Guru Nanak Dev Engineering College, Ludhiana

Department of Computer Science and Engineering

Project Report
of
Database Management System
Laboratory

(LPCCS-107)

Submitted To:

Hardeep Singh Kang

Submitted By:

Amanjot Singh - 2004691
Priyanka Jhamb - 1905379
Vrishti Gupta - 1905064

Contents

1 Tic Tac Toe Game **1**
1.1 Introduction 1
1.2 Tech Stack 1
1.2.1 SQL 1
1.2.2 Python 1
1.2.3 Tkinter 1

2 Project **2**
2.1 Project Flow 2
2.2 Program 2
2.3 SQL 17
2.3.1 Table 17
2.3.2 Data 17
2.4 Output 17

3 Conclusion **20**

References **21**

Chapter 1

Tic Tac Toe Game

1.1 Introduction

"Tic Tac Toe" is a AI Single-Player game played by a Player and computer on a 3x3 board. The computer will choose a random position on the board to place its mark. The Player can then place their mark on the board. The Player can then choose to play again or exit the game. The Player can also see his/her score, wins, loses.

1.2 Tech Stack

1.2.1 SQL

SQL is a database computer language designed for the retrieval and management of data in a relational database. SQL stands for Structured Query Language. The uses of SQL include modifying database table and index structures; adding, updating and deleting rows of data

1.2.2 Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.

1.2.3 Tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

Chapter 2

Project

2.1 Project Flow

1. Importing Required Libraries.
   - os
   - random
   - sqlite3
   - time
   - tkinter
   - deepcopy
   - partial
   - PIL
   - font
   - pyglet
2. Splash Page will appear.
3. First Screen having asking Player's name and Option of Play as well as Link to the Developers Page will show.
4. Second Screen having board of game.
5. Then Player will play.
6. Result Screen will be displayed in which game result, option of play game again and exit option.

2.2 Program

```
import os  # os is a module that allows us to interact with the operating
    system
import random
import sqlite3  # sqlite3 is a module that allows us to connect to a
    database
import time
```

```
import tkinter as tk  # tkiner is a module that allows us to create GUI's
    in python
import tkinter.ttk as ttk  # ttk is the themed tkinter
from copy import deepcopy
from functools import partial
from tkinter import *  # * is used to import all the functions from
    tkinter module
from PIL import Image, ImageTk  # PIL is the Python Imaging Library
from tkinter import font
import pyglet  # pyglet is a module that allows us to play music in python

dir_path = os.path.dirname(os.path.realpath(__file__)) # dir_path is the
    path of the current directory
pyglet.font.add_file(os.path.join(dir_path, 'files', 'pacifico-font', '
    pacifico-v17-latin-regular.ttf')) # add font to pyglet
pyglet.font.add_file(os.path.join(dir_path, 'files', 'merriweather-font',
    'merriweather-v28-latin-regular.ttf')) # add font to pyglet
# from datetime import date, time, datetime  # datetime is a module that
    allows us to get the current date and time
# import time as t  # t is a module that allows us to get the current time


class TicTacToe:
    def __init__(self, window):
        # ====================================Main(root)
    ==================================
        self.style = ttk.Style()
        self.Page_num = 0
        self.window = window
        self.window.geometry("400x400")
        # self.main.geometry(str(self.screen_width) + "x" + str(self.
    screen_height))
        # self.window.state("zoomed")
        self.window.title("Tic Tac Toe")
        # root.tk.call('wm', 'iconphoto', root._w, PhotoImage(file='
    project_icon.jpg'))
        self.window.config(bg="white")
        self.window.resizable(False, False)
        # ====================================Main(root)
    ==================================

        # ====================================Variable_initialize
    ==================================
        self.dir_path = os.path.dirname(os.path.realpath(__file__))
        self.name = StringVar()
        self.splash_page = None
        self.first_page = None
        self.filemenu = None
        self.first_page_text = None
        self.first_page_textbox = None
        self.first_page_textbox_label = None
        self.first_page_button = None
        self.first_page_validation_text = None
        self.check_input_registry = None
```

```
        self.second_page = None
        self.button = None
        self.button_border = None
        self.board = [[" " for _ in range(3)] for _ in range(3)]
        self.possiblemove = None
        self.boardcopy = None
        self.move = None
        self.edge = None
        self.corner = None
        self.pcmove = None
        self.end_page = None
        self.end_page_text = None
        self.end_page_exit_button = None
        self.end_page_play_again_button = None
        self.leader_board = None
        self.leader_board_total_games_played_value = None
        self.leader_board_total_games_played = None
        self.sign = 0
        self.wins = None
        self.losses = None
        self.ties = None
        # ====================================Variable_initialize
==================================

        # ====================================Database
==================================
        self.db = sqlite3.connect(os.path.join(self.dir_path, 'files', '
TicTacToe.db'))
        self.cursor = self.db.cursor()
        self.cursor.execute("CREATE TABLE IF NOT EXISTS player_info(Name
TEXT, Score INTEGER, Loss INTEGER, Tie INTEGER)")
        self.db.commit()
        # ====================================Database
==================================

    def retrieveData(self):
        self.cursor.execute("SELECT * FROM player_info WHERE Name = '%s'"
% self.name.get())
        for row in self.cursor.fetchall():
            # print(row)
            self.wins = row[1]
            self.losses = row[2]
            self.ties = row[3]

    def Splash_screen(self):
        # self.style.configure("TButton", font=('Arial', 10, 'bold'),
foreground="black", background="white")
        self.splash_page = Frame(self.window, bg="black")
        self.splash_page.pack()  # pack() is used to show the frame
        img = Image.open(os.path.join(self.dir_path, 'files', "Splash_page
.jpg"))  # open the image
        photo = ImageTk.PhotoImage(img)  # convert the image to a
PhotoImage
        imga = Label(self.splash_page, image=photo, bd=0)  # create a
```

```
label and set the image as its background and bd=0 means no border and
padx and pady are used to set the padding
        imga.image = photo  # keep a reference to the image to prevent
garbage collection
        imga.pack()
        self.window.after(3000, self.First_screen)  # after() is used to
delay the execution of the function for the specified time

    def First_screen(self):
        self.splash_page.destroy()
        self.first_page = Frame(self.window, bg="white")
        self.first_page.pack(fill=BOTH, expand=True)
        # menubar = Menu(self.window)
        # self.filemenu = Menu(menubar, tearoff=0)  # tearoff=0 means that
    the menu will not be displayed as a separate window
        # menubar.add_cascade(label="File", menu=self.filemenu) #
add_cascade() is used to add a new menu to the menu bar
        # # self.filemenu.add_command(label="New Game", command=self.
New_game)
        # self.filemenu.add_command(label="Exit", command=exit)
        # self.window.config(menu=menubar)
        # ====================================First(root)
=================================
        self.first_page_text = Label(self.first_page, text="Welcome to Tic
    Tac Toe", font=("Pacifico", 23), bg="white", fg="black")
        self.first_page_text.grid(row=0, column=0, columnspan=4)

        self.first_page_textbox_label = Label(self.first_page, text="Enter
    Your Name", font=("Arial", 14), bg="white", fg="black")
        self.first_page_textbox_label.grid(row=1, column=0, padx=(5, 2),
pady=(15, 5))

        self.check_input_registry = self.window.register(self.check_input)
        self.first_page_textbox = ttk.Entry(self.first_page, style="Name.
TEntry", font=("Merriweather", 13), textvariable=self.name)
        self.first_page_textbox.config(validate="key", validatecommand=(
self.check_input_registry, '%P'))
        self.style.configure("Name.TEntry", foreground="black", background
="#8c66ff", selectbackground="#007fff", selectforeground="white")
        self.first_page_textbox.focus()
        self.first_page_textbox.grid(row=1, column=1, columnspan=1, padx
=(1, 15), pady=(22, 10))

        self.first_page_validation_text = Label(self.first_page, text="",
font=("Century", 11), bg="white", fg="red")
        self.first_page_validation_text.grid(row=2, column=0, columnspan
=4, pady=0)

        self.first_page_button = ttk.Button(self.first_page, text="Play",
style="play.Accent.TButton", cursor="arrow", state=DISABLED, command=
self.Validate_Name)
        self.style.configure("play.Accent.TButton", font=("Arial", 15, "
bold"), foreground="white", background="#8c66ff")
        self.first_page_button.grid(row=3, column=0, columnspan=2, padx=5,
```

```
 pady=10)


    # self.DevelopersButton = ttk.Button(self.first_page, text="
Developers", style="Developers.Accent.TButton", cursor="arrow", command
=self.Developers_Screen)
    # self.style.configure("Developers.Accent.TButton", font=("Arial",
 15, "bold"), foreground="white",background="#8c66ff", bd=0, width=33)
    # self.DevelopersButton.grid(row=4, column=0, columnspan=2, padx
=5, pady=133)
    self.DevelopersLink = Label(self.first_page, text="Developers",
font=("Arial", 12), cursor="hand2", bd=0, fg="blue", bg="white")
    # self.style.configure("Developers.Accent.TButton" foreground="
white",background="#8c66ff", bd=0, width=33)
    self.DevelopersLink.grid(row=4, column=0, columnspan=2, pady=14,
sticky=tk.S)
    self.first_page.rowconfigure(4, weight=1)
    CreateToolTip(self.DevelopersLink, text="Click to see Developers",
 h_side="right", v_side="bottom", padx=-10 , pady=30, underline=True)
    self.DevelopersLink.bind("<Button-1>", lambda *_: self.
Developers_Screen())
    # self.DevelopersLink.bind("<Enter>", lambda *_: self.
DevelopersLink.config(underline=True))
    # self.DevelopersLink.bind("<Leave>", lambda *_: self.
DevelopersLink.config(underline=False))
    # self.first_page_button = RoundedButton(self.first_page, textsize
=18, text="Play", radius=20, btnbackground="#0078ff", btnforeground="#
ffffff", clicked=self.Second_screen)
    # self.first_page_button.grid(row=2, column=0, columnspan=4, pady
=5)
    # text box
    # Select 0 or X

    # self.window.after(3000, self.Second_screen)  # Change this code
to navigate to second screen.

 def check_input(self, name_input):
    if len(str(name_input)) >= 1:
        if len(str(name_input)) > 15:
            self.first_page_validation_text.config(text="Name should
be less than 15 characters!!!")
            self.first_page_button.config(state=DISABLED, cursor="
arrow")
            return False
        else:
            self.first_page_validation_text.config(text="")
            self.first_page_button.config(state=NORMAL, cursor="hand2"
)
            return True
    else:
        self.first_page_validation_text.config(text="Required: Please
Enter Your Name!!!")
        self.first_page_button.config(state=DISABLED, cursor="arrow")
        return True
```

```
 def Validate_Name(self):
     if self.name.get() == "":
         self.first_page_validation_text.config(text="Required: Please
Enter Your Name!!!", cursor="arrow")
     else:
         self.first_page_validation_text.config(text="", cursor="hand2"
)
         self.first_page.destroy()
         # self.cursor.execute("INSERT INTO player_info(name) VALUES(?)
", (self.name.get(),))
         # self.cursor.execute("IF EXISTS (SELECT * FROM player_info
WHERE Name=?) BEGIN INSERT INTO player_info(Name) VALUES(?) END", (self
.name.get(),self.name.get()))
         # self.cursor.execute("INSERT IGNORE INTO player_info(Name,
Score) VALUES(?,?)", (self.name.get(),'0'))
         self.name.set(self.name.get().lower())
         self.cursor.execute("INSERT INTO player_info (Name,Score,Loss,
Tie) SELECT '" + self.name.get() + "','0','0','0' WHERE NOT EXISTS (
SELECT Name FROM player_info WHERE Name='" + self.name.get() + "')")
         self.db.commit()
         self.Second_screen()

 def Second_screen(self):
     self.second_page = Frame(self.window, bg="white")
     self.second_page.pack()

     # self.Player_1_name = Label(self.second_page, text="Player : X",
font=("Merriweather", 13), bg="#f7f7f7", fg="black", bd=3,pady=5,
relief=GROOVE, width=10, anchor=CENTER, justify=CENTER,
highlightthickness=4, highlightcolor="black")
     # self.Player_1_name.grid(row=0, column=0, padx=(5, 2), pady=(15,
5))

     # self.Player = ttk.Label(self.second_page, text="Player : X",
style="Player.TLabel", font=("Merriweather", 13), background="#f7f7f7",
 foreground="black",padding=(10,5,10,5) , relief=GROOVE, anchor=CENTER,
 justify=CENTER)
     # self.Player.grid(row=0, column=0,columnspan=2 , padx = (0,50))
     # self.Computer = ttk.Label(self.second_page, text="Computer : O",
 style="Player.TLabel", font=("Merriweather", 13), background="#f7f7f7
", foreground="black",padding=(10,5,10,5) , relief=GROOVE, anchor=
CENTER, justify=CENTER)
     # self.Computer.grid(row=0, column=1, columnspan=2, pady=18, padx
=(18,0))
     self.Player = ttk.Label(self.second_page, text="Player : X", style
="Player.TLabel", font=("Merriweather", 12), background="#f7f7f7",
foreground="black",padding=(10,4,10,5) , relief=GROOVE, anchor=CENTER,
justify=CENTER)
     self.Player.grid(row=0, column=0,columnspan=2 , pady=(8,18), padx
= (0,50))
     self.Computer = ttk.Label(self.second_page, text="Computer : O",
style="Player.TLabel", font=("Merriweather", 12), background="#f7f7f7",
```

```
 foreground="black",padding=(10,4,10,5) , relief=GROOVE, anchor=CENTER,
 justify=CENTER)
      self.Computer.grid(row=0, column=1, columnspan=2, pady=(8,18),
padx=(18,0))

      # l1 = Button(self.second_page, text="Player : X", width=10, state
=DISABLED)
      # l1.grid(row=0, column=2, pady=18)
      # # l1.grid(row=0, column=1, padx = 0)
      # l2 = Button(self.second_page, text="Computer : O", width=12,
state=DISABLED)
      # l2.grid(row=0, column=3, columnspan=2, pady=18)

      self.button = []
      self.button_border = []
      for i in range(3):
          m = 1 + i
          self.button.append(i)
          self.button_border.append(i)
          self.button[i] = []
          self.button_border[i] = []
          for j in range(3):
              n = j
              self.button[i].append(j)
              self.button_border[i].append(j)
              get_t = partial(self.get_text_pc, i, j, self.second_page,
self.Player, self.Computer)
              self.button_border[i][j] = tk.Frame(self.second_page,
highlightbackground="black", highlightthickness=1, bd=0)
              self.button[i][j] = Button(self.button_border[i][j], bd=0,
 command=get_t, height=4, width=9, relief="flat", font=("Arial", 14,"
bold"), cursor="hand2", background="#f8f8f8",disabledforeground="black"
,activebackground="#f8f8f8", activeforeground="black")
              self.button[i][j].grid(row=m, column=n)
              self.button_border[i][j].grid(row=m, column=n)
      self.second_page.mainloop()


 def Result_screen(self, message):
      self.second_page.destroy()
      self.end_page = Frame(self.window, bg="white")
      self.end_page.pack()

      self.end_page_text = Label(self.end_page, text=message, font=("
Pacifico", 24), bg="white", fg="black")
      self.end_page_text.grid(row=0, column=0, columnspan=4, pady=(2,
10))

      self.end_page_exit_button = ttk.Button(self.end_page, text="Exit",
 style="exitbtn.Accent.TButton", cursor="hand2", command=self.window.
destroy)
      self.style.configure("exitbtn.Accent.TButton", font=("Arial", 13,
"bold"), foreground="white", background="#8c66ff")
      self.end_page_exit_button.grid(row=1, column=1, padx=10, pady
```

```
=(6,12))

        self.end_page_play_again_button = ttk.Button(self.end_page, text="
Play Again", style="playagain.Accent.TButton", cursor="hand2", command=
self.Handle_Play_Again)
        self.style.configure("playagain.Accent.TButton", font=("Arial",
13, "bold"), foreground="white", background="#8c66ff")
        self.end_page_play_again_button.grid(row=1, column=2, padx=10,
pady=(6,12))

        self.separator = ttk.Separator(self.end_page, orient=HORIZONTAL,
style="Separator.TSeparator")
        self.separator.grid(row=2, column=0, columnspan=4,padx=(40, 40),
pady=(10,0), sticky="ew")

        self.retrieveData()

        # ----------------------------------------------------------Player's
Scoreboard----------------------------------------------------------
        self.leader_board = Label(self.end_page, text=self.name.get().
capitalize() + "'s Scoreboard", font=("Pacifico", 20), bg="white", fg="
black")
        self.leader_board.grid(row=3, column=0, columnspan=4, padx=10,
pady=(2, 8))
        # ----------------------------------------------------------Total
Games Played row----------------------------------------------------------
        self.leader_board_total_games_played = Label(self.end_page, text="
Total Games Played", font=("Arial", 13), bg="white", fg="black")
        self.leader_board_total_games_played.grid(row=4, column=1,
columnspan=1, padx=10, pady=10)
        self.leader_board_total_games_played_value = Label(self.end_page,
text=(self.wins + self.losses + self.ties), font=("Arial", 13), bg="
white", fg="grey")
        self.leader_board_total_games_played_value.grid(row=4, column=2,
columnspan=1, padx=10, pady=10)
        # ----------------------------------------------------------Total Wins
 row----------------------------------------------------------
        self.leader_board_total_games_played = Label(self.end_page, text="
Wins", font=("Arial", 13), bg="white", fg="black")
        self.leader_board_total_games_played.grid(row=5, column=1,
columnspan=1, padx=10, pady=10)
        self.leader_board_total_games_played_value = Label(self.end_page,
text=self.wins, font=("Arial", 13), bg="white", fg="grey")
        self.leader_board_total_games_played_value.grid(row=5, column=2,
columnspan=1, padx=10, pady=10)
        # ----------------------------------------------------------Total
Losses row----------------------------------------------------------
        self.leader_board_total_games_played = Label(self.end_page, text="
Losses", font=("Arial", 13), bg="white", fg="black")
        self.leader_board_total_games_played.grid(row=6, column=1,
columnspan=1, padx=10, pady=10)
        self.leader_board_total_games_played_value = Label(self.end_page,
text=self.losses, font=("Arial", 13), bg="white", fg="grey")
        self.leader_board_total_games_played_value.grid(row=6, column=2,
```

```
columnspan=1, padx=10, pady=10)

    # ----------------------------------------------------------Total Ties
 row----------------------------------------------------
    self.leader_board_total_games_played = Label(self.end_page, text="
Ties", font=("Arial", 13), bg="white", fg="black")
    self.leader_board_total_games_played.grid(row=7, column=1,
columnspan=1, padx=10, pady=10)
    self.leader_board_total_games_played_value = Label(self.end_page,
text=self.ties, font=("Arial", 13), bg="white", fg="grey")
    self.leader_board_total_games_played_value.grid(row=7, column=2,
columnspan=1, padx=10, pady=10)


 def Developers_Screen(self):
    self.first_page.destroy()
    self.Developers_page = Frame(self.window, bg="white")
    self.Developers_page.pack(fill=BOTH, expand=True)
    self.Developers_page.columnconfigure(0, weight=1)
    self.Developers_page.columnconfigure(1, weight=1)

    back_btn_img = ImageTk.PhotoImage(Image.open(os.path.join(self.
dir_path, 'files', "back.png")))
    self.Back_btn = Button(self.Developers_page, image=back_btn_img,
bd=0, highlightthickness=0, height=36, width=36, relief=FLAT,
background="#ffffff", bg="#ffffff", activebackground="#ffffff", command
=self.Handle_Developers_Page_exit,cursor="hand2")
    self.Back_btn.image = back_btn_img
    self.Back_btn.grid(padx=(10,0), pady=(10,0), sticky=tk.N+tk.W)
    CreateToolTip(self.Back_btn, text="Go Back", h_side="right",
v_side="bottom", padx=-8 , pady=40)

    self.Developers = Label(self.Developers_page, text="Developers",
font=("Pacifico", 23), bg="white", fg="black")
    self.Developers.grid(row=0, column=0, columnspan=4, pady=(10, 0))

    self.Developers_Batch = Label(self.Developers_page, text="Batch
(2019-2023)", font=("Arial", 11), bg="white", fg="black")
    self.Developers_Batch.grid(row=1, column=0, columnspan=4, pady
=(0,5), padx=5)

    self.Developers_Name_AMANJOT = Label(self.Developers_page, text="
Amanjot Singh (B.Tech CSE)", font=("Merriweather", 13), bg="white", fg=
"black")
    self.Developers_Name_AMANJOT.grid(row=2, column=0, columnspan=4,
pady=(20,5), padx=5)
    self.Developers_Name_PRIYANKA = Label(self.Developers_page, text="
Priyanka Jhamb (B.Tech CSE)", font=("Merriweather", 13), bg="white", fg
="black")
    self.Developers_Name_PRIYANKA.grid(row=3, column=0, columnspan=4,
pady=5, padx=5)
    self.Developers_Name_VRISHTI = Label(self.Developers_page, text="
Vrishti Gupta (B.Tech CSE)", font=("Merriweather", 13), bg="white", fg=
"black")
```

```
        self.Developers_Name_VRISHTI.grid(row=4, column=0, columnspan=4,
pady=5, padx=5)
        # self.Developers_page_exit_button = ttk.Button(self.
Developers_page, text="Play Game", style="exitbtn.Accent.TButton",
        #                                         cursor="hand2", command=
self.Handle_Developers_Page_exit)
        # self.style.configure("exitbtn.Accent.TButton", font=("
Developers_pageArial", 13, "bold"), foreground="white",
        #                         background="#8c66ff")
        # self.Developers_page_exit_button.grid(row=7, column=0, padx=20,
pady=100)


    def Handle_Developers_Page_exit(self):
        self.Developers_page.destroy()
        self.First_screen()


    def Handle_Play_Again(self):
        self.end_page.destroy()
        self.Reset_Variables()
        self.Second_screen()


    def Reset_Variables(self):
        self.splash_page = None
        self.first_page = None
        self.filemenu = None
        self.first_page_text = None
        self.first_page_textbox = None
        self.first_page_textbox_label = None
        self.first_page_button = None
        self.first_page_validation_text = None
        self.check_input_registry = None
        self.second_page = None
        self.button = None
        self.button_border = None
        self.board = [[" " for _ in range(3)] for _ in range(3)]
        self.possiblemove = None
        self.boardcopy = None
        self.move = None
        self.edge = None
        self.corner = None
        self.pcmove = None
        self.end_page = None
        self.end_page_text = None
        self.end_page_exit_button = None
        self.end_page_play_again_button = None
        self.leader_board = None
        self.leader_board_total_games_played_value = None
        self.leader_board_total_games_played = None
        self.sign = 0
        self.wins = None
        self.losses = None
```

```
        self.ties = None


    # Decide winner
    @staticmethod
    def winner(b, a):
        return ((b[0][0] == a and b[0][1] == a and b[0][2] == a) or
                (b[1][0] == a and b[1][1] == a and b[1][2] == a) or
                (b[2][0] == a and b[2][1] == a and b[2][2] == a) or
                (b[0][0] == a and b[1][0] == a and b[2][0] == a) or
                (b[0][1] == a and b[1][1] == a and b[2][1] == a) or
                (b[0][2] == a and b[1][2] == a and b[2][2] == a) or
                (b[0][0] == a and b[1][1] == a and b[2][2] == a) or
                (b[0][2] == a and b[1][1] == a and b[2][0] == a))

    # Check the self.board is full or not
    def is_full(self):
        flag = True
        for i in self.board:
            if i.count(' ') > 0:
                flag = False
        return flag

    def wait(self):
        for i in range(3):
            for j in range(3):
                self.button[i][j].config(state=DISABLED, cursor="arrow")
        var = IntVar()
        self.window.after(700, var.set, 1)
        self.window.wait_variable(var)
        for i in range(3):
            for j in range(3):
                self.button[i][j].config(state=ACTIVE, cursor="hand2")

    # Decide the next move of system
    def pc(self):
        self.possiblemove = []
        for i in range(len(self.board)):
            for j in range(len(self.board[i])):
                if self.board[i][j] == ' ':
                    self.possiblemove.append([i, j])  # Append the
possible moves
        # move = []
        if not self.possiblemove:
            return
        else:
            for let in ['O', 'X']:
                for i in self.possiblemove:
                    self.boardcopy = deepcopy(self.board)  # copy the self
.board
                    self.boardcopy[i[0]][i[1]] = let  # make the move
                    if self.winner(self.boardcopy, let):  # check if the
move is a winning move
                        return i
```

```
            self.corner = []
            for i in self.possiblemove:
                if i in [[0, 0], [0, 2], [2, 0], [2, 2]]:  # check if the
move is in the corner
                    self.corner.append(i)
            if len(self.corner) > 0:
                self.pcmove = random.randint(0, len(self.corner) - 1)  #
choose a random corner
                return self.corner[self.pcmove]
            self.edge = []
            for i in self.possiblemove:
                if i in [[0, 1], [1, 0], [1, 2], [2, 1]]:
                    self.edge.append(i)
            if len(self.edge) > 0:
                self.pcmove = random.randint(0, len(self.edge) - 1)
                return self.edge[self.pcmove]

     def get_text_pc(self, i, j, gb, Player, Computer):  # function to get
the text of the pc
        if self.board[i][j] == ' ':
            if self.sign % 2 == 0:
                # Player.configure(borderwidth=2,foreground="#2b2b2b",
background="#f2f2f2")
                # Computer.configure(borderwidth=6,foreground="black",
background="#f7f7f7")
                # l1.config(state=DISABLED)
                # l2.config(state=ACTIVE)
                self.board[i][j] = "X"
                self.sign += 1
                self.button[i][j].config(text=self.board[i][j])

            else:
                self.wait()
                # Player.configure(borderwidth=6,foreground="black",
background="#f7f7f7")
                # Computer.configure(borderwidth=2,foreground="#2b2b2b",
background="#f2f2f2")
                self.button[i][j].config(state=ACTIVE)
                # l2.config(state=DISABLED)
                # l1.config(state=ACTIVE)
                self.board[i][j] = "O"
                self.sign += 1
                self.button[i][j].config(text=self.board[i][j])
        x = True
        if self.winner(self.board, "X"):
            gb.destroy()
            x = False
            self.cursor.execute("UPDATE player_info SET Score = Score + 1
WHERE name = '" + self.name.get() + "'")
            self.db.commit()
            self.Result_screen("You Won !!!")
            # box = tkMessageBox.showinfo("Winner", "Player won the match
")
            # exit() if box == "ok" else None
```

```
        elif self.winner(self.board, "O"):
            gb.destroy()
            x = False
            self.cursor.execute("UPDATE player_info SET Loss = Loss + 1
    WHERE name = '" + self.name.get() + "'")
            self.db.commit()
            obj.Result_screen("Computer Won !!!")
            # box = tkMessageBox.showinfo("Winner", "Computer won the
    match")
            # exit() if box == "ok" else None
        elif self.is_full():
            gb.destroy()
            x = False
            self.cursor.execute("UPDATE player_info SET Tie = Tie + 1
    WHERE name = '" + self.name.get() + "'")
            self.db.commit()
            self.Result_screen("Tie Game !!!")
            # box = tkMessageBox.showinfo("Tie Game", "Tie Game")
            # exit() if box == "ok" else None

        if x:
            if self.sign % 2 != 0:
                self.move = self.pc()
                self.button[self.move[0]][self.move[1]].config(state=
    DISABLED)
                self.get_text_pc(self.move[0], self.move[1], gb, Player,
    Computer)


    def __del__(self):
        self.db.close()



class CreateToolTip(object):
    """
    create a tooltip for a given widget
    """
    def __init__(self, widget, text='widget info', h_side="right", v_side=
    "bottom", padx=10 , pady=30, underline=False):
        self.waittime = 300     #miliseconds
        self.wraplength = 180   #pixels
        self.widget = widget
        self.text = text
        self.underline = underline
        self.h_side = h_side
        self.v_side = v_side
        self.padx = padx
        self.pady = pady
        self.widget.bind("<Enter>", self.enter)
        self.widget.bind("<Leave>", self.leave)
        self.widget.bind("<ButtonPress>", self.leave)
        self.id = None
        self.tw = None
```

```
    def enter(self, event=None):
        if self.underline:
            # self.widget.config(underline=True)
            f = font.Font(self.widget, self.widget.cget("font"))
            f.configure(underline=True)
            self.widget.configure(font=f)
        self.schedule()

    def leave(self, event=None):
        if self.underline:
            # self.widget.config(underline=True)
            f = font.Font(self.widget, self.widget.cget("font"))
            f.configure(underline=False)
            self.widget.configure(font=f)
        self.unschedule()
        self.hidetip()

    def schedule(self):
        self.unschedule()
        self.id = self.widget.after(self.waittime, self.showtip)

    def unschedule(self):
        id = self.id
        self.id = None
        if id:
            self.widget.after_cancel(id)

    def showtip(self, event=None):
        x = y = 0
        x, y, cx, cy = self.widget.bbox("insert")
        if self.h_side == "right":
            x += self.widget.winfo_rootx() + self.padx
        elif self.h_side == "left":
            x += self.widget.winfo_rootx() - self.padx
        if self.v_side == "top":
            y += self.widget.winfo_rooty() - self.pady
        elif self.v_side == "bottom":
            y += self.widget.winfo_rooty() + self.pady
        # x += self.widget.winfo_rootx() + 10
        # y += self.widget.winfo_rooty() + 30
        # creates a toplevel window
        self.tw = tk.Toplevel(self.widget)
        # Leaves only the label and removes the app window
        self.tw.wm_overrideredirect(True)
        self.tw.wm_geometry("+%d+%d" % (x, y))
        label = Label(self.tw, text=self.text, justify='center', anchor="
center", font=("Merriweather",9), background="#ffffff", bg="#ffffff",
activebackground="#ffffff", relief='solid', borderwidth=1 ,wraplength =
 self.wraplength)
        label.pack(ipadx=4,anchor=tk.CENTER)

    def hidetip(self):
        tw = self.tw
```

```
        self.tw= None
        if tw:
            tw.destroy()



if __name__ == "__main__":
    root = Tk()  # root is the main window of the application
    # try:root.wm_attributes('-type', 'splash') # splash screen
    # root.tk.call('tk', 'windowingsystem') # for mac
    # root.attributes('-type', 'dock') # make the window a dock
    # root.overrideredirect(True)  # remove the title bar

    # root.update_idletasks()
    # frm_width = root.winfo_rootx() - root.winfo_x()
    # win_width = root.winfo_width() + 2 * frm_width
    # titlebar_height = root.winfo_rooty() - root.winfo_y()
    # win_height = root.winfo_height() + titlebar_height + frm_width
    # x = root.winfo_screenwidth() // 2 - win_width // 2
    # y = root.winfo_screenheight() // 2 - win_height // 2
    # root.geometry('{}x{}+{}+{}'.format(root.winfo_width(), root.
   winfo_height(), x, y))
    # root.deiconify()

    obj = TicTacToe(root)  # obj is an object of the class TicTacToe
    obj.Splash_screen()
    style = ttk.Style(root)
    root.tk.call('source', os.path.join(dir_path, 'files', 'Theme', 'azure
   .tcl'))
    style.theme_use('azure-light')
    root.iconbitmap(os.path.join(dir_path, 'files', 'app_icon.ico'))
    root.mainloop()
    del obj  # del is a keyword that allows us to delete an object

# Player's Leaderboard
# Total games Played
# Wins
# Loses
# Ties
```

2.3 SQL

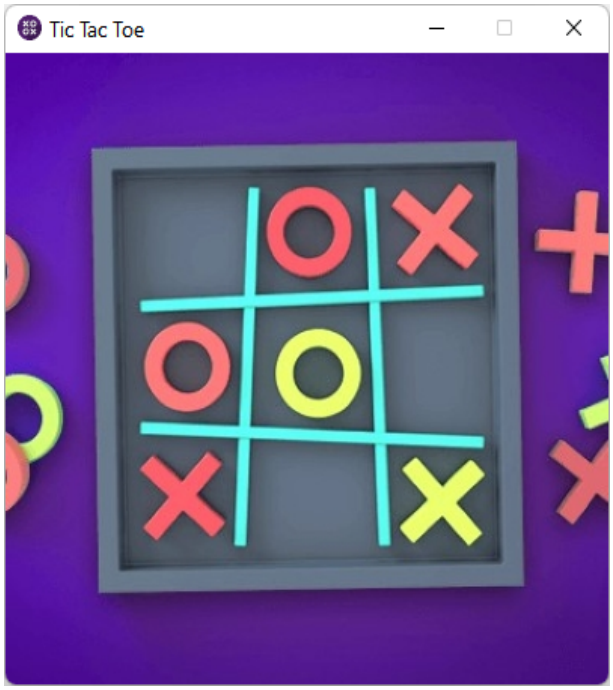
2.3.1 Table

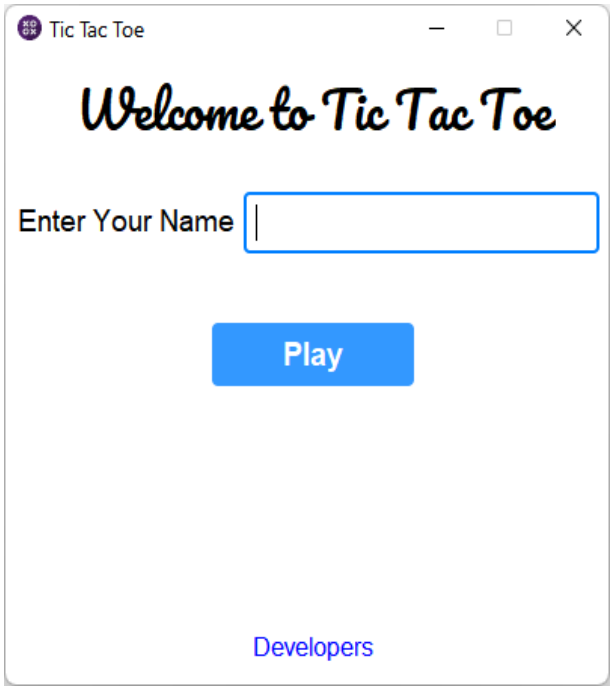
	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	Name	TEXT								NULL
2	Score	INTEGER								NULL
3	Loss	INTEGER								NULL
4	Tie	INTEGER								NULL

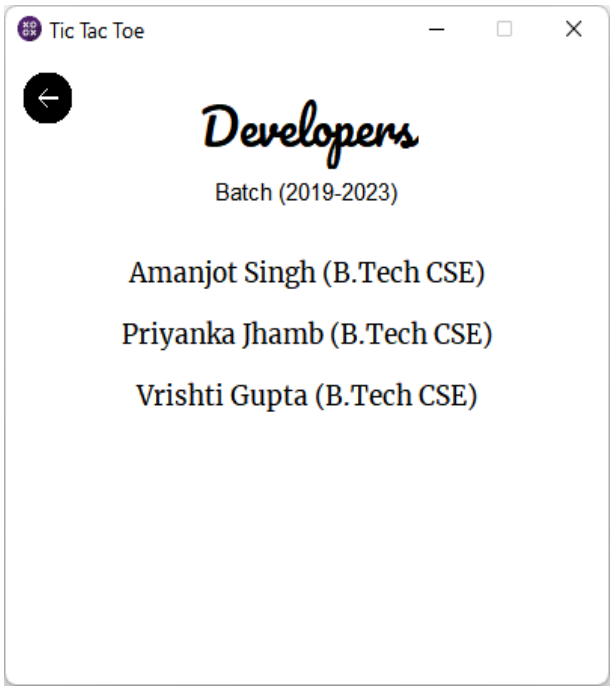
2.3.2 Data

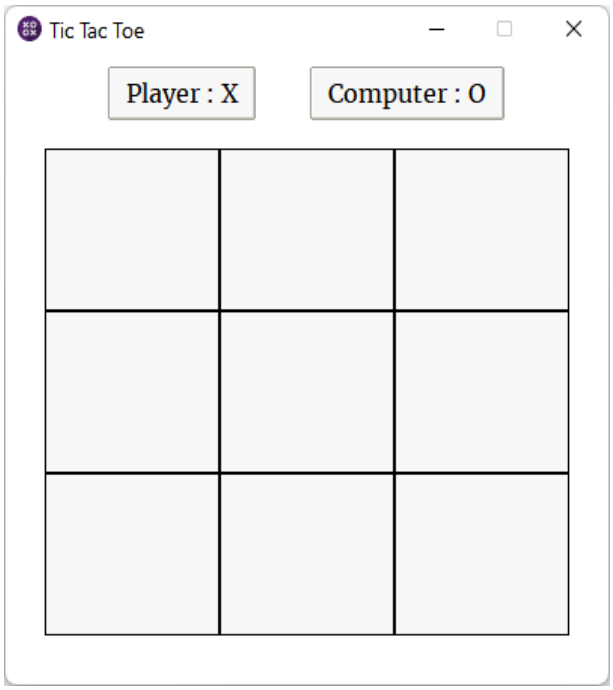
	Name	Score	Loss	Tie
1	amanjot	19	4	7
2	vrishti	14	0	4
3	priyanka	6	1	2

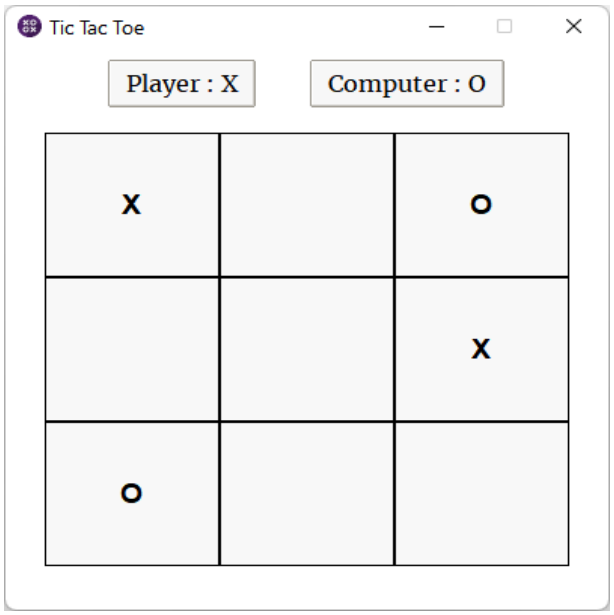
2.4 Output

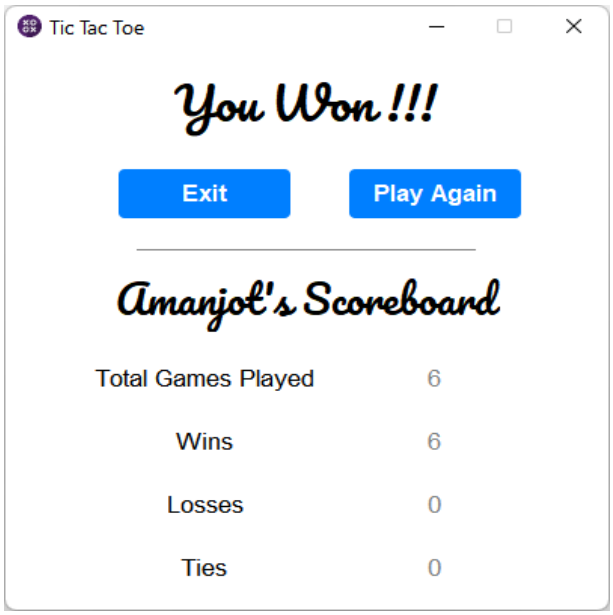


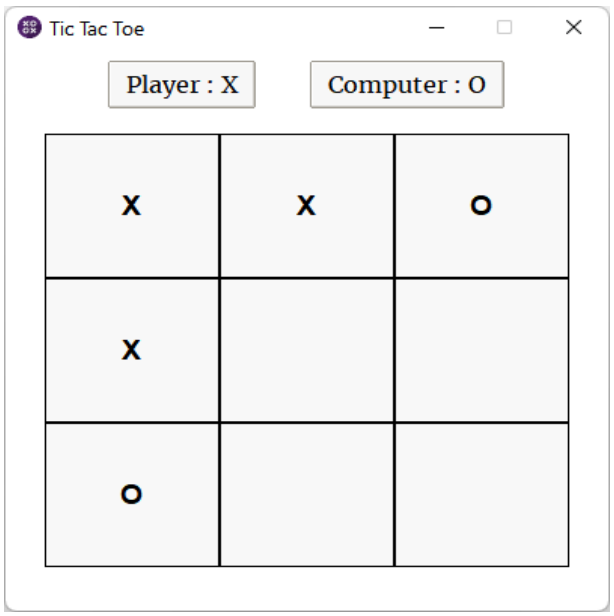


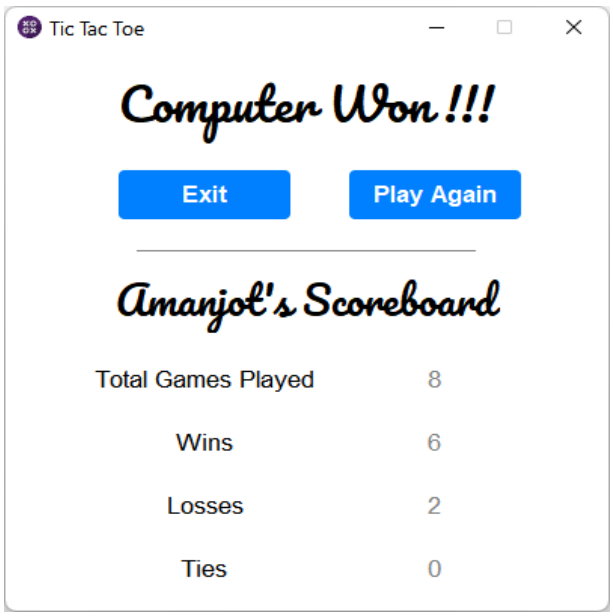

Tic Tac Toe
Developers
Batch (2019-2023)
Amanjot Singh (B.Tech CSE)
Priyanka Jhamb (B.Tech CSE)
Vrishti Gupta (B.Tech CSE)

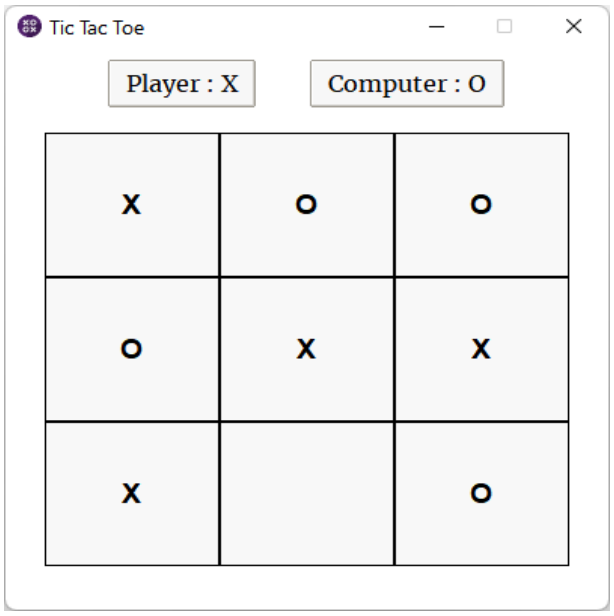

Tic Tac Toe
Player : X
Computer : O

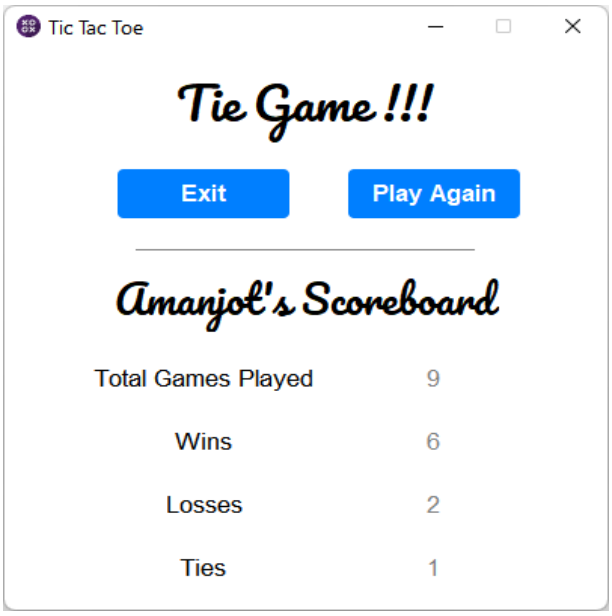

Tic Tac Toe
Player : X
Computer : O
X
O
X
O


Tic Tac Toe
You Won !!!
Exit
Play Again
Amanjot's Scoreboard
Total Games Played 6
Wins 6
Losses 0
Ties 0


Tic Tac Toe
Player : X
Computer : O
X
X
O
X
O


Tic Tac Toe
Computer Won !!!
Exit
Play Again
Amanjot's Scoreboard
Total Games Played 8
Wins 6
Losses 2
Ties 0


Tic Tac Toe
Player : X
Computer : O
X
O
O
O
X
X
X
O


Tic Tac Toe
Tie Game !!!
Exit
Play Again
Amanjot's Scoreboard
Total Games Played 9
Wins 6
Losses 2
Ties 1

Chapter 3

Conclusion

The Tic Tac Toe game is most familiar among all the age groups. Intelligence can be a property of any purpose-driven decision maker. This basic idea has been suggested many times.
An algorithm of playing Tic Tac Toe has been presented and tested which has used MINIMAX procedure, as in this, computer always tries to win or reduce the chances of player from winning. Here computer while placing move look ahead for the winning chances of itself and the player and accordingly it makes the move and this works in efficient way. Overall the system works without any bugs. Database also has been included to store the scores of the player and Leader board also will be represented to motivate the player.

References

[1] Python Documentation

[2] Tkinter Documentation

[3] Geeksforgeeks Site

[4] StackOverflow