**FLIPPING THE TILES**

*A*

*Mini Project Report*

*Submitted in partial fulfilment of the Requirements for the award of the Degree of*

# BACHELOR OF ENGINEERING

IN

# INFORMATION TECHNOLOGY

By

#### Shreya Chepuri 1602-20-737-040

**Akshitha Namplally 1602-20-737-004**

**Anjan Sai Jagarlamudi 1602-20-737-006**



# Department of Information Technology Vasavi College of Engineering (Autonomous)

**ACCREDITED BY NAAC WITH 'A++' GRADE**

# (Affiliated to Osmania University and Approved by AICTE) Ibrahim Bagh, Hyderabad-31

**2022**

# Vasavi College of Engineering (Autonomous)

**ACCREDITED BY NAAC WITH 'A++' GRADE**

# (Affiliated to Osmania University and Approved by AICTE) Hyderabad-500 031

**Department of Information Technology**



**DECLARATION BY THE CANDIDATE**

#### We, Shreya chepuri, Akshitha Nampally and Anjan Sai Jagarlamudi,

bearing hall ticket numbers, **1602-20-737-040,1602-20-737-004** and **1602-20-737-006**, hereby declare that the project report entitled **"FLIPPING THE TILES"** is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**Shreya Chepuri**

**1602-20-737-040**

**Akshitha Nampally**

**1602-20-737-004**

**Anjan Sai Jagarlamudi**

**1602-20-737-006**

(Faculty In-Charge) (Head,Dept of IT)

**ACKNOWLEDGMENT**

We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering for his encouragement.

We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi College of Engineering, for introducing the Mini-Project module in our curriculum, and also for his suggestions, motivation, and co-operation for the successful completion of our Mini Project.

We also want to thank and convey our gratitude towards our mini project coordinators Divya Lingineni and RajyaLakshmi, for guiding us in understanding the process of project development & giving us timely suggestions at every phase.

We would also like to sincerely thank the project reviewers for their valuable inputs and suggestions

**ABSTRACT**

The main objective of the project is to provide a memory game in a simple way. This project is useful for any kind of people especially young minds who wants to improve their memory power.

When the game starts, all the tiles are turned face down. The player then flips over two cards, selecting them by clicking on them. If the two tiles have the same image, they remain face up.

If not, they will be flipped face down again after a short delay. The person who finishes the game in least possible moves will be the highest to score.

The framework which was used in this project was GUI using python.

**TABLE OF CONTENTS**

**1.Introduction**

a. Information

b. Features

**2.Technology**

a. Software Requirements

b. Hardware Requirements

**3.Proposed Work**

a. Design

b. Implementation

c. Testing

**4.Results**

**5.Outcomes apart from curriculum**

**6.Conclusion and future work**

**7. References**

**INTRODUCTION**

**a. Information**

As kids grow up and head towards school age, the importance of learning through playing increases. Through games and playing, children can nurture their imagination, improve their ability to listen, as well as develop important cognitive skills that will aid their future development and memory games are a vital part of this.

Everyone has played some kind of memory game at some point. Whether that’s spot the difference, simple pairing games using playing cards, or even doing a crossword. They all require the players to use their memory to complete the game and in doing so, users are developing their key skills.

**b. Motivation**

Playing memory games can improve other brain functions, such as attention, concentration, and focus. Memory games give space to critical thinking and that helps children nurture their attention to detail.

Memory games can improve visual recognition. With many memory games based on spotting differences, or linking two related images, children improve their visual discrimination. This will lead to an acceleration in distinguishing images from one and another.

Short-term memory is key to playing memory games and playing them often will improve function in this area. A good short-term memory can improve a person's long-term memory too. Both are linked and being able to move things from your short-term memory into long-term will improve learning in other areas.

Though memory games are a short-term boost, players have to plan their moves as they go. From revealing a card to plotting their next move, children can learn the importance of thinking ahead and plotting their next choice.

**c. Features**

Our objective is to create a simple memory game where young minds can improve their memory power and "Flipping the tiles" is one such game.

●In this, we have a certain even number of tiles, in which each number or figure has a pair.

●The tiles are facing downwards, and we have to flip them to see them.

●In a turn, one flips 2 tiles, if the tiles match, then they are removed.

● If not then they are flipped and placed back in the position.

●We keep on doing this until all the tiles have been matched and removed.

**Technology**

**a. Hardware Requirements**

The hardware components used in order to create the game are as follows

* Computer with 2GB RAM
* Internet
* External drives for backup

**b. Software Requirements**

* We use python Idle in order to develop the code for the game.
* We use my files in order to store the data for leaderboard i.e. user name and number of moves.
* Operating System. We have chosen Windows operating system for its best support and user friendliness.

**PROPOSED WORK**

**a. Design**

**Use Case Diagram**

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how).

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

**Use Case Diagram**

****

**DESCRIPTION**

**User**

The user has to run the program and selects whatever he wants from the main menu which contains options like play, leaderboard, Instructions and exit according to the need.

**System**

TheSystem responds according to the user actions and gives the information whatever he wants.

**Levels**

The user can select the range (Level) of the game he wants play i.e., Easy, Medium, and Hard, system responds to it and displays the level of user's choice

**Leaderboard**

The user can view the highest scores in a particular level with help of Leaderboard.

**Instructions**

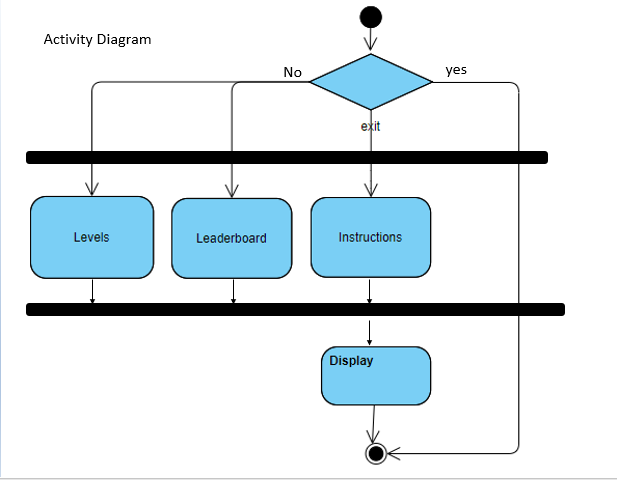
If the user is new to the game, then he can select Instructions in order to get the rule of the

game.

**Exit**

It allows user to exit from the game

**ACTIVITY DIAGRAM**

****

**MODULE WISE CODE**

from tkinter import \*

import random

from tkinter import ttk

import time

print("\n\n\n \n\n\n")

print("\t\*0\* \* 0 \* 0 \* 0 \* 0 \* 0 \* 0 \* 0 \* 0 \* 0 \* 0\n")

print("\t\*0\* \* FLIPPING THE TILES \* 0\n")

print("\t\*0\* \* 0 \* 0 \* 0 \* 0 \* 0 \* 0 \* 0 \* 0 \* 0 \* 0\n")

print("\n\n \n\n")

global moves1

def play():

print("Enter your name:")

global s

s=input();

print("\n\t\t\tEnter Level:\n\n");

print("\t\t\t1.Easy\n");

print("\t\t\t2.medium\n");

print("\t\t\t3.Hard\n");

print("\t\t\tEnter 4 to exit\n");

print("\t\t\tEnter 5 to go bck to main menu:\n")

print("\t\tEnter your choice:\n\n")

option = int(input());

if(option==1):

PuzzleWindow=Tk()

PuzzleWindow.title('Flipping the tiles')

tabs = ttk.Notebook(PuzzleWindow)

easy= ttk.Frame(tabs)

tabs.add(easy, text ='Easy')

tabs.pack(expand = 1, fill ="both")

global base1,ans1,board1,moves1,prev1

def draw(a,l,m):

global base1

if a=='A':

d=base1.create\_rectangle(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='red')

elif a=='B':

d=base1.create\_rectangle(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='yellow')

elif a=='C':

d=base1.create\_rectangle(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='blue')

elif a=='D':

d=base1.create\_oval(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='red')

elif a=='E':

d=base1.create\_oval(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='yellow')

elif a=='F':

d=base1.create\_oval(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='blue')

elif a=='G':

d=base1.create\_polygon(100\*l+50,m\*100+20,100\*l+20,100\*m+100-20,100\*l+100-20,100\*m+100-20,fill='red')

elif a=='H':

d=base1.create\_polygon(100\*l+50,m\*100+20,100\*l+20,100\*m+100-20,100\*l+100-20,100\*m+100-20,fill='green')

def quizboard():

global base1,ans1,board1,moves1

count=1

for i in range(4):

for j in range(4):

rec=base1.create\_rectangle(100\*i,j\*100,100\*i+100,100\*j+100,fill="white")

if(board1[i][j]!='.'):

draw(board1[i][j],i,j)

count+=1

if count==16:

base1.create\_text(200,450,text="No. of moves: "+str(moves1),font=('arial',20))

with open("high\_score.txt","a") as f:

f.write(s)

f.write(",")

f.write(str(moves1))

f.write("\n")

def call(event):

global base1,ans1,board1,moves1,prev1

i=event.x//100

j=event.y//100

if board1[i][j]!='.':

return

moves1+=1

#print(moves)

if(prev1[0]>4):

prev1[0]=i

prev1[1]=j

board1[i][j]=ans1[i][j]

quizboard()

else:

board1[i][j]=ans1[i][j]

quizboard()

if(ans1[i][j]==board1[prev1[0]][prev1[1]]):

print("matched")

prev1=[100,100]

quizboard()

return

else:

board1[prev1[0]][prev1[1]]='.'

quizboard()

prev1=[i,j]

return

base1=Canvas(easy,width=500,height=500)

base1.pack()

ans1 = list('AABBCCDDEEFFGGHH')

random.shuffle(ans1)

ans1 = [ans1[:4],

ans1[4:8],

ans1[8:12],

ans1[12:]]

base1.bind("<Button-1>", call)

moves1=IntVar()

moves1=0

prev1=[100,100]

board1=[list('.'\*4) for count in range(4)]

quizboard()

elif(option==2):

PuzzleWindow=Tk()

PuzzleWindow.title('Flipping the tiles')

tabs = ttk.Notebook(PuzzleWindow)

easy= ttk.Frame(tabs)

window2= ttk.Frame(tabs)

tabs.add(window2, text ='medium')

tabs.pack(expand = 1, fill ="both")

global base2,ans2,board2,moves2,prev2

def draw1(a,l,m):

global base2

if a=='A':

d=base2.create\_rectangle(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='red')

elif a=='B':

d=base2.create\_rectangle(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='yellow')

elif a=='C':

d=base2.create\_rectangle(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='blue')

elif a=='D':

d=base2.create\_oval(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='red')

elif a=='E':

d=base2.create\_oval(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='yellow')

elif a=='F':

d=base2.create\_oval(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='blue')

elif a=='G':

d=base2.create\_polygon(100\*l+50,m\*100+20,100\*l+20,100\*m+100-20,100\*l+100-20,100\*m+100-20,fill='red')

elif a=='H':

d=base2.create\_polygon(100\*l+50,m\*100+20,100\*l+20,100\*m+100-20,100\*l+100-20,100\*m+100-20,fill='green')

elif a=='I':

d=base2.create\_polygon(100\*l+50,m\*100+20,100\*l+20,100\*m+100-20,100\*l+100-20,100\*m+100-20,fill='yellow')

elif a=='J':

d=base2.create\_polygon(100\*l+50,m\*100+20,100\*l+20,100\*m+100-20,100\*l+100-20,100\*m+100-20,fill='blue')

elif a=='K':

d=base2.create\_polygon(100\*l+50,m\*100+20,100\*l+20,100\*m+100-20,100\*l+100-20,100\*m+100-20,fill='black')

elif a=='L':

d=base2.create\_polygon(100\*l+50,m\*100+20,100\*l+20,100\*m+100-20,100\*l+100-20,100\*m+100-20,fill='orange')

elif a=='M':

d=base2.create\_rectangle(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='black')

elif a=='N':

d=base2.create\_rectangle(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='orange')

elif a=='O':

d=base2.create\_rectangle(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='green')

elif a=='P':

d=base2.create\_oval(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='black')

elif a=='Q':

d=base2.create\_oval(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='orange')

elif a=='R':

d=base2.create\_oval(100\*l+20,m\*100+20,100\*l+100-20,100\*m+100-20,fill='green')

def puzzleboard2():

global base2,ans2,board2,moves2

count=1

for i in range(6):

for j in range(6):

rec=base2.create\_rectangle(100\*i,j\*100,100\*i+100,100\*j+100,fill="white")

if(board2[i][j]!='.'):

draw1(board2[i][j],i,j)

count+=1

if count>36:

base2.create\_text(300,650,text="No. of moves: "+str(moves2),font=('arial',20))

with open("high\_score1.txt","a") as file:

file.write(s)

file.write(",")

file.write(str(moves2))

file.write("\n")

def call2(event):

global base2,ans2,board2,moves2,prev2

i=event.x//100

j=event.y//100

if board2[i][j]!='.':

return

moves2+=1

if(prev2[0]>6):

prev2[0]=i

prev2[1]=j

board2[i][j]=ans2[i][j]

puzzleboard2()

else:

board2[i][j]=ans2[i][j]

puzzleboard2()

if(ans2[i][j]==board2[prev2[0]][prev2[1]]):

prev2=[100,100]

puzzleboard2()

return

else:

board2[prev2[0]][prev2[1]]='.'

puzzleboard2()

prev2=[i,j]

return

base2=Canvas(window2,width=1000,height=1000)

base2.pack()

ans2 = list('AABBCCDDEEFFGGHHIIJJKKLLMMNNOOPPQQRR')

random.shuffle(ans2)

ans2 = [ans2[:6],

ans2[6:12],

ans2[12:18],

ans2[18:24],

ans2[24:30],

ans2[30:]

]

base2.bind("<Button-1>", call2)

moves2=IntVar()

moves2=0

prev2=[100,100]

board2=[list('.'\*6) for count in range(6)]

puzzleboard2()

elif(option==3):

PuzzleWindow=Tk()

PuzzleWindow.title('Flipping the tiles')

tabs = ttk.Notebook(PuzzleWindow)

easy= ttk.Frame(tabs)

window3= ttk.Frame(tabs)

tabs.add(window3, text ='Hard')

tabs.pack(expand = 1, fill ="both")

global base3,ans3,board3,moves3,prev3

def draw2(a,l,m):

global base3

if a=='A':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='red')

elif a=='B':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='yellow')

elif a=='C':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='blue')

elif a=='D':

d=base3.create\_oval(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='red')

elif a=='E':

d=base3.create\_oval(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='yellow')

elif a=='F':

d=base3.create\_oval(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='blue')

elif a=='G':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='red')

elif a=='H':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='green')

elif a=='I':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='yellow')

elif a=='J':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='blue')

elif a=='K':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='black')

elif a=='L':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='orange')

elif a=='M':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='black')

elif a=='N':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='orange')

elif a=='O':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='green')

elif a=='P':

d=base3.create\_oval(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='pink')

elif a=='Q':

d=base3.create\_oval(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='green')

elif a=='R':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='pink')

elif a=='S':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='purple')

elif a=='T':

d=base3.create\_oval(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='purple')

elif a=='U':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='purple')

elif a=='V':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='pink')

elif a=='W':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='maroon')

elif a=='X':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='maroon')

elif a=='Y':

d=base3.create\_oval(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='maroon')

elif a=='Z':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='brown')

elif a=='a':

d=base3.create\_oval(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='brown')

elif a=='b':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='brown')

elif a=='c':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='aqua')

elif a=='d':

d=base3.create\_rectangle(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='aqua')

elif a=='e':

d=base3.create\_oval(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='aqua')

elif a=='f':

d=base3.create\_polygon(80\*l+50,m\*80+20,80\*l+20,80\*m+80-20,80\*l+80-20,80\*m+80-20,fill='magenta')

elif a=='g':

d=base3.create\_oval(80\*l+20,m\*80+20,80\*l+80-20,80\*m+80-20,fill='magenta')

def quizboard3():

global base3,ans3,board3,moves3

count=1

for i in range(8):

for j in range(8):

e=base3.create\_rectangle(80\*i,j\*80,80\*i+80,80\*j+80,fill="white")

if(board3[i][j]!='.'):

draw2(board3[i][j],i,j)

count+=1

if count>64:

base3.create\_text(300,650,text="No. of moves: "+str(moves3),font=('arial',20))

with open("high\_score2.txt","a") as file1:

file1.write(s)

file1.write(",")

file1.write(str(moves1))

file1.write("\n")

def call3(event):

global base3,ans3,board3,moves3,prev3

i=event.x//80

j=event.y//80

if board3[i][j]!='.':

return

moves3+=1

if(prev3[0]>8):

prev3[0]=i

prev3[1]=j

board3[i][j]=ans3[i][j]

quizboard3()

else:

board3[i][j]=ans3[i][j]

quizboard3()

if(ans3[i][j]==board3[prev3[0]][prev3[1]]):

print("matched")

prev3=[100,100]

quizboard3()

return

else:

board3[prev3[0]][prev3[1]]='.'

quizboard3()

prev3=[i,j]

return

base3=Canvas(window3,width=1000,height=1000)

base3.pack()

ans3 = list('AABBCCDDEEFFGGHHIIJJKKLLMMNNOOPPQQRRSSTTUUWWXXYYZZaabbccddeeffgg')

random.shuffle(ans3)

ans3 = [ans3[:8],

ans3[8:16],

ans3[16:24],

ans3[24:32],

ans3[32:40],

ans3[40:48],

ans3[48:56],

ans3[56:]

]

base3.bind("<Button-1>", call3)

moves3=IntVar()

moves3=0

prev3=[80,80]

board3=[list('.'\*8) for count in range(8)]

quizboard3()

elif(option==4):

return;

elif(option==5):

main\_menu()

else:

print("Enter a valid option\n");

print("\n\t\t1.To play again:");

print("\n\t\tPress 2 to return to Main Menu. ");

choice =int(input());

if(choice==1):

play();

elif(choice==2):

main\_menu();

else:

print("\n\t\tThank You for playing!!!");

def instructions():

file1= open("Rules.txt","r");

print(file1.read());

def Easy\_Leaderboard():

global moves1

print("\n\n\*\*\*LEADERBOARD\*\*\*\*\n\n")

high\_score=[]

with open("high\_score.txt") as p:

for line in p:

s,moves1=line.split(',')

moves1=int(moves1)

high\_score.append((s,moves1))

high\_score.sort(key=lambda x:x[1])

if not high\_score:

print("No Recorded Score\n")

print("\n1.For Main\_Menu\n")

d=int(input())

if(d==1):

main\_menu()

else:

exit()

else:

for s,moves1 in high\_score:

print(s,moves1)

print("\n")

print("\n1.For Main\_Menu\n")

c=int(input())

if(c==1):

main\_menu()

else:

exit()

def Med\_Leaderboard():

global moves2,s

print("\n\n\*\*\*LEADERBOARD\*\*\*\*\n\n")

high\_score1=[]

with open("high\_score1.txt") as q:

for line in q:

s,moves2=line.split(',')

moves2=int(moves2)

high\_score1.append((s,moves2))

high\_score1.sort(key=lambda y:y[1])

if not high\_score1:

print("No Recorded Score\n\n")

print("\n1.For Main\_Menu\n")

e=int(input())

if(e==1):

main\_menu()

else:

exit()

else:

for s,moves2 in enumerate(high\_score1):

print(s,moves2)

print("\n")

print("\n1.For Main\_Menu\n")

c1=int(input())

if(c1==1):

main\_menu()

else:

exit()

def Hard\_Leaderboard():

global moves3

print("\n\n\*\*\*LEADERBOARD\*\*\*\*\n\n")

high\_score2=[]

with open("high\_score2.txt") as r:

for line in r:

s,moves3=line.split(',')

moves3=int(moves3)

high\_score2.append((s,moves3))

high\_score2.sort(key=lambda z:z[1])

if not high\_score2:

print("No Recorded Score\n")

print("\n1.For Main\_Menu\n")

g=int(input())

if(g==1):

main\_menu()

else:

exit()

else:

for s,moves3 in high\_score2:

print(s,moves3)

print("\n")

print("\n1.For Main\_Menu\n")

c1=int(input())

if(c2==1):

main\_menu()

else:

exit()

def main\_menu():

print("\t^0^ \* ^0^ \* --MENU-- \* ^0^ \* ^0^ \*\n");

print("\t\t1) Play\n");

print("\t\t2) View High Scores\n");

print("\t\t3) Instructions\n");

print("\t\t4) Exit");

print("\n\t\tSelect your option: ")

opt=int(input());

if(opt==1):

play();

elif(opt==2):

print("Leaderboard for ?\n")

print("1. for Easier level\n")

print("2. for Medium level\n")

print("3. for Harder level\n")

print("Enter 4 for main\_menu\n")

print("Enter 5 to exit\n")

print("Enter option:\n")

optl=int(input())

if(optl==1):

Easy\_Leaderboard();

elif(optl==2):

Med\_Leaderboard();

elif(optl==3):

Hard\_Leaderboard();

elif(optl==4):

main\_menu();

elif(optl==5):

exit()

else:

print("Enter valid option:");

elif(opt==3):

instructions()

time.sleep(5)

print("\n\nDo you want to play the game?(y/n)\n")

ch=input()

if(ch=='y'):

play()

elif(ch=='n'):

exit()

elif(opt==4):

exit()

main\_menu()

**c. Testing**

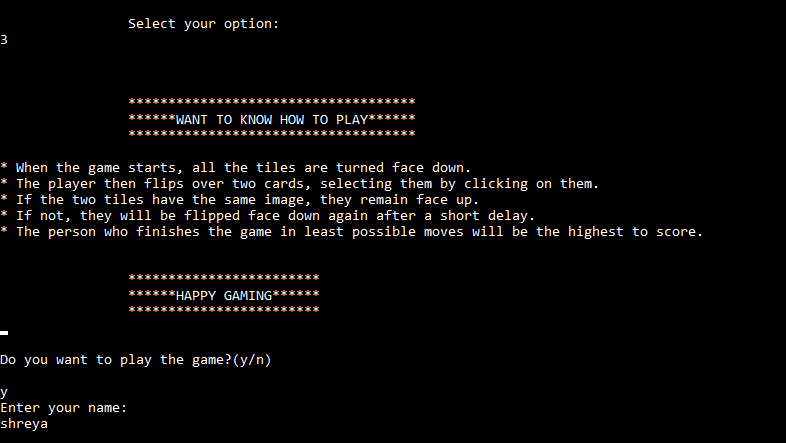
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | | | |
| **Test Case Id:** TC01 | | **Use Case ID:** UC01 | |
| **Test Case Title:** Levels | |  | |
| **Test Case Description:**  Verify whether different levels are being accessible and playable | |
| **Test Steps** | **Expected Result** | | **Actual Result** |
| 1.open the game using any editor and run.  2.Select play from the main menu  3.Select the level user wants to play. | The user gets access to the levels if he follows the procedure mentioned and gets an GUI interface of game. | | The user successfully accessed the levels(pass). |

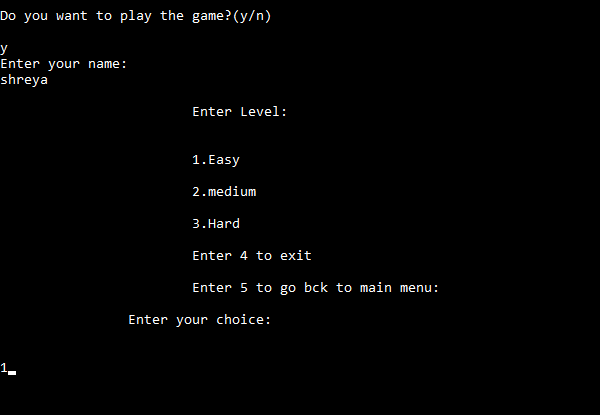
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | | | |
| **Test Case Id:** TC02 | | **Use Case ID:** UC02 | |
| **Test Case Title:** Leaderboard | |  | |
| **Test Case Description:**  Verify whether user can access Leaderboard and high  scores are shown | |
| **Test Steps** | **Expected Result** | | **Actual Result** |
| 1.open the game using any editor and run.  2.Select Leaderboard from the main menu  3.Select the required leaderboard user wants to view. | The user gets access to the leaderboard if he follows the procedure mentioned and gets an interface displaying results in that particular domain of level. | | The user successfully accessed the leaderboard and system displayed the high scores(pass). |

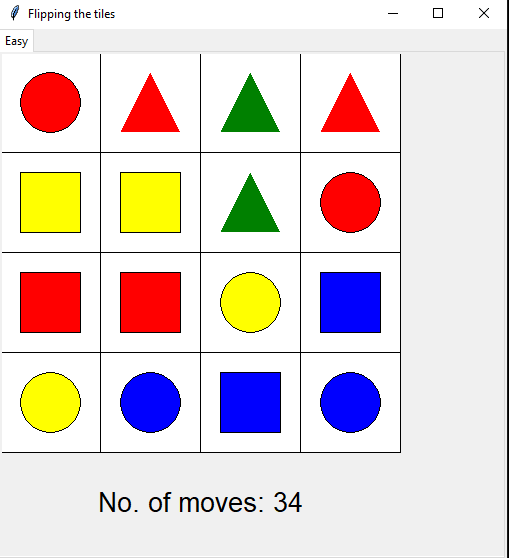
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | | | |
| **Test Case Id:** TC03 | | **Use Case ID:** UC03 | |
| **Test Case Title:** Instructions | |  | |
| **Test Case Description:**  Verify whether Instructions are being displayed when the user selects for it. | |
| **Test Steps** | **Expected Result** | | **Actual Result** |
| 1.open the game using any editor and run.  2.Select Instructions from the main menu. | The system displays the instructions and asks whether he wants to continue to play or not | | The user successfully accessed the instructions (pass). |

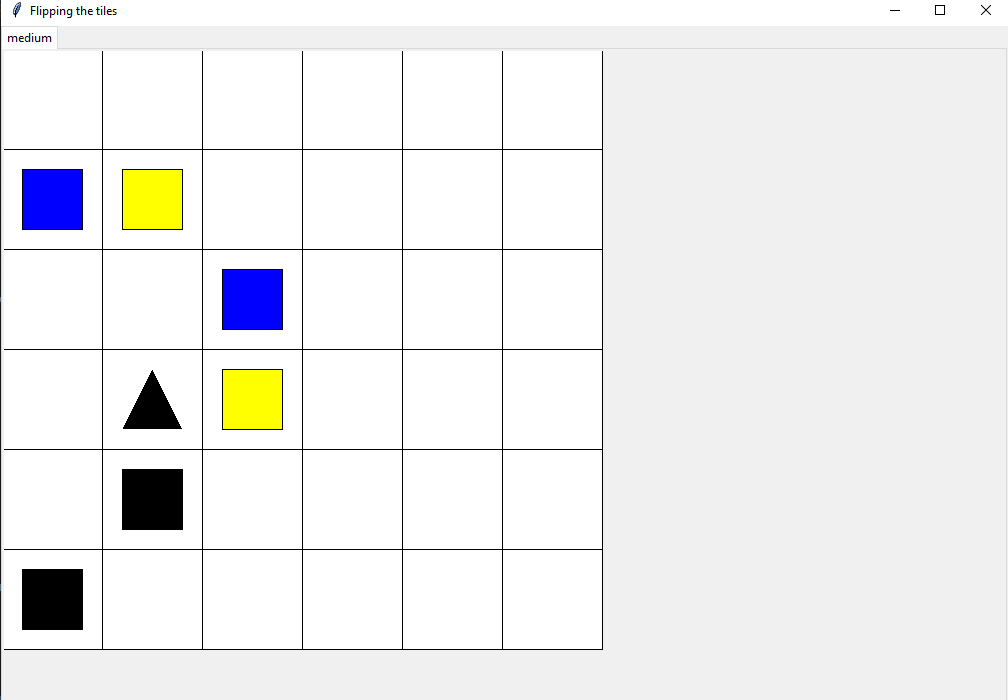
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | | | |
| **Test Case Id:** TC04 | | **Use Case ID:** UC04 | |
| **Test Case Title:** Exit | |  | |
| **Test Case Description:**  Verify whether game is being terminated or not when user opt for exit | |
| **Test Steps** | **Expected Result** | | **Actual Result** |
| 1.open the game using any editor and run.  2.Select exit from the main menu | The game terminates. | | The system successfully terminated the game(pass). |

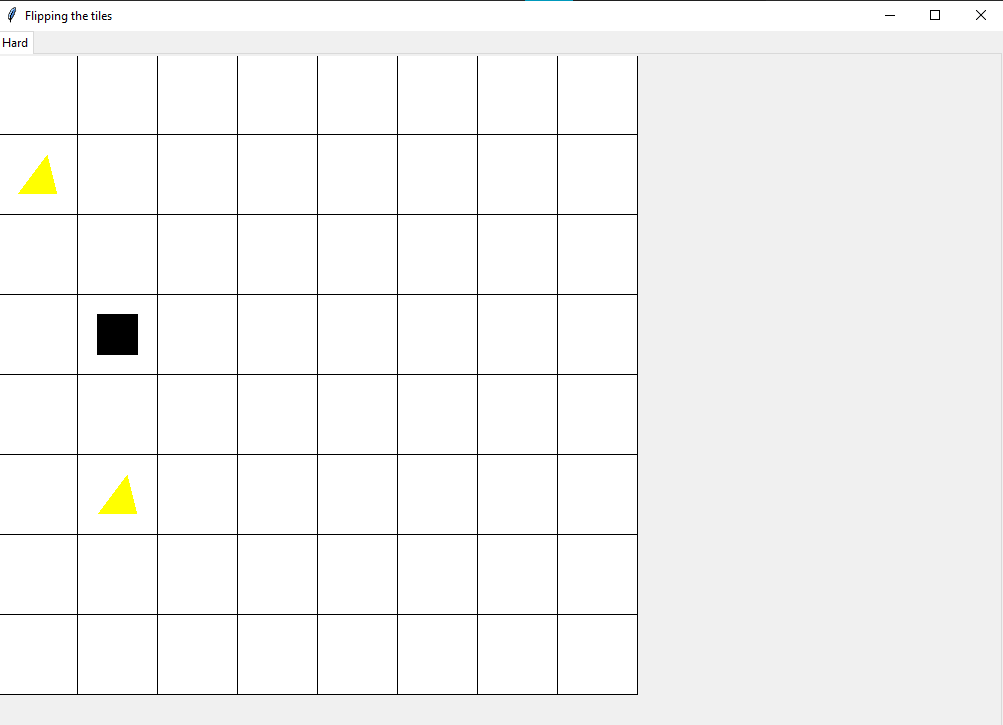
**RESULTS**

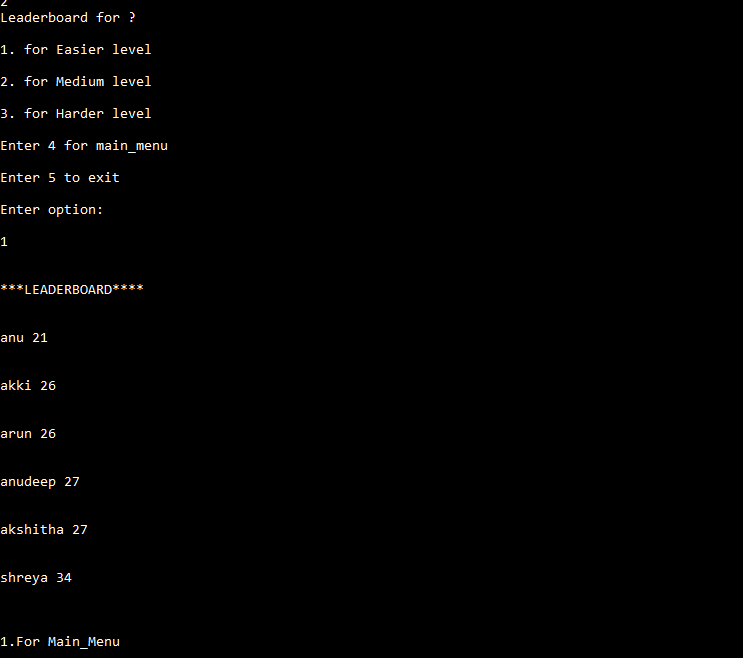
****

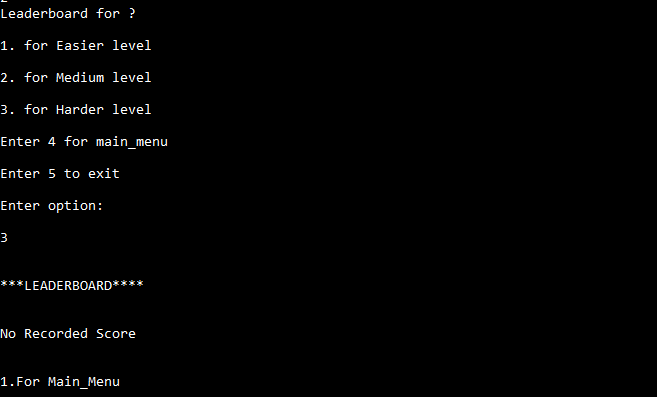
****

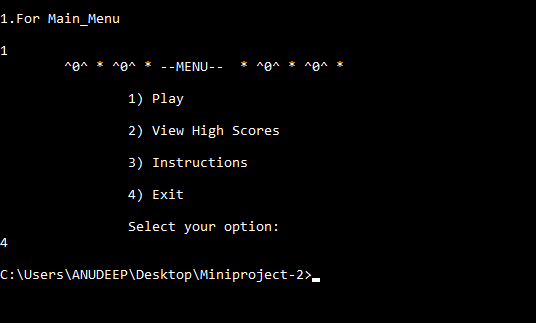
****

****

****

****

****

****

**Outcomes from the project apart from curriculum**

● I have got to know how to use different kinds of modules.

●I have discovered different kinds of libraries and modules and got to know their implementation and working.

●I have got to know about pygame, turtle most importantly tkinter.

●I have learnt how to build GUI using tkinter module.

●I have got to know how to handle events in GUI and about the components in them.

**CONCLUSION AND FUTURE WORK:**

As we know Gaming is one of the largest segments of the entertainment industry. It is a demanding career in today's modern world.

With the wide access to the internet has created a humongous demand for online games and with the advent of technology, this industry is growing at fast pace.

So, this kind of memory games still have a scope in future world.

**Future work**

● Level Extension

● Improve Graphical Representation

● Introduce new game features

● Take user responses through website and produce web rank list.

The project is successfully implemented with all the features and steps of Flipping the tiles as per the requirements.

**11. REFERENCES**

● Allen Downey,” Think Python: How to Think Like a Computer Scientist”,

O’Reilly publications,2nd Edition.

● nptel courses: https://nptel.ac.in/noc/courses/noc22/SEM1/noc22

● YouTube videos: https://youtu.be/JBnT0Zr4WNA