LOVELY PROFESSIONAL UNIVERSITY

PROJECT REPORT

on

"TIC TAC TOE GAME WITH TIMER"

Submitted by: -

Devansh Agarwal (11908637) Ankit Gupta (11903499)

Under the Guidance of

Mr. Dipen Saini

School of Computer Science and Engineering Lovely Professional University, Phagwara

DECLARATION

We hereby declare that we have completed our project on **"TIC TAC TOE GAME WITH TIMER"** under the guidance of **"Mr. Dipen Saini"**. We have declared that we have worked with full dedication during these project periods and our learning outcomes partial fulfill the requirements for the award of degree of Bachelor of technology in Computer Science and Engineering during the period of 2020-21 in Lovely Professional University, Phagwara

Date: 1-NOV-2020

Devansh Agarwal (11908637)

Ankit Gupta (11903499)

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our Teacher, **MR. DIPEN SAINI** who gave us the golden opportunity to do this wonderful project of "**TIC TAC TOE GAME WITH TIMER**".

Who also helped us in completing our project. We all came to know about so many new things, we are really thankful to them.

Secondly, we would like to thanks our parents and friends who helped us a lot in finalizing this project within the limited time frame.

We acknowledge with thanks the kind of patronage, loving inspiration and timely guidance, which we have received from our Teacher **MR. DIPEN SAINI,** our family members and friends whose guidance, encouragement, suggestion and very constructive criticism have contributed immensely to the evolution of our ideas on this project.

We extremely grateful to our Departmental staff members, Lab technicians and Non-teaching staff members for their extreme help throughout our project.

Finally, we express our heartful thanks to all of our friends who helped us in successful completion of this project.

ABSTRACT

The purpose of our project “TIC TAK TOE WITH TIMER” is to computerize our very famous childhood game noughts and crosses which we used to play on paper with our friends in which two player are used to mark noughts and crosses on a space of 3x3 grid but in this computerized we are using a timer which will count from 30 seconds till 0 seconds.

INDEX

1. INTRODUCTION

1.1 INDROUCTION OF THE PROJECT: -

So, as we all know that "TIC TAC TOE" is the computerized version of our very famous childhood game noughts and crosses which we used to play on paper with our friends. In which two player are used to mark noughts and crosses on a space of 3x3 grid. But this computerized version is slightly different form that we used to play on paper because in this version we are using a timer which will count from 30 seconds up to 0 seconds. In this given period of 30 seconds one of the player has to try his chance and if he/she used to try in between of that period of time then the game will continue but if he/she don't used to try in that given period of time then it will show "GAME OVER" on the screen and the players need to restart the game from the starting using start button given on the screen of the player. And if all the players try their attempt in that given period of time of 30 seconds then it will show the name of player who has won the match or either it will show whether "X" or "O" won on the screen and if the game between the players gets draw then it will display "DRAW" on the computer screen. Name of the player has been filled on the top section given for the names of the players next to section "PLAYER1" and "PLAYER2". Players can quiet the game at any time when they want to quiet the game, it all depends on the player's decision.

1.2 PROBLEM INTRODUCTION: -

At the time of working of this project "TIC TAK TOE with TIMMER" we are not able to make timer and restart button properly. As, it just being displayed on the computer screen but it is not working when we click the restart button and count down is not getting started in the code, it is just showing the time section on the computer screen.

1.3 OBJECTIVES OF THE PROGRAM: -

The main objective of this project is to give players a challenge to take right decision while playing game in a limited period of time. So, that they can prepare them selves to play under a limited period of time and are able to take right decision in the game which will result in the wining of the game on their side.

1.4 MODULES: -

1. Name of both players
2. Time limit
3. Restart and quiet button to restart the game from starting and quieting the game when the player want to quiet.
4. Display "WINs!" or either display "TIE!" on the computer screen.

2. REQUIREMENT SPECIFICATION

2.1 INTRODUCTION: -

To be used efficiently, all computer software needs certain hardware components or the other software resources to be present on a computer. These pre-requisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

2.2 HARDWARE REQUIREMENTS: -

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

HARDWARE REQUIREMENTS FOR PRESENT PROJECT: -

PROCESSOR: - Intel dual Core, i3

RAM: - 1 GB

HARD DISK: - 80 GB

2.3 SOFTWARE REQUIREMENTS: -

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These

requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

SOFTWARE REQUIREMENTS FOR PRESENT PROJECT:-

OPERATING SYSTEM: - Windows 7/ XP/8/10

PROGRAMMING LANGUAGE: - Python

3.SOFTWARE SPECIFICATION

PYTHON: -

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

FEATURES OF PYTHON: -

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and crossplatform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.
- **Apart from the above-mentioned features, Python has a big list of good features, few are listed below −**
- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

4.SYSTEM IMPLEMENTATION

4.1 SAMPLE CODE: - from

```
tkinter   import   *   import

tkinter.messagebox

tk = Tk() tk.title("Tic

Tac Toe")


pa = StringVar()

playerb = StringVar()

p1 = StringVar() p2 =

StringVar()


player1_name = Entry(tk, textvariable=p1, bd=5)

player1_name.grid(row=1, column=1, columnspan=8) player2_name

= Entry(tk, textvariable=p2, bd=5) player2_name.grid(row=2,

column=1, columnspan=8)


bclick = True flag

= 0



def disableButton():

   button1.configure(state=DISABLED)

button2.configure(state=DISABLED)    button3.configure(state=DISABLED)

button4.configure(state=DISABLED)    button5.configure(state=DISABLED)
```

```
button6.configure(state=DISABLED)     button7.configure(state=DISABLED)
button8.configure(state=DISABLED)     button9.configure(state=DISABLED)



def btnClick(buttons):
   global bclick, flag, player2_name, player1_name, playerb, pa
if buttons["text"] == " " and bclick == True:
      buttons["text"] = "X"
bclick = False        playerb =
p2.get() + " Wins!"        pa =
p1.get() + " Wins!"
checkForWin()
      flag += 1



   elif buttons["text"] == " " and bclick == False:
      buttons["text"] = "O"
bclick = True
checkForWin()
      flag += 1
else:
      tkinter.messagebox.showinfo("Tic-Tac-Toe", "Button already Clicked!")


def checkForWin():
        if (button1['text'] == 'X' and button2['text'] == 'X' and button3['text'] == 'X' or
         button4['text'] == 'X' and button5['text'] == 'X' and button6['text'] == 'X' or
         button7['text'] =='X' and button8['text'] == 'X' and button9['text'] == 'X' or
         button1['text'] == 'X' and button5['text'] == 'X' and button9['text'] == 'X' or
```

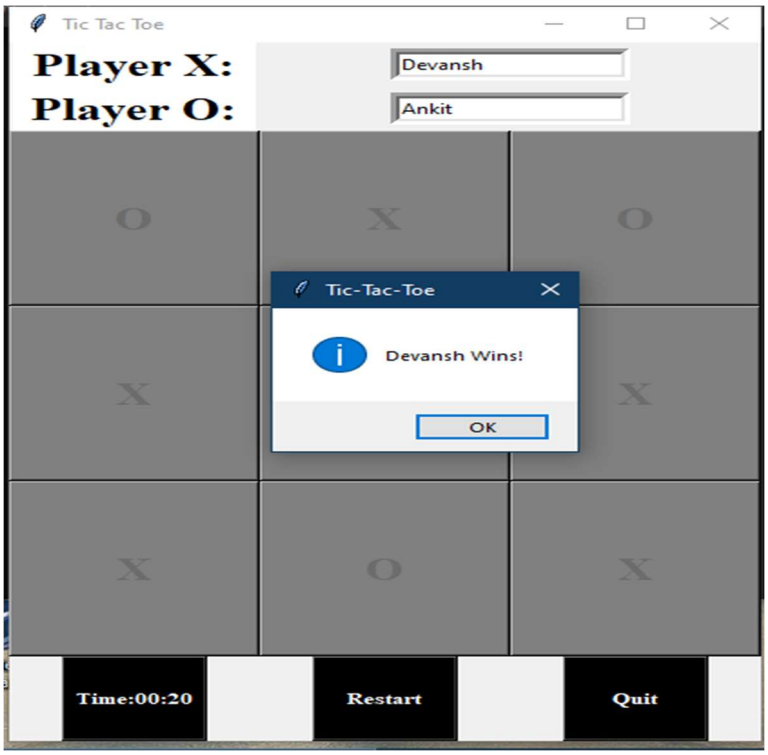
```
        button3['text'] == 'X' and button5['text'] == 'X' and button7['text'] == 'X' or
        button1['text'] == 'X' and button2['text'] == 'X' and button3['text'] == 'X' or
        button1['text'] == 'X' and button4['text'] == 'X' and button7['text'] == 'X' or
        button2['text'] == 'X' and button5['text'] == 'X' and button8['text'] == 'X' or
         button7['text'] == 'X' and button6['text'] == 'X' and button9['text'] == 'X'):
      disableButton()       tkinter.messagebox.showinfo("Tic-Tac-
Toe", pa)

   elif(flag == 8):
      tkinter.messagebox.showinfo("Tic-Tac-Toe", "It is a Tie")

   elif (button1['text'] == 'O' and button2['text'] == 'O' and button3['text'] == 'O' or
button4['text'] == 'O' and button5['text'] == 'O' and button6['text'] == 'O' or
button7['text'] == 'O' and button8['text'] == 'O' and button9['text'] == 'O' or
button1['text'] == 'O' and button5['text'] == 'O' and button9['text'] == 'O' or
button3['text'] == 'O' and button5['text'] == 'O' and button7['text'] == 'O' or
button1['text'] == 'O' and button2['text'] == 'O' and button3['text'] == 'O' or
button1['text'] == 'O' and button4['text'] == 'O' and button7['text'] == 'O' or
button2['text'] == 'O' and button5['text'] == 'O' and button8['text'] == 'O' or
button7['text'] == 'O' and button6['text'] == 'O' and button9['text'] == 'O'):
      disableButton()       tkinter.messagebox.showinfo("Tic-Tac-
Toe", playerb)



buttons = StringVar()

label = Label( tk, text="Player X:", font='Times 20 bold', bg='white', fg='black', height=1, width=8)
label.grid(row=1, column=0)
```

```
label = Label( tk, text="Player O:", font='Times 20 bold', bg='white', fg='black', height=1, width=8)
label.grid(row=2, column=0)

button1 = Button(tk, text=" ", font='Times 20 bold', bg='gray', fg='white', height=4, width=8,
command=lambda: btnClick(button1)) button1.grid(row=3, column=0)

button2 = Button(tk, text=' ', font='Times 20 bold', bg='gray', fg='white', height=4, width=8,
command=lambda: btnClick(button2)) button2.grid(row=3, column=1)

button3 = Button(tk, text=' ',font='Times 20 bold', bg='gray', fg='white', height=4, width=8,
command=lambda: btnClick(button3)) button3.grid(row=3, column=2)

button4 = Button(tk, text=' ', font='Times 20 bold', bg='gray', fg='white', height=4, width=8,
command=lambda: btnClick(button4)) button4.grid(row=4, column=0)

button5 = Button(tk, text=' ', font='Times 20 bold', bg='gray', fg='white', height=4, width=8,
command=lambda: btnClick(button5)) button5.grid(row=4, column=1)

button6 = Button(tk, text=' ', font='Times 20 bold', bg='gray', fg='white', height=4, width=8,
command=lambda: btnClick(button6)) button6.grid(row=4, column=2)

button7 = Button(tk, text=' ', font='Times 20 bold', bg='gray', fg='white', height=4, width=8,
command=lambda: btnClick(button7)) button7.grid(row=5, column=0)

button8 = Button(tk, text=' ', font='Times 20 bold', bg='gray', fg='white', height=4, width=8,
command=lambda: btnClick(button8)) button8.grid(row=5, column=1)

button9 = Button(tk, text=' ', font='Times 20 bold', bg='gray', fg='white', height=4, width=8,
command=lambda: btnClick(button9)) button9.grid(row=5, column=2)
```

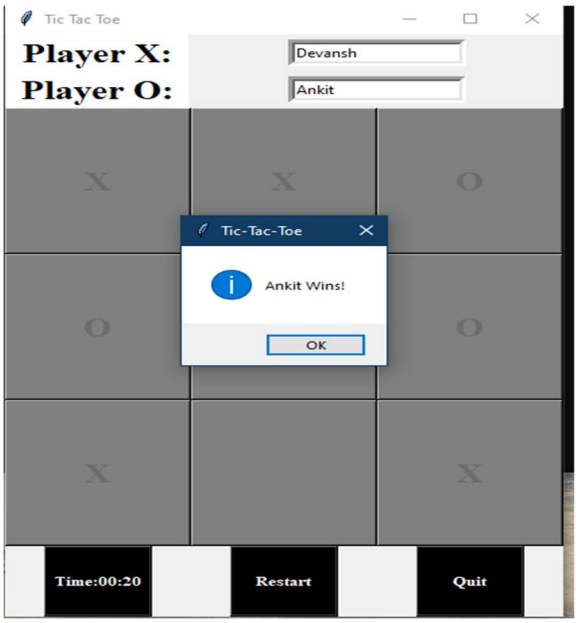
```
tme = Button(tk, text='Time:00:30', font='Times 10 bold', bg='black', fg='white', height=4, width=10,
command=None) tme.grid(row=6, column=0)

res = Button(tk, text='Restart', font='Times 10 bold', bg='black', fg='white', height=4, width=10,
command=lambda: None) res.grid(row=6, column=1)

qui = Button(tk, text='Quit', font='Times 10 bold', bg='black', fg='white', height=4, width=10,
command=tk.destroy) qui.grid(row=6, column=2)

tk.mainloop()
```

4.2. SCREENSHOTS

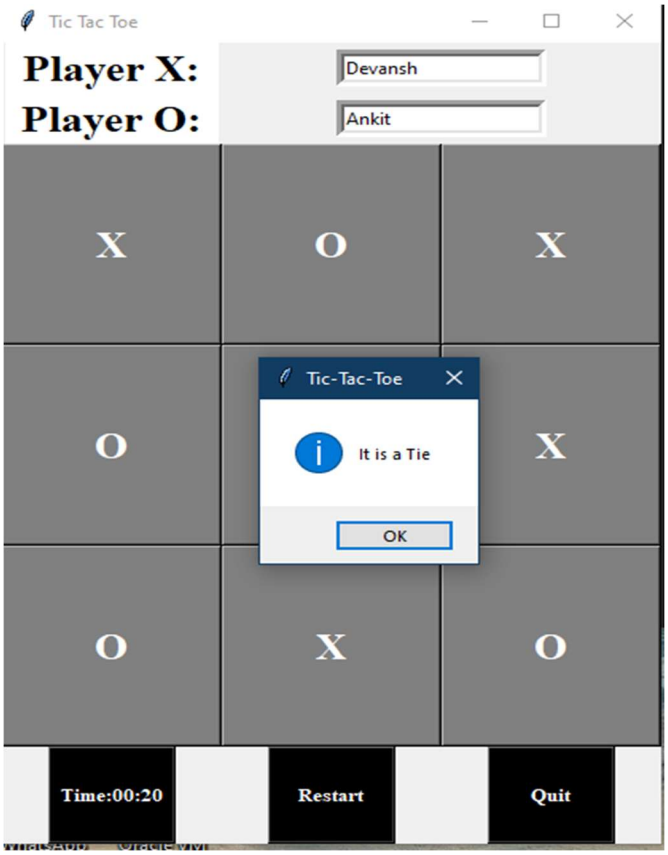
I. **1st Player WINS!**



II. 2nd Player WINS!



III. TIE! Between both players



5. CONLUSION

So, the conclusion of our project is that players has to play and take decisions in a limited spam of time in their childhood game which they used to play on paper on a space of 3x3 grid.

6. BIBLIOGRAPHY

1) Introduction to Programming using Python by Y. Daniel Liang, Pearson

2) Python Programming using Problem Solving Approach by Reema Thareja, Oxford University Press

3) https://w3schools.com

4) https://stackoverflow.com/

5) https://geeksfrgeeks.com/