

A  
Project Report On  
**Tic Tac Toe Game**

**Submitted in the Partial fulfillment of the requirement for awarding the  
degree of Bachelor of Computer Application (BCA)**

**SEM VI, Year 2022-23**

**By**

**AKHIL DELVADIYA(2021000124)**

**AJAY DALSANIYA(2020018330)**

**SHYAM CHAUHAN(2020004235)**

**Project Internal Guide**

**(Prof.Khushbu Juneja)**

**Principal**

**(Dr. Vimal Parmar)**

**DR SUBHASH COLLEGE OF COMPUTER SCIENCE**

**Dr Subhash Technical Campus - Junagadh**

**Affiliated to**

**BHAKTA KAVI NARSINH MEHTA UNIVERSITY – JUNAGADH**



**Dr. Subhash College of Computer Science Junagadh**

## ***Certificate***

**Roll No. 5**

**Enrollment No. 2021000124**

**This is to certify that the Project / practical work was satisfactorily carried out and hence submitted this report is the Bonafede work of Mr. AKHIL DELVADIYA Student of Bachelor of Computer Applications Sem 6 in the BCA Laboratory of Dr. Subhash College of Computer Science Junagadh during the academic year 2022-23.**

**Internal Guide**

---

**HOD**

---

**Principal**

---



**Dr. Subhash College of Computer Science Junagadh**

## ***Certificate***

**Roll No.3**

**Enrollment No. 2020018330**

**This is to certify that the Project / practical work was satisfactorily carried out and hence submitted this report is the Bonafede work of Mr. AJAY DALSANIYA Student of Bachelor of Computer Applications Sem 6 in the BCA Laboratory of Dr. Subhash College of Computer Science Junagadh during the academic year 2022-23.**

**Internal Guide**

---

**HOD**

---

**Principal**

---



**Dr. Subhash College of Computer Science Junagadh**

## ***Certificate***

**Roll No.67**

**Enrollment No. 2020004235**

**This is to certify that the Project / practical work was satisfactorily carried out and hence submitted this report is the Bonafede work Of Mr. SHYAM CHAUHAN Student of Bachelor of Computer Applications Sem 6 in the BCA Laboratory of Dr. Subhash College of Computer Science Junagadh during the academic year 2022-23.**

**Internal Guide**

---

**HOD**

---

**Principal**

---

## **Preface**

This report contains all the outputs of a project undertaken to develop a working model of \_\_\_\_\_ and to study the existing working system. As part of this study, we analyzed the requirements of the working model and tried to improve it using the technological functionality that the current computer science provides.

This volume contains the documentation summary of the working system, the background study and analysis, the technical requirements and the working specification of the developed system.

We recommend the user to take the advantage of the study presented here and implement the system with their required improvisation. We hope this study will reduce the burden of the reanalysis of the development phase.

## **Acknowledgment**

“It is not possible to prepare a project report without the assistance & encouragement of other people. This one is certainly no exception.”

On the very outset of this report, We would like to extend our sincere & heartfelt obligation towards all the personages who have helped us in this endeavor. Without their active guidance, help, cooperation & encouragement, We would not have made headway in the project. We are ineffably indebted to our HOD / Prof. \_\_\_\_\_ for conscientious guidance and encouragement to accomplish this assignment.

We are extremely thankful and pay our gratitude to our project guide Prof \_\_\_\_\_ his/her valuable guidance and support on completion of this project in its presently.

We extend our gratitude to Department Of Computer Science, Dr Subhash Technical Campus for giving us this opportunity.

### **Thanking You**

**Akhil Delvadiya(2021000124)**

**Ajay Dalsaniya(2020018330)**

**Shyam chauhan(2020004235)**

# INDEX

Sr. No.	Title		Page No.
<b>1</b>	<b>Project Profile</b>		<b>1</b>
	1.1	Introduction	1
	1.2	Definition	1
	1.3	Scope	1
	1.4	Objective	1
<b>2</b>	<b>System Analysis and Specification</b>		<b>2</b>
	2.1	Existing System	2
	2.2	Limitations of Existing System	2
	2.3	Feasibility Study	2
	2.4	Front End & Back End Tools	4
<b>3</b>	<b>System Requirement Specification</b>		<b>7</b>
	3.1	Proposed System and Advantages	7
<b>4</b>	<b>System Design</b>		<b>8</b>
	4.1	Data Flow Diagram	8
	4.2	Activity Diagram	10
	4.3	Use Case Diagram	11
	4.4	Data Dictionary (Database)	11
<b>5</b>	<b>Screen Design</b>		<b>12</b>
	5.1	Input Design	12
	5.2	Output Design	12
<b>6</b>	<b>Coding</b>		<b>13</b>
<b>7</b>	<b>Testing (Manual, Test Cases and Test Data)</b>		<b>22</b>
	7.1	Manual Testing	22
	7.2	Test Cases	24
<b>8</b>	<b>Enhancements</b>		<b>26</b>
	8.1	Advantages of Your Project	26
	8.2	Limitation of Your Project	26
	8.3	Future Scope	26
<b>9</b>	<b>References</b>		<b>27</b>

## **1- Project Profile**

### **1.1 Introduction**

TIC-TAC-TOE is a very popular game, so let's implement an automatic tic-tac-toe game using python. python is high level interpreted and general purpose dynamic programming language that focuses on code readability. The python is widely used in bigger organizations because of its multiprogramming paradigm.

Our project is to implement the TIC-TAC-TOE game which will features like multiplayer mode.—

### **1.2 Definition**

Tic tac toe python, also known as Noughts and Crosses or Xs and Os, is very simple two player game where both player get to choose any of symbols between X and O. This game is played on 3X3 grid board and one by one each player gets a chance to mark its respective symbol on the empty spaces of grid.

### **1.3 Scope**

The objective of this tic-tac-toe game python project is to build a tic-tac-toe game so you can play it without wasting paper and improve your concentration. To build this game we can use the tkinter module with the concept of python.

### **1.4 Objectives**

- ✓ To provide user happy environment and good GUI.
- ✓ TIC TAC TOE game is an very easy game which is mostly played among children and to also helps them to improve their concentration.
- ✓ The objective of this TIC TAC TOE game python project is to build a tic tac toe game so you can play it without wasting paper and improve your concentration to build game .we use the tkinter module with concept of python.
- ✓ To play this game require two players to play one is X and the other is O and the both players play by putting their marks in empty squares.
- ✓ To win the game players have to get 3 of her marks in a row(up,down,across,or diagonally).



## **2 . System Analysis and Specification**

### **2.1 Existing System**

The Existing system is a simple game to play with paper and pen between to people. Here the whole process will be carried out in the hand written format making the nine square grids, placing X's and O's and checking the winner. This process will repeat every time. So, it will be a tedious job to draw a nine square grids every time with paper and pen. The human effort is more here. Along with that the retrieval of the information is not easy as the records are maintained in the hand written papers. This application requires correct feed on input into the respective field.

### **2.2 Limitations of Existing System**

The game usually takes less than a minute to play.

People who are very new to the game of Tic-Tac-Toe or have never been taught the various tips and tricks to win at the game may start to feel a little taken advantage of by those who know how to win. At first, it might not be too obvious, but after losing round after round, it can become a tad obvious.

It can get boring if every game ends in a tie.

### **2.3 Feasibility Study**

Feasibility study is the likelihood that whether the system will be useful to the organization or not. After studying the requirements, whether the proposed project is feasible or not is determined by checking the various feasibilities.

The main aim of the feasibility study is to determine whether developing system is financially and technically feasible. The feasibility study involves analysis of the problem and collection of data which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system, as well as study of various constraint on the behavior of the system.

The three aspects in the feasibility study portion are:

- Technical Feasibility
- Economical Feasibility
- Operational Feasibility

Brief introduction of above 3 feasibilities is as under:

#### **➤ Technical Feasibility**

Technical feasibility is considered in terms of technical requirements and their availability in the market. It determines whether the current level of technology supports the proposed system or not. The technical possibility of proposed system is as follow:

- Since the main aim of technical feasibility study is to determine whether unit posses the hardware as well as related software for the project.
- Proposed system does not require much technical detail.
- The current manual system is not so much sufficient for processing of all kinds of tasks.
- It needs low configuration computer system or high configuration computer system and the required operation system.
- It just required windows operating system.
- The organization is ready to purchase the entire required device, required by the proposed system.
- This technical specification is easily available in the market. Therefore the proposed system is technical feasible.

Technical feasibility is considered in terms of technical requirements and their availability in the market. It determines whether the current level of technology supports the proposed system or not. The technical possibility of proposed system is as follow.

In order to develop the web-application except some basic requirement there is no need of extra technology. Also no other workforce is required to develop the application. Yes, Internet is required but you can also run it off-line by sharing

Hardware	Specification
Processor	1 GHz or Higher Intel
MotherBoard	Compatible with the processor
HardDisk	250 Gb or More
RAM	1 or 2 GB
Mouse	Optical
Software	Specification
Front End	IDLE
Back End	SQL Server
Operating System	Microsoft Windows 7 or higher version
Server Technology	Windows Server

### ➤ Economical Feasibility

The economical feasibility is considered in terms of money or rupees value. The organization measures the cost effectiveness of the project. The economical feasibility of the proposed system is as under

- The organization is ready to invest in the proposed system for latest
- As the personnel and the manager know the computer operating, the unit need technology and better result.
- If any of the staff in the firm knows how to operate the computer, than the not have to appoint any computer operator.

- The unit has to spend not much amount for the computer hardware and firm needs number to recruit a new person. Therefore the project possesses the economical feasibility.
- The organization is capable for paying the money for proposed system.
- Therefore the project possesses the economical feasibility.

The economical feasibility is considered in terms of money or rupees value. It measures the cost effectiveness of the project. The economical feasibility of the proposed system is as under. As it is an on-line application a minimal cost of web-hosting will be charged. But it will be affordable. There would not be any charge to have look to site off-line while under construction.

Hardware	Specification	Price
Processor	1 GHz or Higher Intel	₹3380
MotherBoard	Compatible with the processor	₹3230
HardDisk	500 Gb or More	₹1660
RAM	2 GB	₹495
Monitor	Lg 18.5inch LED	4500
SMPS	450 Watt	₹500
Software	Specification	price
Front End	IDLE	₹1999
Back End	SQL Server	₹3000
Operating System	Microsoft Windows 7 or higher version	₹4800
Server Technology	Windows Server	₹3000 approx

### ➤ Operational Feasibility

The operational feasibility deals with the matter whether the proposed system fulfills the requirement of the firm or the department requirement.

The possibility of the operational feasibility is as under.

- The expected users of the proposed system re honestly ready for the new
- The proposed system covers all the aspect of the working system.
- The proposed system will fulfill the firm's requirements.
- The changes made in this system are quite beneficial or not.

We can say that proposed system is operationally beneficial.

The operational feasibility deals with the matter whether the proposed system fulfills the requirement of the user or not.

The possibility of the operational feasibility is as under. Those who have the knowledge of Internet and regularly surfing the sites can easily make a visit to the site. They would not require special skills, just follow the links or use site-maps.

## 2.4 Front End and Back End Tools

### 1.Front End Tool

Python 3.6.0 IDLE 2016

IDLE stand for Integrated Development and learning Environment is a integrated development environment for python. It has been bundled with the default implementation of the language since 1.2.2b.1. .It is packaged as an optional part of the python packaging with many Linux distribution. It is completely written in python3 and

GUI toolkit. IDLE is intended to be a simple IDE and suitable for beginners as well as advanced users. To that end it is cross platform and avoids feature clutter. The features provided by the IDLE includes:

- ✓ Python shell with syntax highlighting.
- ✓ Integrated debugger with stepping, persist ant breakpoints and call stack visibility.
- ✓ Multi-window text editor with syntax highlighting, auto completion, smart indenting etc.

- Development Tools and Technologies

- **Python**

Python is general purpose interpreted, interactive, object-oriented, and high level programming language. It was created by Guido Van Rossum during 1985-1990. Like perl, python source code is also available under the GNU General Public Licence (GPL) Python is named after a TV show called “Monty Python’s Flying Circus”.

Python 3.0(also called “Python 3000” or “Py3K”) was released in December 3,2008. The latest version of python acumulated new and redundant ways to program the same task, python 3.6 had an emphasis on removing duplicate constructs and modules,in keeping with “there should be one and preferable only one- obvious way to do it”. Python’s dynamic typing encourage the programmer to write the code that is clear, well structured as well as easy to understand. The features of dynamic typing are:

- ✓ Types are bounded to values but not to variables.
- ✓ Function and method lookup is done at runtime.
- ✓ Values are inspect-able.
- ✓ There is an interactive interpreter, more than one, in fact.
- ✓ You can list the methods supported by any given object.

Because code is automatically compiled to byte code and executed, python is suitable for use as a scripting language, web application implementing language etc. because of its strong structuring constructs (nested code blocks, functions, modules and packages) and its consistent use of objects and oop. Python enables you to write clear and logical code for small and projects.

### ❖ GUI (Graphical USER INTERFACE):

Our goals in the article is to provide you with and introductory of GUI programming in order to learn the GUI programming. You must first understand a few core aspects of GUI. We’ll access Tk from its python interface called Tkinter. It is not the latest and greter nor does it have the most robust set of GUI’s that erun on most platforms. Setting up GUI application is similar to how an artist must put all the work. In GUI programming a top-level root windowing object certain all the little windowing objects that will be a part of a your GUI application. These can be text labels, buttons, listboxs, frames etc. these individual little GUI components are known as widgets. Top level windows are those that show up stand alone as part of your application. Interestingly, you can have more than one top level window for you GUI. But only one of them should be your root window.

The top level window can be created using this,

## Import tkinter Top=tkinter.TK()

The object returned by tkinter.Tk() is usually referred to as the root window. Within this window you can place multiple component pieces together to form your GUI. Tk has three geometry managers that help with positioning your widget set.

- ✓ Placer: you provide the size of widget and locations to place them. This manager then places them for you.
- ✓ Packer: it packs widgets into the correct places.
- ✓ Grid: it is used to specify GUI widget placement based on grid coordinates.

Now once the packer has determined the sizes and alignments of your widgets, it will then place them on the screen for you. When all the widgets are in place we instruct the application to infinite main loop. In Tkinter the code that does it is:

## Tkinter.mainloop()

This is normally the last piece of sequential code your program runs.

## 2.Back End Tool

Python	Coding language
xampp	server
MySQL Database	Database Connectivity

Xampp:-

- It is simple light weight Apache distribution that makes it extremely easy for developer to create a local web server for testing and deployment to create a local web server for testing and deployment purpose.
- Since most actual web server use the same commonly as xampp it makes transitioning from a local test server externally easy as well. • Xampp is also cross platform which means it works equally well on Linux, Mac and windows • Xampp stand for cross platform(X) , Apache(A), MariaDB, PHP. The term xampp is an approved acronym.

**MySQL :-** • MySQL is an open-source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL), the most popular language for adding, accessing and processing data in a database. MySQL is noted mainly for its speed, reliability, and flexibility. It is commonly employed with most of the popular serverside scripting language including Python, PHP, JSP and ASP. It is a multithreaded, multi-user, SQL (Structured Query Language) relational database server (RDBMS). MySQL is available either under the GNU, GPL (General Public Licence) or under other licences when the GPL is inapplicable to the intended use. MySQL is a freely available third-party database engine designed to provide fast access to stored data. Data can be stored, updated and deleted using languages such as python. The data can be retrieved from the database to allow the generation of Dynamic Webpages.

### **3.System Requirement Specification**

#### **3.1 Proposed system and Advantages**

To overcome the drawbacks of the existing system, the proposed system has been evolved. This project aims to reduce the paper work and saving time to generate accurate results from the player's perspective. The system provides with the best graphical user interface. The efficient reports can be generated by using this proposed system.

##### **➤ Advantages of proposed system**

- 1.The game has been made user friendly with proper use of graphical interface.
- 2.The user can play as many games without any interpretation.
- 3.It is highly reliable, approximate result from user.
- 4.The game has been made as a through expert system.
- 5.The player can win the game,draw the game or will lose the game.
- 6.It is good brain exercise for all age group people

##### **• Rules of game:**

Traditionally the first player play with "X", so you can decide who wants to go with "X" and who wants to go with "O".

Only one player can play at a time.

If any of the players have filled a square then the other player and the same player can not override that squares.

There are only two conditions that may match will be draw or may win.

The player that succeed in playing three respective marks(X or O)in a horizontal, vertical, or diagonal row wins the game.

##### **➤ Project implementation**

After initializing this game, a 3 X 3 a hash square board grid will pop up in the screen of the user.

The game will be played between tow players.

One of the players has to one 'O' and other 'X' to mark their respective cells.

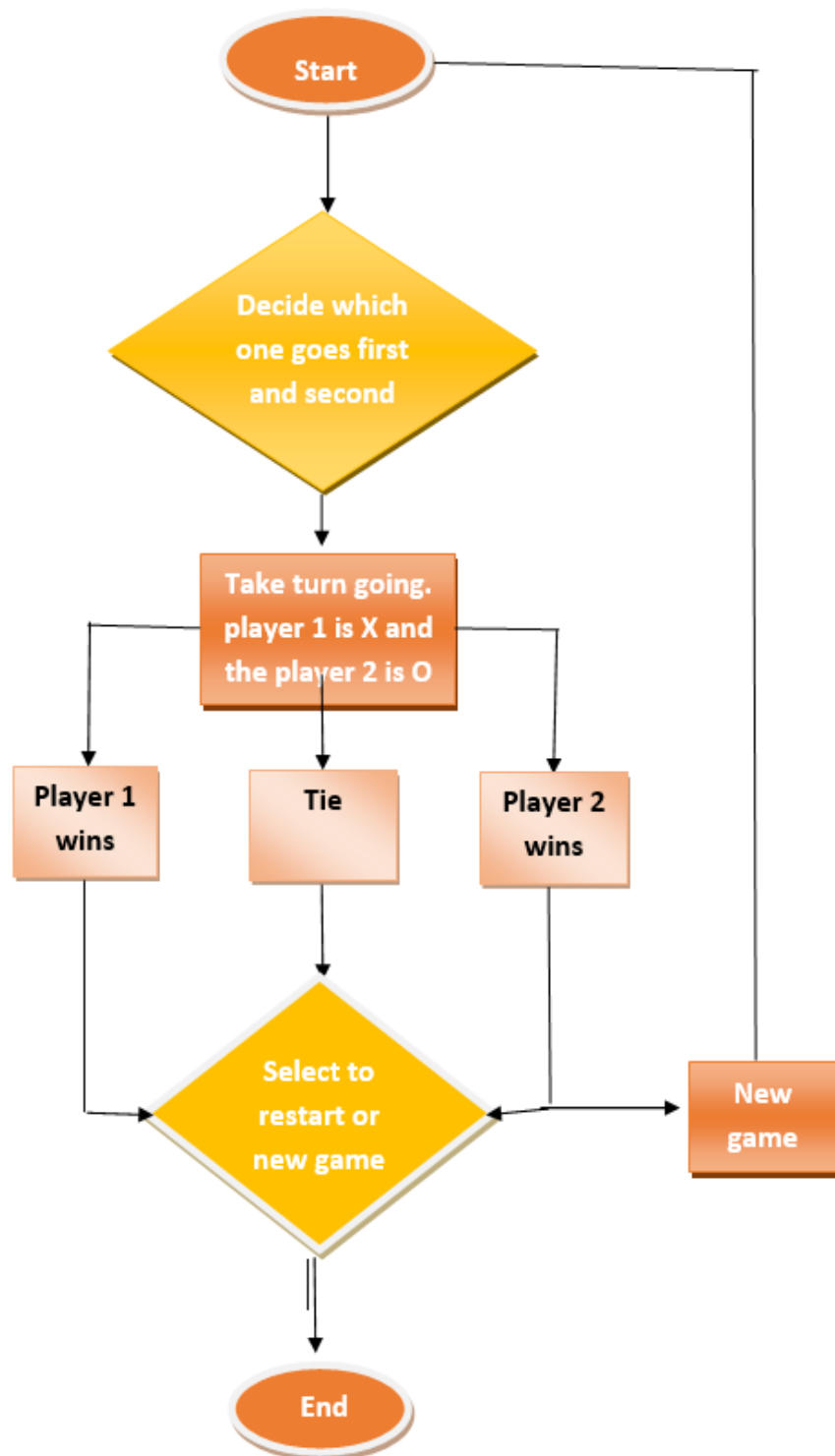
The player will have to input numerical characters, from 1 to 9, to select a position for X or O into the space they for example: they are playing with O and input 2, the O will go to the first row- the second column. If the player wants to place O in the third row =first colun,then they have to enter 7,and it is similar for the other positions.

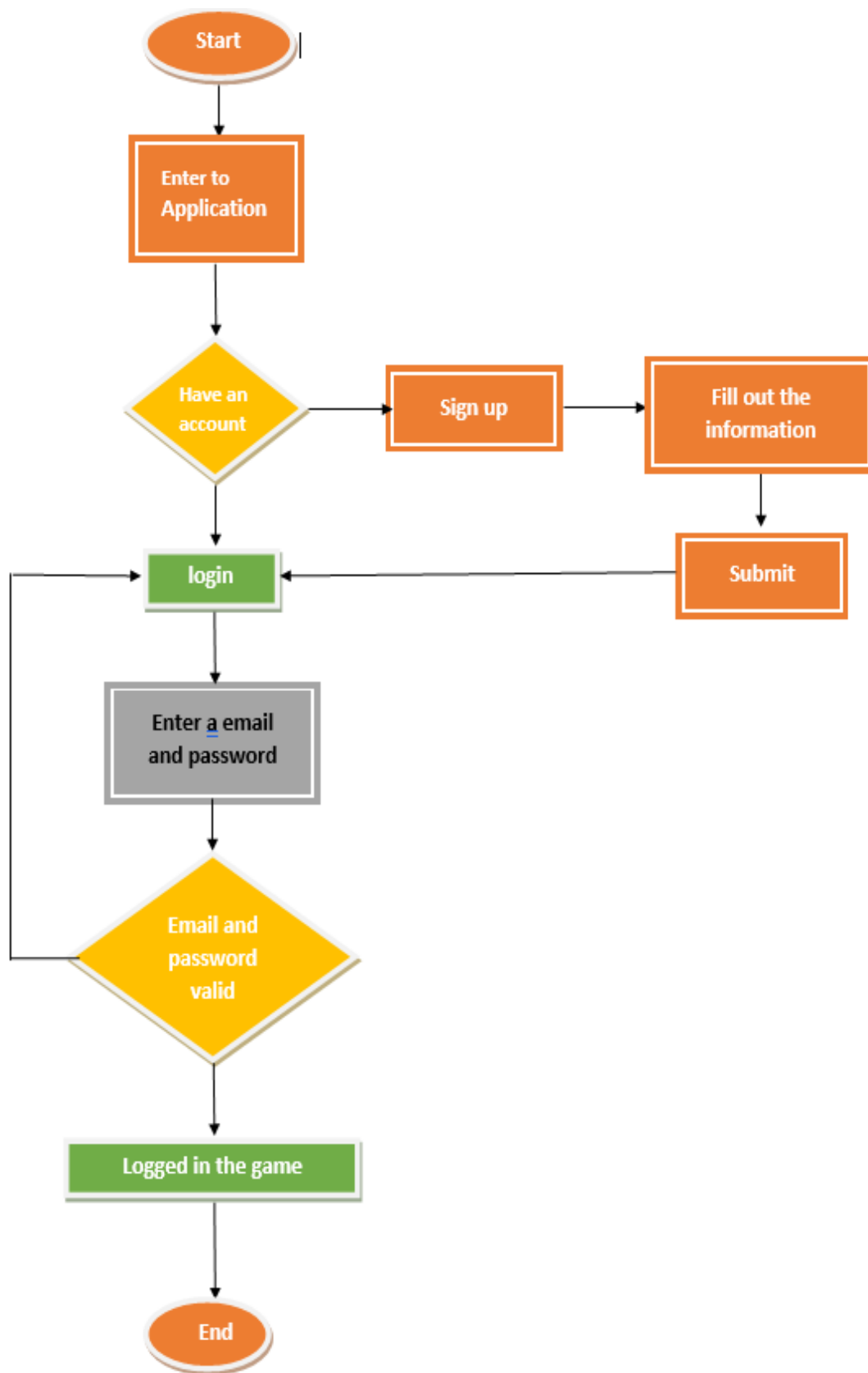
The game starts with one of the players and the game ends when one of the players has one whole row/column/diagonal filled with his/her respective character('O' or 'X').

If the blank spaces in the grid are all filled, and there is no winner, then the game is said to be a draw.

## 4.System Design

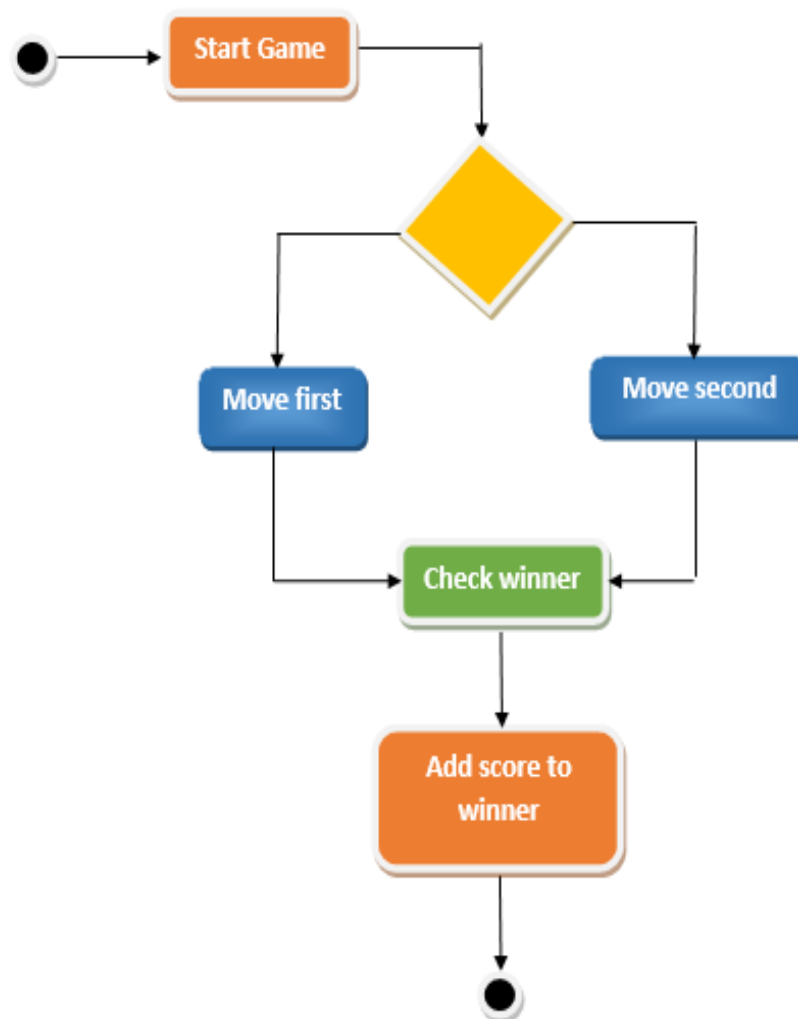
### 4.1 Data flow Diagram



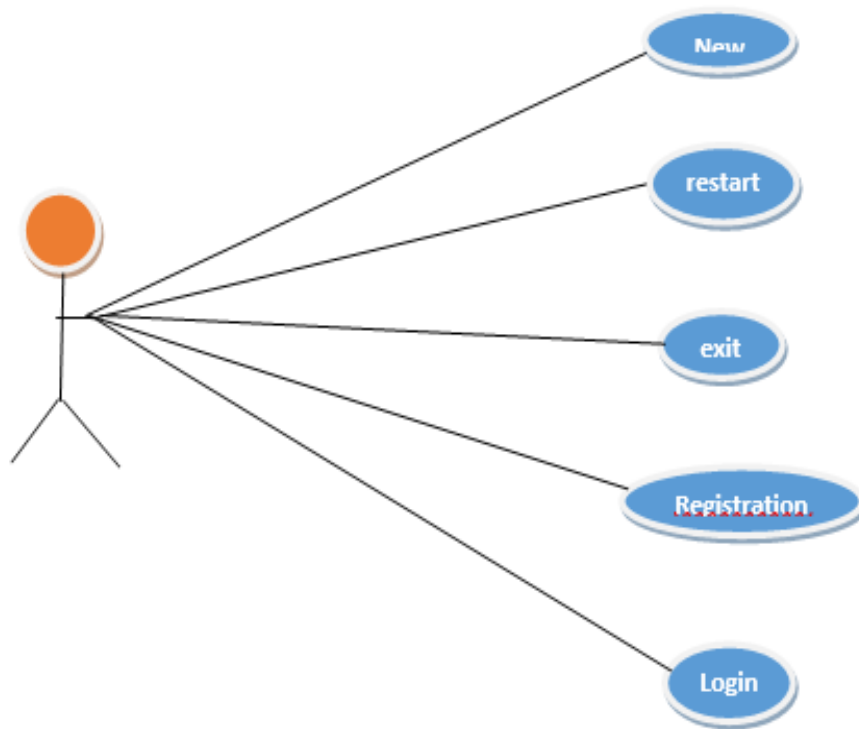
**login**



## 4.2 Activity Diagram



### 4.3 Use case



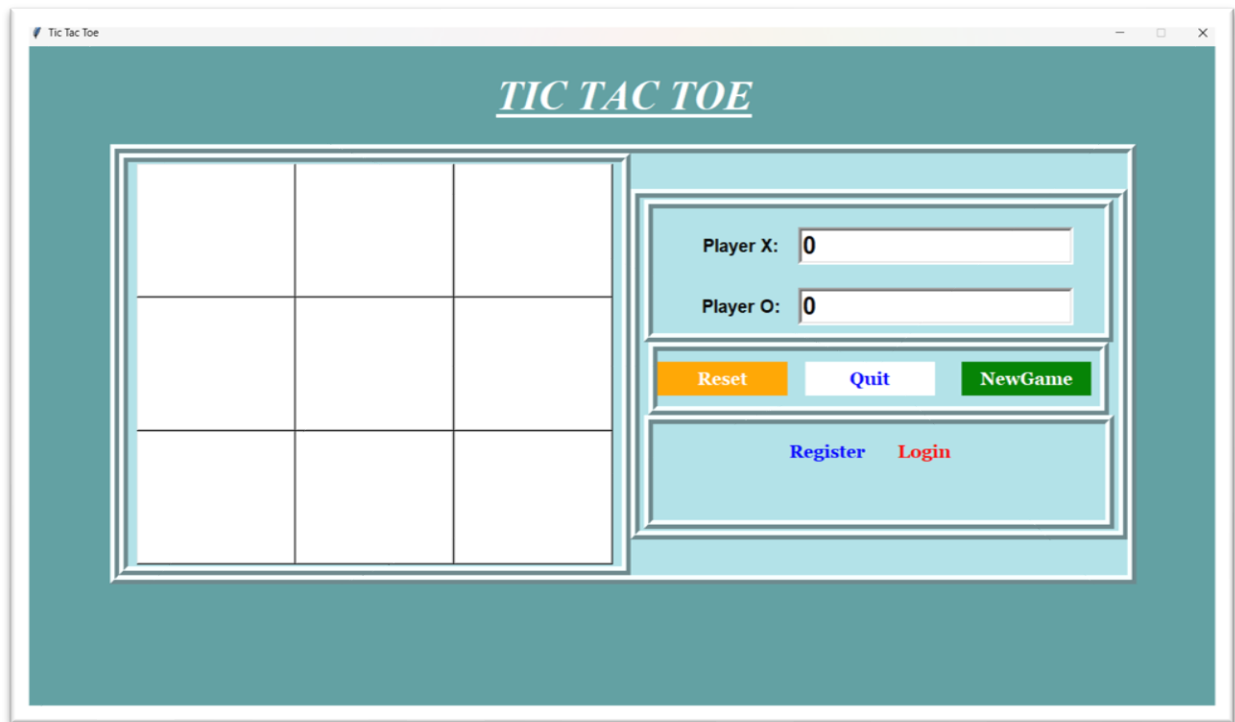
### 4.4 Data Dictionary (Database)

Database Name	Tic Tac Toe
Table	login

login	
firstname	Varchar(50)
lastname	Varchar(50)
email	Varchar(50)
password	Varchar(50)

## 5.Screen Design

### 5.1 Input Design



***TIC TAC TOE***

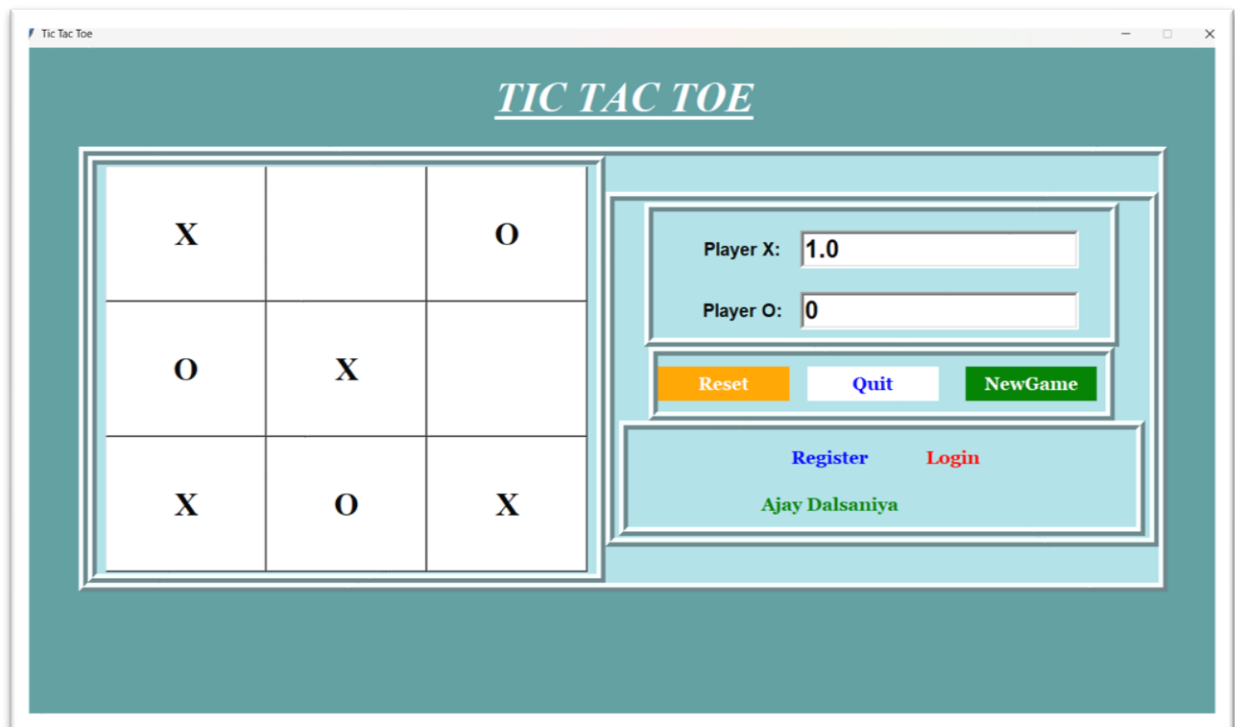

Player X:

Player O:

[Reset](#) [Quit](#) [NewGame](#)

[Register](#) [Login](#)

### 5.2 Output Design



***TIC TAC TOE***

X		O
O	X	
X	O	X

Player X:

Player O:

[Reset](#) [Quit](#) [NewGame](#)

[Register](#) [Login](#)

Ajay Dalsaniya

## 6 Coding

```

from tkinter import *
from tkinter import ttk
import tkinter as tk
from tkinter import messagebox
import mysql.connector

root = Tk()
root.title("Tic Tac Toe")

root.configure(bg="Cadet Blue")
root.resizable(0, 0)
root.geometry("1350x750+10+10")

buttons = StringVar()
i = True

def checker(buttons):
    global i
    if buttons['text'] == "":
        if (i):
            i = False
            buttons['text'] = 'X'
            winner()
        else:
            i = True
            buttons['text'] = 'O'
            winner()

def winner():
    if (btn1['text'] == 'X' and btn2['text'] == 'X' and btn3['text'] == 'X'):
        n = float(playerX.get())
        score = (n + 1)
        playerX.set(score)
        messagebox.showinfo("Game says", 'X is Winner')

    elif (btn4['text'] == 'X' and btn5['text'] == 'X' and btn6['text'] == 'X'):
        n = float(playerX.get())
        score = (n + 1)
        playerX.set(score)
        messagebox.showinfo("Game says", 'X is Winner')

    elif (btn7['text'] == 'X' and btn8['text'] == 'X' and btn9['text'] == 'X'):
        n = float(playerX.get())
        score = (n + 1)
        playerX.set(score)
        messagebox.showinfo("Game says", 'X is Winner')

    elif (btn1['text'] == 'X' and btn4['text'] == 'X' and btn7['text'] == 'X'):
        n = float(playerX.get())
        score = (n + 1)

```

```

playerX.set(score)
messagebox.showinfo("Game says", 'X is Winner')

elif (btn2['text'] == 'X' and btn5['text'] == 'X' and btn8['text'] == 'X'):
    n = float(playerX.get())
    score = (n + 1)
    playerX.set(score)
    messagebox.showinfo("Game says", 'X is Winner')

elif (btn3['text'] == 'X' and btn6['text'] == 'X' and btn9['text'] == 'X'):
    n = float(playerX.get())
    score = (n + 1)
    playerX.set(score)
    messagebox.showinfo("Game says", 'X is Winner')

elif (btn1['text'] == 'X' and btn5['text'] == 'X' and btn9['text'] == 'X'):
    n = float(playerX.get())
    score = (n + 1)
    playerX.set(score)
    messagebox.showinfo("Game says", 'X is Winner')

elif (btn3['text'] == 'X' and btn5['text'] == 'X' and btn7['text'] == 'X'):
    n = float(playerX.get())
    score = (n + 1)
    playerX.set(score)
    messagebox.showinfo("Game says", 'X is Winner')

elif (btn1['text'] == 'O' and btn2['text'] == 'O' and btn3['text'] == 'O'):
    n = float(playerO.get())
    score = (n + 1)
    playerO.set(score)
    messagebox.showinfo("Game says", 'O is Winner')

elif (btn4['text'] == 'O' and btn5['text'] == 'O' and btn6['text'] == 'O'):
    n = float(playerO.get())
    score = (n + 1)
    playerO.set(score)
    messagebox.showinfo("Game says", 'O is Winner')

elif (btn7['text'] == 'O' and btn8['text'] == 'O' and btn9['text'] == 'O'):
    n = float(playerO.get())
    score = (n + 1)
    playerO.set(score)
    messagebox.showinfo("Game says", 'O is Winner')
elif (btn1['text'] == 'O' and btn4['text'] == 'O' and btn7['text'] == 'O'):
    n = float(playerO.get())
    score = (n + 1)
    playerO.set(score)
    messagebox.showinfo("Game says", 'O is Winner')
elif (btn2['text'] == 'O' and btn5['text'] == 'O' and btn8['text'] == 'O'):
    n = float(playerO.get())
    score = (n + 1)
    playerO.set(score)

```

```

        messagebox.showinfo("Game says", 'O is Winner')
    elif (btn3['text'] == 'O' and btn6['text'] == 'O' and btn9['text'] == 'O'):
        n = float(playerO.get())
        score = (n + 1)
        playerO.set(score)
        messagebox.showinfo("Game says", 'O is Winner')
    elif (btn1['text'] == 'O' and btn5['text'] == 'O' and btn9['text'] == 'O'):
        n = float(playerO.get())
        score = (n + 1)
        playerO.set(score)
        messagebox.showinfo("Game says", 'O is Winner')
    elif (btn3['text'] == 'O' and btn5['text'] == 'O' and btn7['text'] == 'O'):
        n = float(playerO.get())
        score = (n + 1)
        playerO.set(score)
        messagebox.showinfo("Game says", 'O is Winner')

    elif (btn1['text'] != " and btn2['text'] != " and btn3['text'] != " and btn4['text'] != "
          and btn5['text'] != " and btn6['text'] != " and btn7['text'] != " and btn8['text'] != " and
          btn1['text'] != "):
        messagebox.showinfo("Game Says", "Tied Game")

def Quit():
    msg = messagebox.askquestion("Confirm", "Are you want to Quit? You still have
chances!")
    if msg == 'yes':
        root.destroy()

def reset():
    btn1['text'] = "
    btn2['text'] = "
    btn3['text'] = "
    btn4['text'] = "
    btn5['text'] = "
    btn6['text'] = "
    btn7['text'] = "
    btn8['text'] = "
    btn9['text'] = "

def NewGame():
    reset()
    playerX.set(0)
    playerO.set(0)

# frames
Tops = Frame(root, bg="Cadet Blue", pady=2, width=1350, height=100, relief=RIDGE)
Tops.grid(row=0, column=0)

MainFrame = Frame(root, bg="Powder Blue", bd=10, height=600, width=1350,
relief=RIDGE)
MainFrame.grid(row=1, column=0)

```

```
Leftframe = Frame(MainFrame, bg="Powder Blue", bd=10, height=490, width=730,  
padx=10, pady=2, relief=RIDGE)  
Leftframe.pack(side=LEFT)
```

```
Rightframe = Frame(MainFrame, bg="Powder Blue", bd=10, height=490, width=750,  
padx=5, pady=2, relief=RIDGE)  
Rightframe.pack(side=RIGHT)
```

```
Rightframe1 = Frame(Rightframe, bg="Powder Blue", bd=10, height=300, width=560,  
padx=35, pady=2, relief=RIDGE)  
Rightframe1.grid(row=0, column=0)
```

```
Rightframe2 = Frame(Rightframe, bg="Powder Blue", bd=10, height=350, width=600,  
padx=0, pady=2, relief=RIDGE)  
Rightframe2.grid(row=1, column=0)
```

```
Rightframe3 = Frame(Rightframe, bg="Powder Blue", bd=10, height=200,  
width=600, padx=145, pady=2, relief=RIDGE)  
Rightframe3.grid(row=2, column=0,)
```

### # design game

```
lbl=Label(root,text="TIC TAC TOE", font=('times', 35, 'bold','underline','italic'),height=2,  
width=20,bg='Cadet Blue',fg='white').grid(row=0,column=0)
```

```
btn1 = Button(Leftframe, text="", font=('times', 26, 'bold'), height=3, width=8, bg='white',  
command=lambda: checker(btn1))  
btn1.grid(row=0, column=0, sticky=S + N + E + W)
```

```
btn2 = Button(Leftframe, text="", font=('times', 26, 'bold'), height=3, width=8, bg='white',  
command=lambda: checker(btn2))  
btn2.grid(row=0, column=1)
```

```
btn3 = Button(Leftframe, text="", font=('times', 26, 'bold'), height=3, width=8, bg='white',  
command=lambda: checker(btn3))  
btn3.grid(row=0, column=2)
```

```
btn4 = Button(Leftframe, text="", font=('times', 26, 'bold'), height=3, width=8, bg='white',  
command=lambda: checker(btn4))  
btn4.grid(row=1, column=0)
```

```
btn5 = Button(Leftframe, text="", font=('times', 26, 'bold'), height=3, width=8, bg='white',  
command=lambda: checker(btn5))  
btn5.grid(row=1, column=1)
```

```
btn6 = Button(Leftframe, text="", font=('times', 26, 'bold'), height=3, width=8, bg='white',  
command=lambda: checker(btn6))  
btn6.grid(row=1, column=2)
```

```
btn7 = Button(Leftframe, text="", font=('times', 26, 'bold'), height=3, width=8, bg='white',  
command=lambda: checker(btn7))  
btn7.grid(row=2, column=0)
```

```
btn8 = Button(Leftframe, text=" ", font=('times', 26, 'bold'), height=3, width=8, bg='white',
              command=lambda: checker(btn8))
btn8.grid(row=2, column=1)
```

```
btn9 = Button(Leftframe, text=" ", font=('times', 26, 'bold'), height=3, width=8, bg='white',
              command=lambda: checker(btn9))
btn9.grid(row=2, column=2)
```

```
reset_btn = Button(Rightframe2, text='Reset', font=('Georgia', 15, 'bold'), height=1,
                  width=10, bg="Orange", fg="white", command=reset, bd=0)
reset_btn.grid(row=1, column=0, pady=10)
```

```
exitButton = Button(Rightframe2, text="Quit", font=('Georgia', 15, 'bold'), height=1,
                   width=10, bg="white", command=Quit, bd=0, fg="Blue")
exitButton.grid(row=1, column=1, padx=20)
```

```
new_btn = Button(Rightframe2, text='NewGame', font=('Georgia', 15, 'bold'), height=1,
                 width=10, bg="Green", fg="white", command=NewGame, bd=0)
new_btn.grid(row=1, column=2, padx=10)
```

```
playerX = IntVar()
playerO = IntVar()
```

```
playerX.set(0)
playerO.set(0)
```

```
lblplayerX = Label(Rightframe1, font=('arial', 15, 'bold'), bg="Powder Blue", text="Player
X:", width=8, height=3,
                  padx=2, pady=2)
lblplayerX.grid(row=0, column=0)
txtplayerX = Entry(Rightframe1, font=('arial', 20, 'bold'), bd=4, fg="Black",
                  textvariable=playerX, width=20,
                  justify=LEFT)
txtplayerX.grid(row=0, column=1)
```

```
lblplayerO = Label(Rightframe1, font=('arial', 15, 'bold'), bg="Powder Blue", text="Player
O:", width=10, height=2,
                  padx=2, pady=2)
lblplayerO.grid(row=2, column=0)
txtplayerO = Entry(Rightframe1, font=('arial', 20, 'bold'), bd=4, fg="Black",
                  textvariable=playerO, width=20,
                  justify=LEFT)
txtplayerO.grid(row=2, column=1)
```

```
reset()
```

### # login from design

```
def datalogin():
    Top = Toplevel()
    Top.title("Log In ")
    # Set the window size
```



```
# Here 0,0 represents the starting point of the window
Top.geometry("1280x800+0+0")
Top.config(bg="white")
```

```
# =====DESIGN PART=====
```

```
frame1 = Frame(Top, bg='red')
frame1.place(x=0, y=0, width=450, relheight=1)
```

```
label1 = Label(Top, text="TIC", font=("times new roman", 28, "bold"), bg="red",
fg="white").place(x=100, y=300)
label2 = Label(Top, text=" TAC ", font=("times new roman", 30, "bold"), bg="red",
fg="white").place(x=162,y=300)
```

```
label3 = Label(Top, text=" TOE", font=("times new roman", 30, "bold"), bg="red",
fg="white").place(x=255,y=300)
```

```
label4 = Label(Top, text="LOGIN...", font=("times new roman", 20, "bold"), bg="red",
fg="Blue").place(x=100, y=360)
```

```
# =====Entry Field & Buttons=====
```

```
frame2 = Frame(Top, bg="gray95")
frame2.place(x=450, y=0, relwidth=1, relheight=1)
```

```
frame3 = Frame(frame2, bg="white",bd=10,relief=RIDGE)
frame3.place(x=140, y=150, width=500, height=450)
```

```
label1 = Label(frame2, text="Login", font=("times new roman", 28, "bold"),
fg="red").place(x=300, y=75)
```

```
email_label = Label(frame3, text="Email Address", font=("helvetica", 20, "bold"),
bg="white", fg="gray").place(x=50,
y=40)
```

```
email_entry = Entry(frame3, font=("times new roman", 15, "bold"), bg="white",
fg="gray",textvariable=emaillogin,bd=5,relief=RIDGE)
email_entry.place(x=50, y=80, width=300)
```

```
password_label = Label(frame3, text="Password", font=("helvetica", 20, "bold"),
bg="white", fg="gray").place(x=50,
y=120)
```

```
password_entry = Entry(frame3, font=("times new roman", 15, "bold"), bg="white",
fg="gray", show="*",textvariable=passwordlogin,bd=5,relief=RIDGE)
password_entry.place(x=50, y=160, width=300)
```

```
login_button = Button(frame3, text="Log In",command=login_func, font=("times new
roman", 15, "bold"), bd=0, cursor="hand2", bg="blue",
fg="white").place(x=50, y=250, width=300)
```

```
passwordlogin = StringVar()
```

```
emaillogin = StringVar()
```

```
def login_func():
```

```
    if emaillogin.get() == "" or passwordlogin.get() == "":
```

```

        messagebox.showerror("Error!", "All fields are required")
    else:
        try:
            connection = mysql.connector.connect(host='localhost', user='root', password="",
            port='3304', database='tic-tac-toe')

            cur = connection.cursor()
            cur.execute("select * from login where email=%s and
            password=%s",(emaillogin.get(), passwordlogin.get()))

            row = cur.fetchone()
            if row == None:
                messagebox.showerror("Error!", "Invalid USERNAME & PASSWORD")
            else:
                connection = mysql.connector.connect(host='localhost', user='root', password="",
                port='3304', database='tic-tac-toe')
                cur = connection.cursor()
                cur.execute("select firstname, lastname from login where email=%s and
                password=%s",(emaillogin.get(), passwordlogin.get()))
                frow = cur.fetchone()

                fatch_lbl.config(text=frow)

                messagebox.showinfo("Success", "Wellcome to the PySeek family")
                # Clear all the entries

            connection.close()

        except Exception as e:
            messagebox.showerror("Error!", f"Error due to {str(e)}")

fname =StringVar()
lname =StringVar()
email =StringVar()
npassword =StringVar()
password =StringVar()
age =StringVar()

# registration data form
def data():
    Top = Toplevel()
    Top.geometry("1280x800+0+0")
    Top.title("Registration")

    #frame = Frame(Top, bg="white")
    #frame.place(x=100,y=50,width=500,height=550)

    frame1 = Frame(Top, bg="Blue")
    frame1.place(x=0, y=0, width=450, relheight = 1)

    frame2 = Frame(Top, bg="gray100")
    frame2.place(x=450,y=0,relwidth=1, relheight=1)

```

```

frame3 = Frame(frame2, bg="white",relief=RIDGE,bd=10)
frame3.place(x=140,y=150,width=500,height=450)

label1 = Label(frame1, text="TIC", font=("helvetica", 30, "bold"), bg="Blue",
fg="white").place(x=100, y=300)
label2 = Label(frame1, text=" TAC TOE", font=("helvetica", 30, "bold"), bg="Blue",
fg="white").place(x=162,y=300)
label3 = Label(frame1, text="Register...", font=("helvetica", 20, "bold"),
bg="Blue",fg="Green2").place(x=100, y=360)

title1 = Label(frame3, text="Sign Up", font=("times new
roman",25,"bold"),bg="white",fg="Blue").place(x=20, y=10)

title2 = Label(frame3, text="Join with us", font=("times new roman",13),bg="white",
fg="red").place(x=20, y=50)

f_name_1 = Label(frame3, text="First name",
font=("helvetica",15,"bold"),bg="white",fg="Blue").place(x=20, y=100)

l_name_1 = Label(frame3, text="Last name",
font=("helvetica",15,"bold"),bg="white",fg="Blue").place(x=240, y=100)

fname_txt = Entry(frame3,font=("arial"),textvariable=fname,bd=4,fg='red',relief=RIDGE)
fname_txt.place(x=20, y=130, width=200)

lname_txt = Entry(frame3,font=("arial"),textvariable=lname,bd=4,fg='red',relief=RIDGE)
lname_txt.place(x=240, y=130, width=200)

email_1 = Label(frame3, text="Email",
font=("helvetica",15,"bold"),bg="white",fg="Blue").place(x=20, y=180)

email_txt = Entry(frame3,font=("arial"),textvariable=email,bd=4,fg='red',relief=RIDGE)
email_txt.place(x=20, y=210, width=420)

password_1 = Label(frame3, text="Password", font=("helvetica", 15, "bold"),
bg="white",fg="Blue").place(x=200, y=260)

password_txt = Entry(frame3,
font=("arial"),show="*",textvariable=password,bd=4,fg='red',relief=RIDGE)
password_txt.place(x=150, y=300, width=200)

signup = Button(frame3,text="Sign Up",font=("times new roman",18,
"bold"),bd=0,cursor="hand2",bg="green2",fg="Blue",command=registor).place(x=150,y=380,width=220)

```

#### # insert data store in database

```

def registor():
    #if fname.get()==" " or lname.get()==" " or email.get()==" " or password.get()==" " :
        #messagebox.showerror("Error!","Sorry!, All fields are required")

    if fname.get()==" ":
        messagebox.showinfo("Alert", "Enter your name first")

```

```

elif lname.get()=="":
    messagebox.showinfo("Alert", "Enter your name Last")

elif email.get()=="":
    messagebox.showinfo("Alert", "Enter Email")

elif password.get()=="":
    messagebox.showinfo("Alert", "Enter Password")

else:
    try:
        connection = mysql.connector.connect(host="localhost", user="root",
        password="",port='3304', database=" tic-tac-toe")
        cur = connection.cursor()

        cur.execute("select email from login where email=('email.get())")

        row=cur.fetchone()

        # Check if th entered email id is already exists or not.
        if row != None:
            messagebox.showerror("Error!", "The email id is already exists, please try again
with another email id")
        else:
            cur.execute("insert into login (firstname,lastname,email,password)
values(%s,%s,%s,%s)",
            (fname.get(),lname.get(),email.get(),password.get()))

            connection.commit()
            connection.close()
            messagebox.showinfo("Congratulations!", "Register Successful")

    except Exception as es:
        messagebox.showerror("Error!",f"Error due to{str(es)}")

btn_register = Button(Rightframe3, text="Register", font=('Georgia', 15,
'bold'),fg="Blue",bg="Powder Blue",bd=0,
                    command=data) # button registration command through registration from(data)
btn_register.grid(row=0, column=0,pady=10)

btn_login = Button(Rightframe3, text="Login", font=('Georgia', 15, 'bold'), bg="Powder
Blue",bd=0,fg="Red",
                    command=datalogin) # button login command through login from(datalogin)
btn_login.grid(row=0, column=1,padx=20)
fatch_lbl = Label(Rightframe3,font=("Georgia",15,'bold'),bg="Powder Blue",fg='Green')
fatch_lbl.place()
fatch_lbl.grid(row=1,column=0,pady=8)
root.mainloop()
class SignUp:
    pass

```

## 7. Testing

### 7.1 Manual Testing

Manual Testing is a type of software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identify the bugs, issues, and defects in the software application. Manual software testing is the most primitive technique of all testing types and it helps to find critical bugs in the software application.

Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual Testing concepts does not require knowledge of any testing tool. One of the Software Testing Fundamental is “100% Automation is not possible“. This makes Manual Testing imperative.

The key concept of manual testing is to ensure that the application is error free and it is working in conformance to the specified functional requirements.

Test Suites or cases, are designed during the testing phase and should have 100% test coverage.

It also makes sure that reported defects are fixed by developers and re-testing has been performed by testers on the fixed defects.

Basically, this testing checks the quality of the system and delivers bug-free product to the customer.

### Black Box Testing

**Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.



The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation

### Black Box Testing Techniques

Following are the prominent amongst the many used in Black box Testing

- **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
- **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable

by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.

- **Decision Table Testing:** A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

## White Box Testing

**White Box Testing** is a testing technique in which software's internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

The term "Whitebox" was used because of the see-through box concept. The clear box or Whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

## White Box Testing Techniques

A major White box testing technique is Code Coverage analysis. Code Coverage analysis eliminates gaps in a Test Case suite. It identifies areas of a program that are not exercised by a set of test cases. Once gaps are identified, you create test cases to verify untested parts of the code, thereby increasing the quality of the software product

There are automated tools available to perform Code coverage analysis. Below are a few coverage analysis techniques a box tester can use:

**Statement Coverage:-** This technique requires every possible statement in the code to be tested at least once during the testing process of software engineering .

**Branch Coverage –** This technique checks every possible path (if-else and other conditional loops) of a software application.

Apart from above, there are numerous coverage types such as Condition Coverage, Multiple Condition Coverage, Path Coverage, Function Coverage etc. Each technique has its own merits and attempts to test (cover) all parts of software code. **Using Statement and Branch coverage you generally attain 80-90% code coverage which is sufficient.**

Following are important Whitebox Testing Techniques:

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Condition Coverage
- Multiple Condition Coverage

- Finite State Machine Coverage
- Path Coverage
- Control flow testing
- Data flow testing

## 7.2 Test Cases

It is suggested to work in team of two (you could use MSTEams to communicate with your team member). You and a partner are developing a program that lets a user play tic tac toe against the computer. The goal is to identify test cases before the code is written (using V-Model). You want to maximize code coverage by testing as many paths through the logic as possible at the same time as controlling time and expense by designing a minimal set of test cases to give you confidence that the code works.

### Rules of the game

- The user can choose to play X or O.
- X always goes first .
- Then X and O take turn placing a marker on the board.
- The goal is to win by placing three markers (X-X-X) or (O-O-O) across one row, down on column , or along a diagonal.

The game often ends in a tie when the nine positions are filled but neither X nor O completed a line.

### Scope of this exercise

You are engaged in an development process. In this increment , you are focusing on the logic of deciding what move to make. In other words., test the artificial intelligence of a computer playing a game.

The logic for computer's move is to try the following in order:

- If you have two pieces in a line and can complete a line by placing the third , put your X or O in Winning cell.\
- If the user has two markers in a line and can complete the line on his/her next move, put your marker in the cell that blocks the user from winning.
- If the middle of the board is free, put a marker there.
- If at least one corner is free, put a marker in a corner. Choose the corner randomly.
- Select randomly from any of the unoccupied cells.

**Note :** The five rules above simplify the logic, and don't always pick the best move. Therefore it is possible for the user to beat the computer. The user does not have to follow the same rules and make very silly, but legal, moves.

**Note :** To analyse these properly you should rely on tried and true methods of information processing For Example , you can come up with shortcut way of referring to conditions a-e above by naming them respectively :

- WIN move, b.-BLOCK move , c. -CENTER move , d.-CORNER move and e.- SIDE move (or RANDOM move )

### The test objective

The test objective is to verify that the program is applying the rules above in all situation . The goal of this exercise is to functional tests. Your tasks is to analyses, design and document enough test cases to be confident of the verification testing .Testing every possibility configuration of pieces is virtually impossible , even for a game as simple as tic-tac -toe .

Your company will outsource your test cases to test specialists who will implement and execute the tests. Tester will compare actual result with expected result and report all deviation as possible defects.



## **8.Enhancements**

### **8.1 Advantages of Your Project**

- **Teaches strategy skills.**

Playing Tic Tac Toe may seem like a game of chance at first, but after playing a few rounds, players usually notice that there's a certain strategy and pattern of movements that can end up in a win. Because of this, it teaches players (both young and old) to start thinking strategically, which can be highly beneficial in other areas of life too.

- **Develops logical thinking.**

There's a great deal of logic that can be applied to a game of Tic-Tac-Toe. Simple moves that are quite logical are the ones that result in a win. This can help people to start thinking about things differently; from a more logical point of view.

- **Improves the ability to concentrate and focus.**

To win at a game of Tic-Tac-Toe, players need to focus or concentrate for a few moments at a time. This practice at focus and concentration can help to develop these skills. This is a great skill to have in all areas of life.

- **Teaches good sportsmanship.**

Both kids and adults can benefit from learning how to be graceful when winning and losing. This is called good sportsmanship. Players of Tic-Tac-Toe must be able to handle themselves whether they win or lose. It's a great lesson to learn from the game.

- **Develops problem-solving skills.**

Winning at Tic-Tac-Toe requires problem-solving. A player must solve the problem of where to put their mark to avoid losing. If you are trying to incorporate some mental stimulation in an older adult's life or introduce basic problem-solving skills to young children, playing Tic-Tac-Toe is a great option.

### **8.2 Limitation of your project**

- GUI is not so attractive.
- Only mouse interface is implemented, keyboard is not activated in the game.

### **8.3 Future scope**

- keyboard functions will be added.
- we want to design more complex for the games in future
- Background Music
- Different themes..

## 9.References:

- [www.google.com](http://www.google.com)
- [www.youtube.com](http://www.youtube.com)
- [www.w3school.com](http://www.w3school.com)