((math.pow(enemyx[i]-
playerx,2))+(math.pow(enemyy[i]-playery+5,2)))

    if distance<30:
        return True
    else:
        return False

slashimg = pygame.image.load("45.png")
def slash(x,y):
    screen.blit(slashimg,(x,y+10))

running = True
while running:
    screen.fill((0,0,0))
    screen.blit(bg,(0,0))
    for event in pygame.event.get():
        if event.type == pygame.QUIT:

            pygame.quit()
```



```
try:
    sys.exit()
finally:
    running = False
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_LEFT:
        x_change = -5
    if event.key == pygame.K_RIGHT:
        x_change = 5
    if event.key == pygame.K_UP:
        y_change = -5
    if event.key == pygame.K_DOWN:
        y_change = 5
    if event.key == pygame.K_KP0:
        snd = mixer.Sound("gulp.mp3")
        snd.play()
        slash(playerx,playery)
    for i in range(numenemy):
        collision = isCollision(enemyx,enemyy,playerx,playery,i)
        if collision:
            score_value += 1
            enemyx[i] = random.randint(10,650)
```

```
enemyy[i] = random.randint(20,50)
```

```
if event.type == pygame.KEYUP:
```

```
    if event.key == pygame.K_LEFT:
```

```
        x_change = 0
```

```
    if event.key == pygame.K_RIGHT:
```

```
        x_change = 0
```

```
    if event.key == pygame.K_UP:
```

```
        y_change = 0
```

```
    if event.key == pygame.K_DOWN:
```

```
        y_change = 0
```

```
    if event.key == pygame.K_KP0:
```

```
        slash(playerx,playery)
```

```
for i in range(numenemy):
```

```
    collision = isCollision(enemyx,enemyy,playerx,playery,i)
```

```
    if collision:
```

```
        score_value += 1
```

```
        enemyx[i] = random.randint(10,650)
```

```
        enemyy[i] = random.randint(20,50)
```

```
playerx = playerx + x_change
playery = playery + y_change
#GameOver initialized
if live ==0:
    for j in range(numenemy):
        enemyy[j]=2000
    while live==0:
        playery= 2000
        playerx= 2000
        break
    gameover()
if playerx <= 0:
    playerx = 0
elif playerx >= 736: #Taking size of img in mind
    playerx = 736
if playery <= 0:
    playery = 0
elif playery >= 536:
    playery = 536

for i in range(numenemy):
    enemyx[i] = enemyx[i] + x_z[i]
```

```
enemyy[i] = enemyy[i] + y_z[i]
if enemyx[i] <= 0:
    x_z[i] = 1
elif enemyx[i] >= 736: #Taking size of img in mind
    x_z[i] = -1
if enemyy[i] <= 0:
    y_z[i] = 2
elif enemyy[i] >= 536 and enemyy[i]<=537:
    live-=1
```

```
enemy(enemyx[i],enemyy[i],i)
player(playerx,playery)
```

```
scores(textx,texty)
```

```
lives(textx1,texty1)
```

```
# Initialises Gamewon
```

```
if score_value==20:
```

```
    gamewon()
```

```
    for j in range(numenemy):
```

```
        enemyy[j]=2000
```

```
while score_value>=0:
```

```
    playery= 2000
```

```
    playerx= 2000
```

```
    break
```

```
pygame.display.update()
```

```
# Game Resolution
```

```
screen_width=800
```

```
screen_height=600
```

```
screen=pygame.display.set_mode((screen_width, screen_height))
```

```
# Text Renderer
```

```
def text_format(message, textFont, textSize, textColor):
```

```
    newFont=pygame.font.Font(textFont, textSize)
```

```
    newText=newFont.render(message, 0, textColor)
```

```
return newText
```

Colors

```
white=(255, 255, 255)
```

```
black=(0, 0, 0)
```

```
gray=(50, 50, 50)
```

```
red=(255, 0, 0)
```

```
green=(0, 255, 0)
```

```
blue=(0, 0, 255)
```

```
yellow=(255, 255, 0)
```

Game Font

```
font = "Retro.ttf"
```

Game Framerate

```
clock = pygame.time.Clock()
```

```
FPS=30
```

Main Menu

```
def main_menu():  
    icon = pygame.image.load('book.png')  
    pygame.display.set_icon(icon)  
  
    menu=True  
    selected="start"  
  
    while menu:  
        bgk = pygame.image.load('i01_bird.png')  
  
        for event in pygame.event.get():  
            if event.type==pygame.QUIT:  
                pygame.quit()  
                quit()  
            if event.type==pygame.KEYDOWN:  
                if event.key==pygame.K_UP:  
                    selected="start"  
                elif event.key==pygame.K_DOWN:  
                    selected="quit"
```

```
elif event.key==pygame.K_LEFT:
```

```
    selected="admin"
```

```
if event.key==pygame.K_RETURN:
```

```
    if selected=="start":# here put game function
```

```
        game()
```

```
        popup()
```

```
if selected=="quit":
```

```
    pygame.quit()
```

```
    quit()
```

```
if selected=="admin":
```

```
    admin()
```

```
# Main Menu
```

```
screen.fill(blue)
```

```
title=text_format("Hungry Bird", font, 90, yellow)
```

```
if selected=="start":
```

```
    text_start=text_format("START", font, 75, white)
```

```
else:
```

```
    text_start = text_format("START", font, 75, black)
```

```
if selected=="quit":
```



```
text_quit=text_format("QUIT", font, 75, white)
else:
    text_quit = text_format("QUIT", font, 75, black)
if selected=="admin":
    text_admin=text_format("ADMIN", font, 75, white)
else:
    text_admin=text_format("ADMIN", font, 75, black)

title_rect=title.get_rect()
start_rect=text_start.get_rect()
quit_rect=text_quit.get_rect()
admin_rect=text_admin.get_rect()

# Main Menu Text
screen.blit(bgk,(0,0))
screen.blit(title, (screen_width/2 - (title_rect[2]/2), 80))
screen.blit(text_start, (screen_width/2 - (start_rect[2]/2), 300))
screen.blit(text_quit, (screen_width/2 - (quit_rect[2]/2), 360))
screen.blit(text_admin, (screen_width/2 - (admin_rect[2]/2), 420))
pygame.display.update()
```

```
clock.tick(FPS)
```

```
pygame.display.set_caption("WELCOME TO THE GAME")
```

```
#Initialize the Game
```

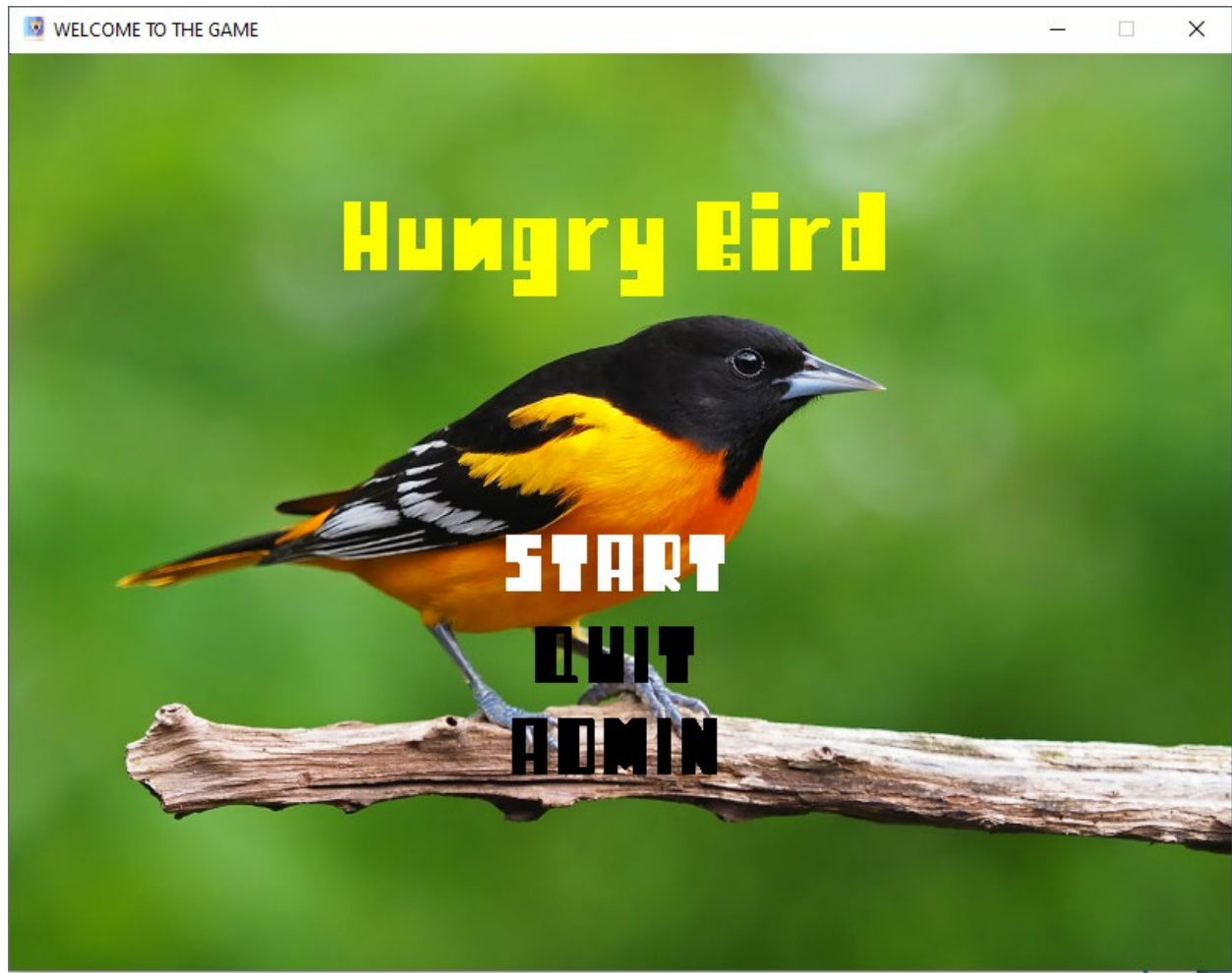
```
main_menu()
```

```
con.commit()
```

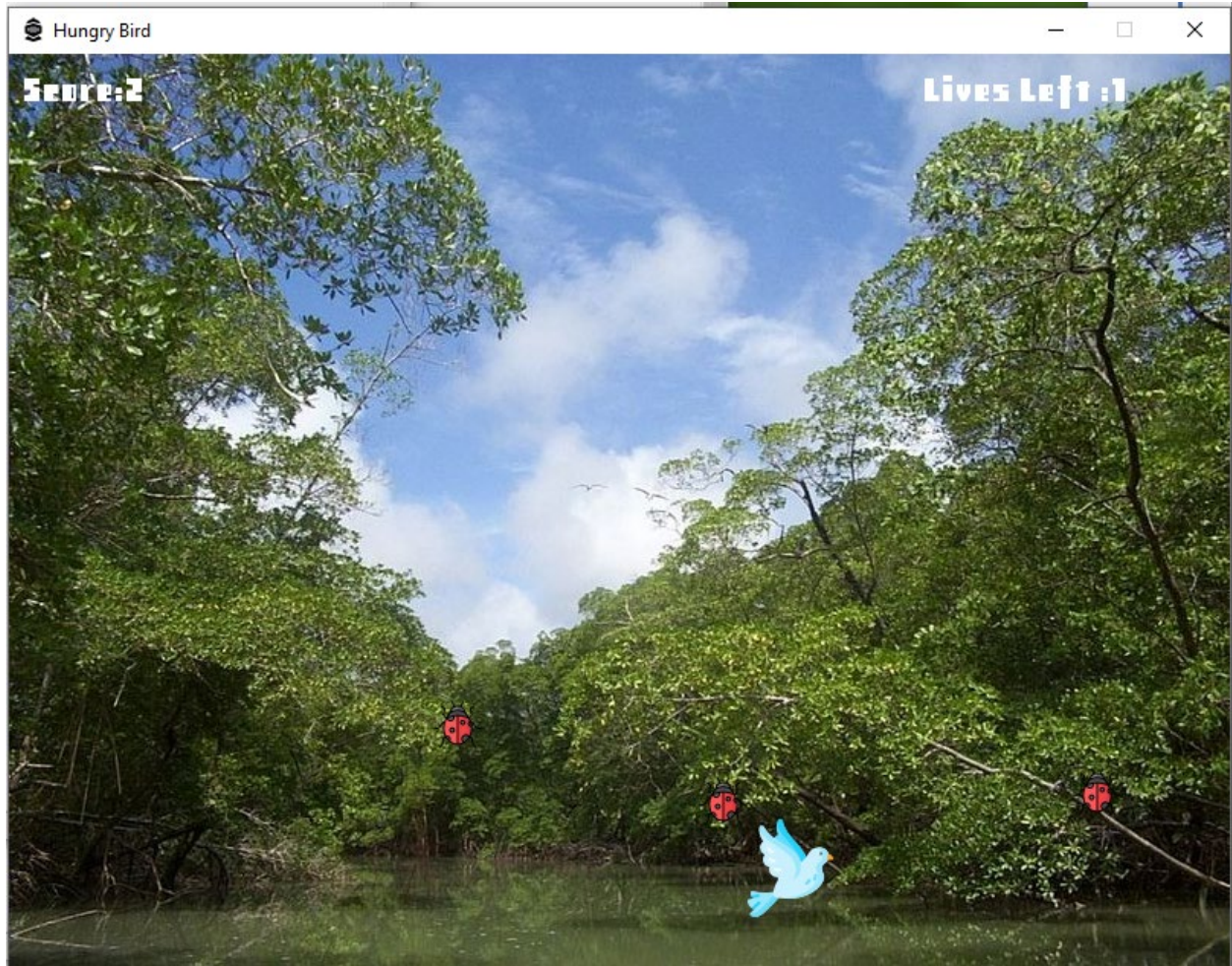
```
pygame.quit()
```

```
quit()
```

OUTPUT



this is the main menu screen which pops up when the program starts



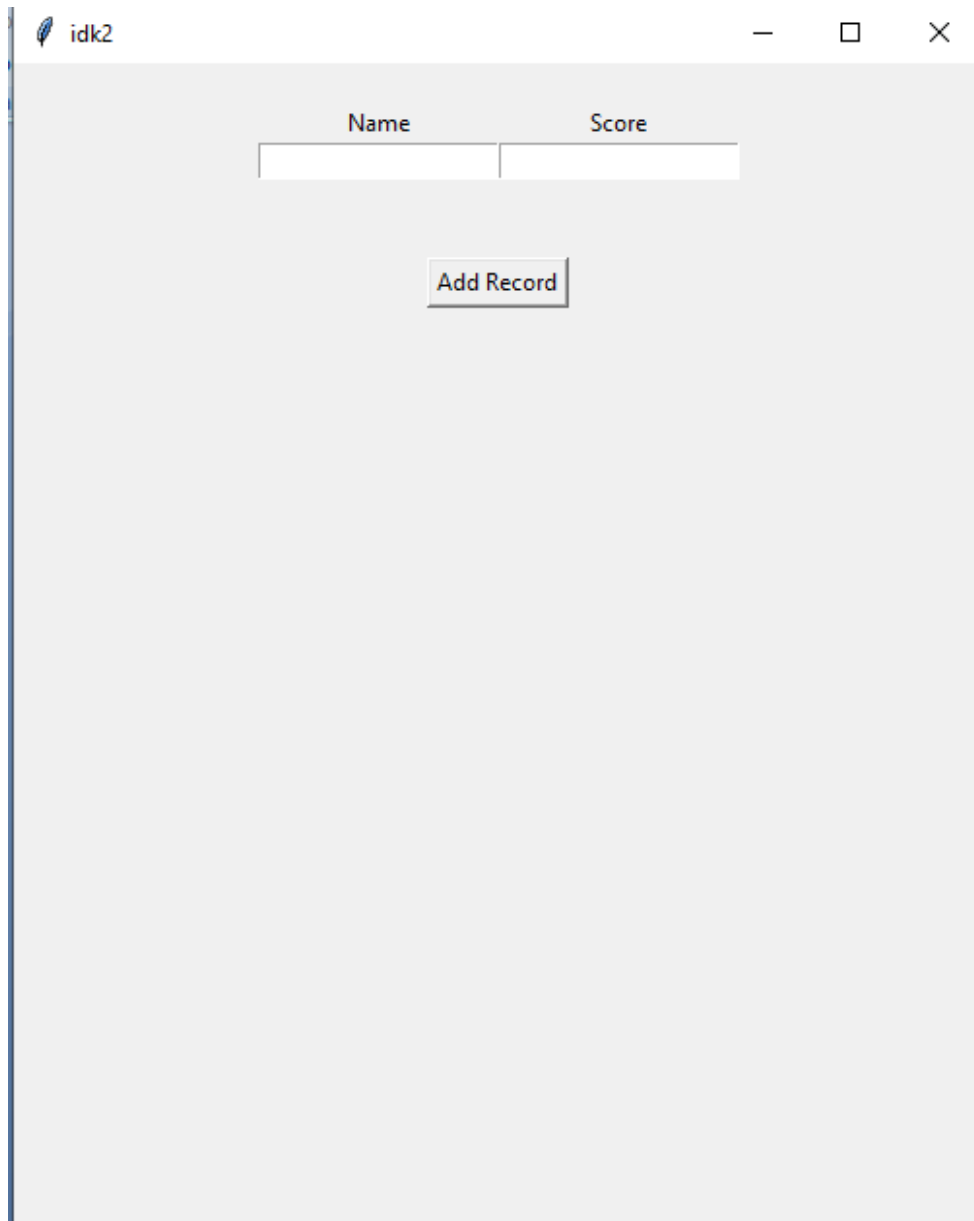
This is what the game looks like in action. You can see the score on the top left hand side and the lives left on the top right side.



#This is the game screen when you run out of lives i.e lose the game.

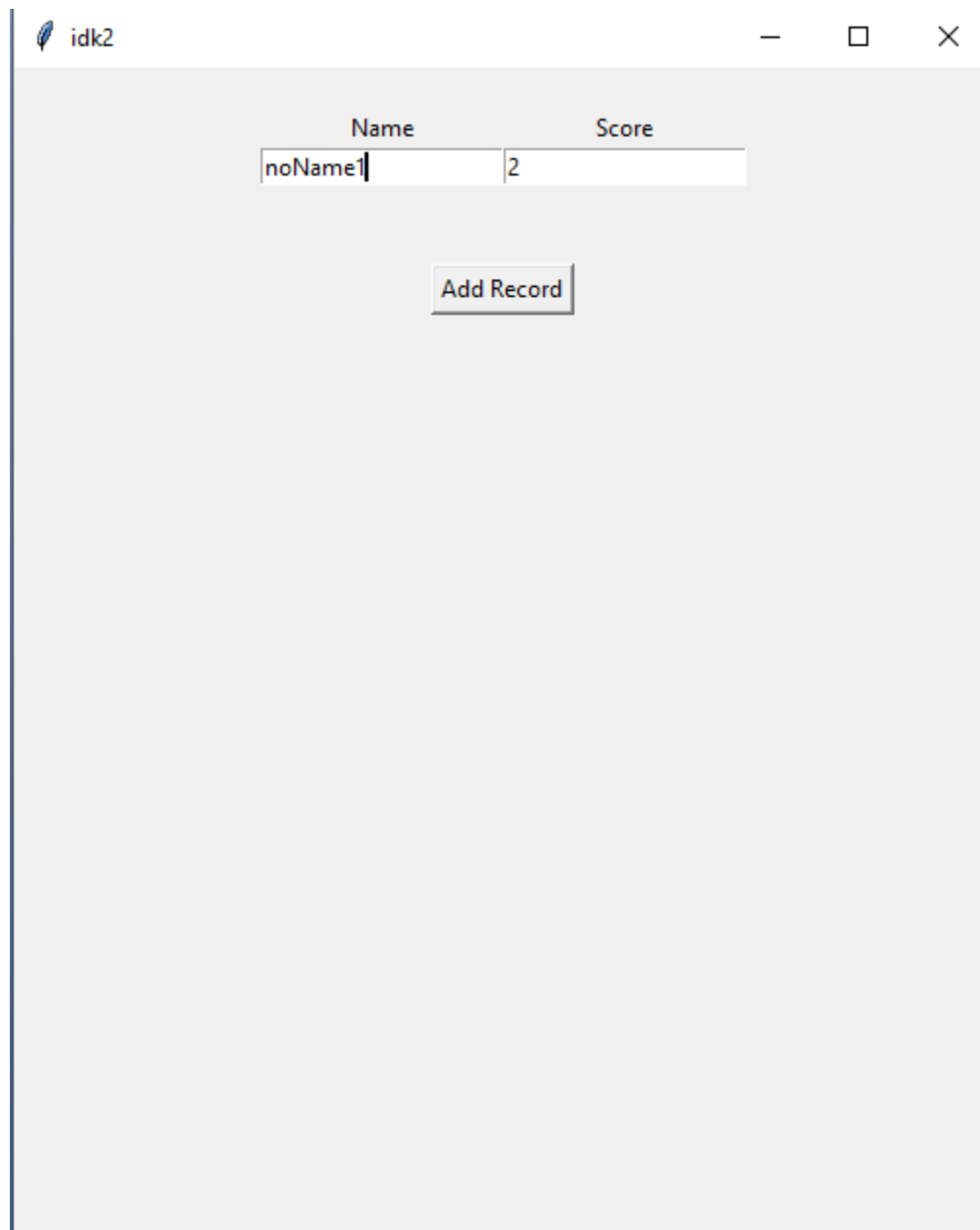
#This is the game screen when you have won the game i.e reached 20 points.





The screenshot shows a Java Swing window titled "idk2". Inside the window, there is a form with two input fields. The first field is labeled "Name" and the second field is labeled "Score". Below these fields is a button labeled "Add Record". The window has standard Windows-style controls (minimize, maximize, close) in the top right corner.

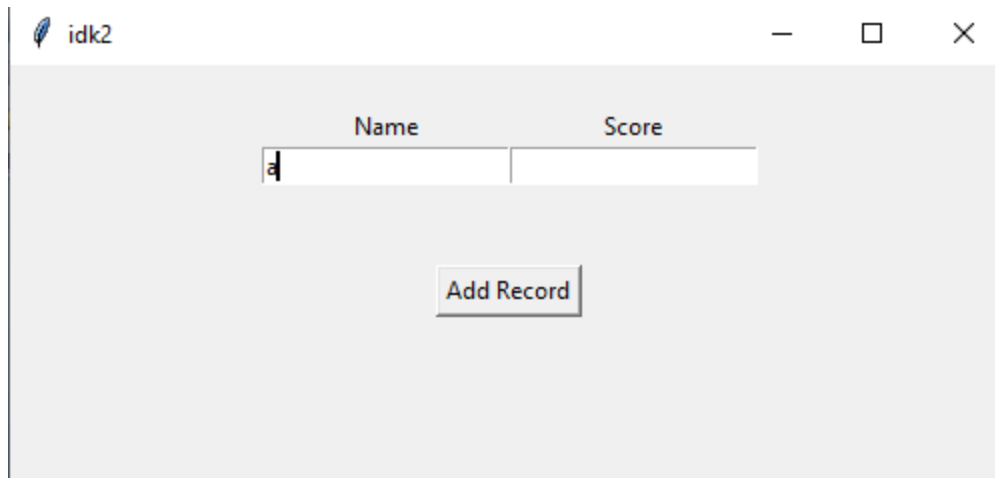
#This window pops up at the end of the game, for the player to enter their name and score.



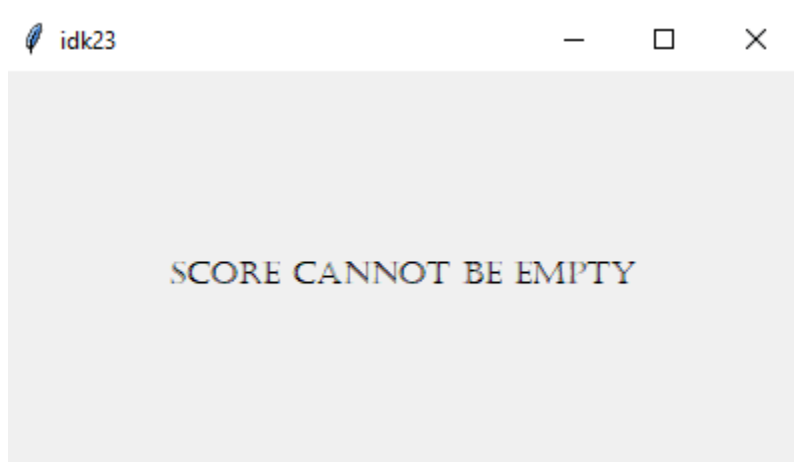
The screenshot shows a Java Swing window titled "idk2". Inside the window, there is a form with two input fields. The first field is labeled "Name" and contains the text "noName1". The second field is labeled "Score" and contains the number "2". Below these fields is a button labeled "Add Record".

Name	Score
noName1	2

Add Record



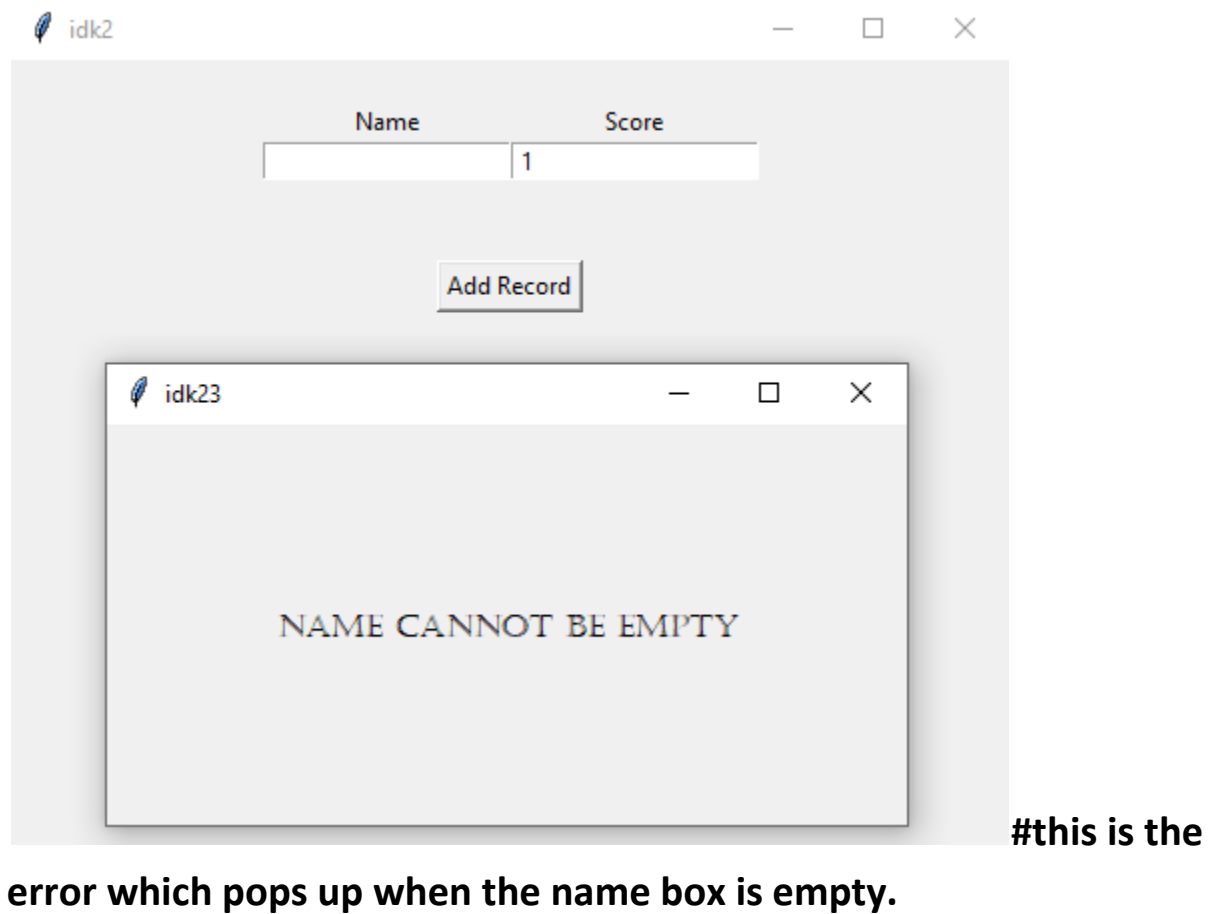
The screenshot shows a Java Swing window titled "idk2". Inside the window, there is a form with two input fields: "Name" and "Score". The "Name" field contains the character 'a'. Below these fields is a button labeled "Add Record".

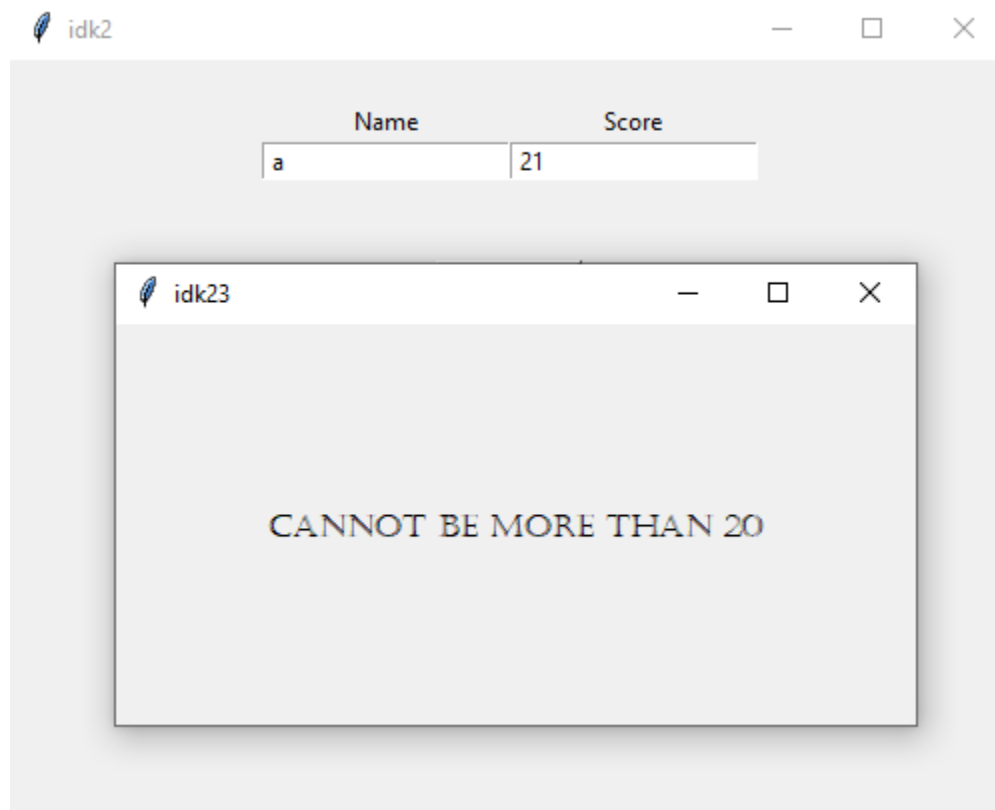


The screenshot shows a Java Swing window titled "idk23". The window displays a message in a monospaced font: "SCORE CANNOT BE EMPTY".

This is the error

which pops up when the score box is empty.





#This is the error when the score entered is more than 20

The screenshot shows a window titled 'idk' with a light gray background. At the top, there are two input fields labeled 'Name' and 'Game Score'. Below these fields are three buttons: 'Update Record', 'Add Record', and 'Remove all records'. At the bottom, there is a table with three columns: 'Name', 'GAME SCORE', and 'ID'. The table contains two rows of data.

Name	GAME SCORE	ID
noName1	2	31196
noName2	20	762874

#This is the admin window , where all the records entered are visible.

idk

Name Game Score

noName3 13

Update Record

Add Record

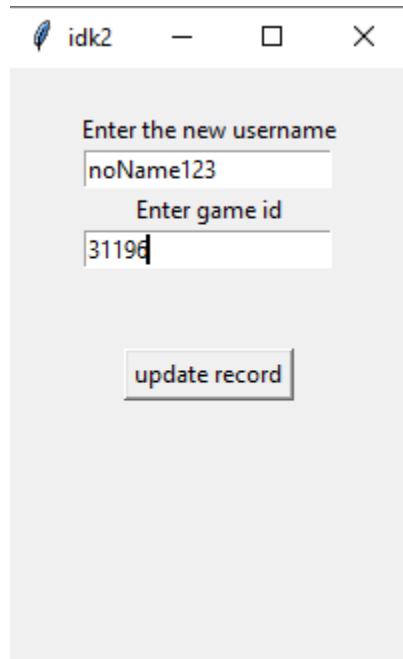
Remove all records

Name	GAME SCORE	ID
noName1	2	31196
noName2	20	762874

#adding a record from admin mode

```
mysql> select * from record;
+-----+-----+-----+
| name   | score | id     |
+-----+-----+-----+
| noName1 | 2     | 31196  |
| noName2 | 20    | 762874 |
| noName3 | 13    | 488298 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

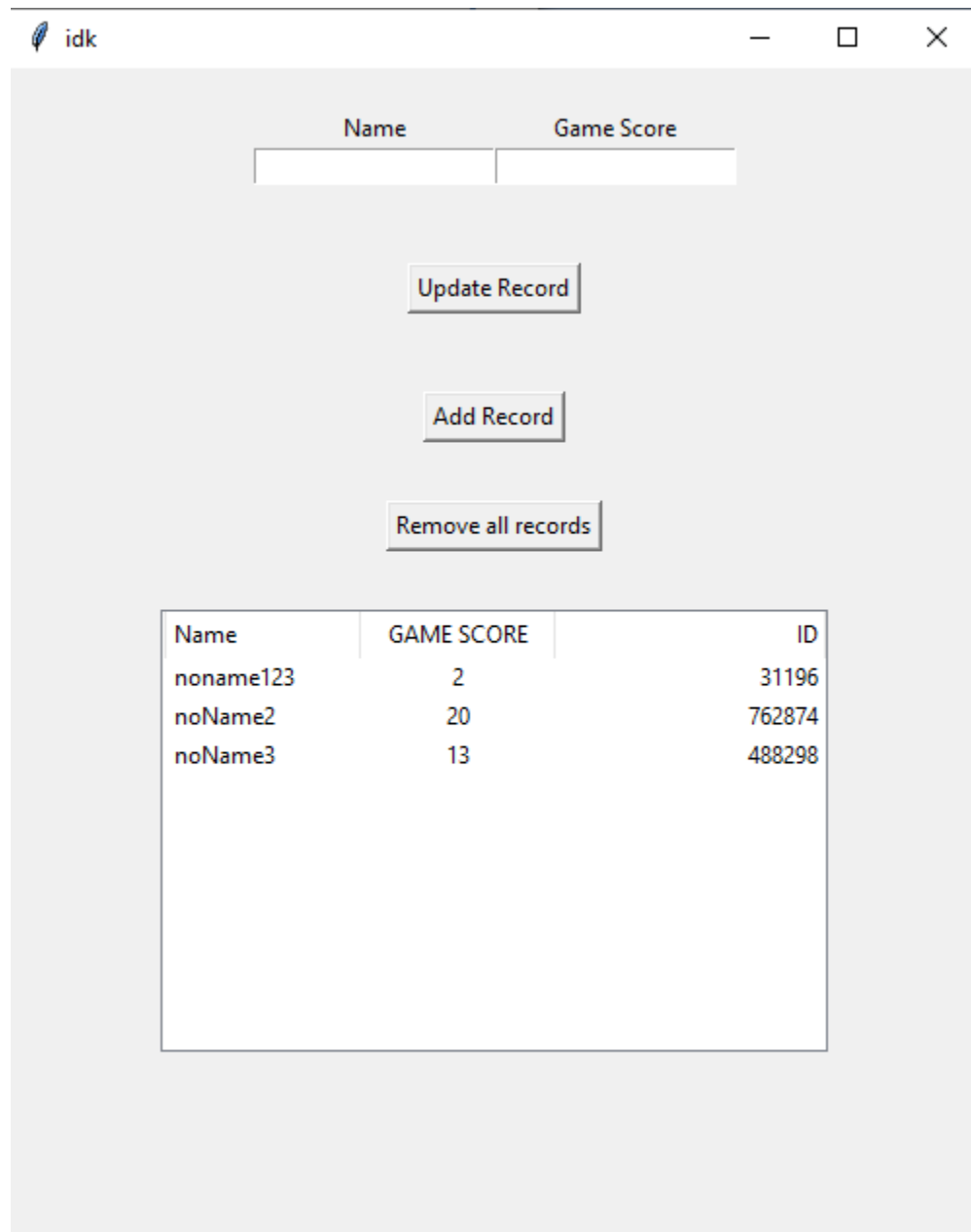
#record added through admin in mysql



#the update function does not show change on the front end immediately on the front end but does so on the back end i.e in mysql.

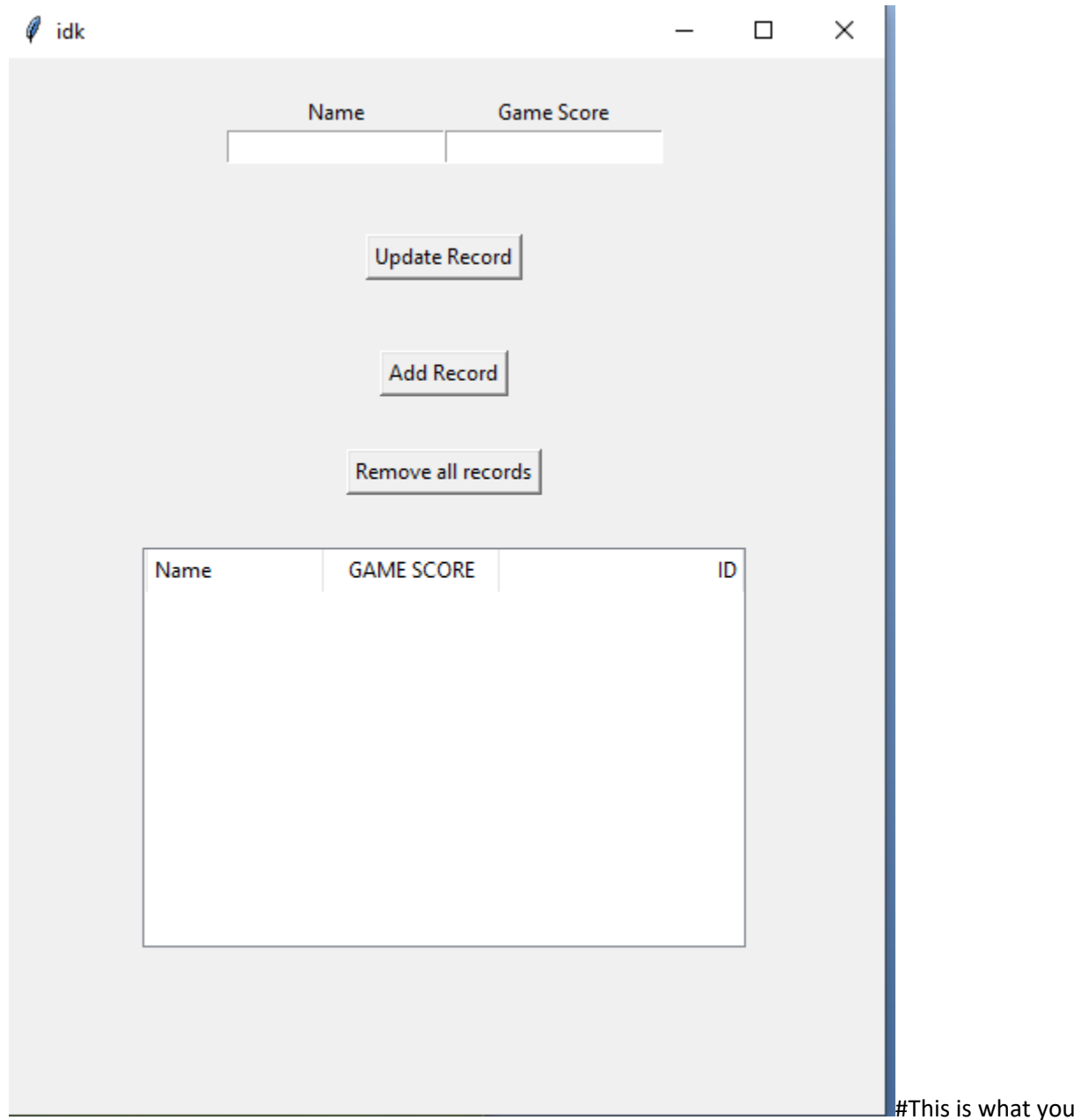
```
mysql> select * from record;
+-----+-----+-----+
| name       | score | id     |
+-----+-----+-----+
| noname123  | 2     | 31196  |
| noName2    | 20    | 762874 |
| noName3    | 13    | 488298 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

as you can see the changes are taking place on the back end in the first run



Name	GAME SCORE	ID
noname123	2	31196
noName2	20	762874
noName3	13	488298

The changes are visible on the front end too when you run the game again



The screenshot shows a Java Swing window titled "idk" with a light gray background. At the top, there are two text input fields labeled "Name" and "Game Score". Below these fields are three buttons: "Update Record", "Add Record", and "Remove all records". At the bottom of the window is a table with three columns: "Name", "GAME SCORE", and "ID". The table is currently empty.

Name	GAME SCORE	ID
------	------------	----

get when you press remove all records

```
mysql> select * from record;  
Empty set (0.00 sec)
```

#the table become empty after pressing the remove all records button

Media Used1



: in.png



:rainforest.png



: doveR.png



:45.png



: book.png



:i01_bird.png



: ladybugs.png

Font used is Retro.ttf in the program.

CONCLUSION

The entire project that has been displayed before this page, the hundreds of line of code written and the tremendous amount of time and energy spent by the group on this project has successfully borne fruit and the game works quite well.

This project shows the working of SQL, Python and Pygame, and also provides a moderately well made game that people can spend their time playing.

Further Enhancements

Due to constraints on time, knowledge and expertise we were not able to make the following enhancements:

- We wanted to make our Admin mode accessible only to the developers or “Admin” by making it password oriented.
- Our update function is not fully functioning as expected. It is not changing the data on the front end but it is doing so on the back end which is mysql.
- The score input system is quite counter intuitive as such that the user has to manually enter the score. We hoped to turn it into a automated storage system but it proved to be beyond our capabilities.

Bibliography

- <https://www.google.co.in>
- <https://stackoverflow.com>
- <https://www.youtube.com>
- <https://github.com>
- [Computer science with Python – Sumita Arora](#)
- <https://www.geeksforgeeks.org>
- <https://www.w3schools.com>
- <https://soundcloud.com>