

**FACULTY OF ENGINEERING AND
TECHNOLOGY**

SCHOOL OF COMPUTING

**DEPARTMENT OF COMPUTING
TECHNOLOGIES**

**18CSC301T FORMAL LANGUAGE AND AUTOMATA
MINI PROJECT REPORT**

PROJECT TITLE: Tic Tac Toe using GUI



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

TEAM MEMBER

- 1. RA2111003010222- KALPANA**
- 2. RA2111003010240- NAMRATHA**
- 3. RA2111003010214 - LIKHITA**
- 4. RA2111003010250- KEERTHANA**

Objective:

The Objective is to create a Tic Tac Toe game using a graphical user interface (GUI) in Python. The objective is to design a user-friendly interface where players can interact with the game by clicking on the game board cells represented as buttons. The GUI should visually update to display the moves made by each player and determine the winner or a draw condition.

The Tic Tac Toe game follows these rules:

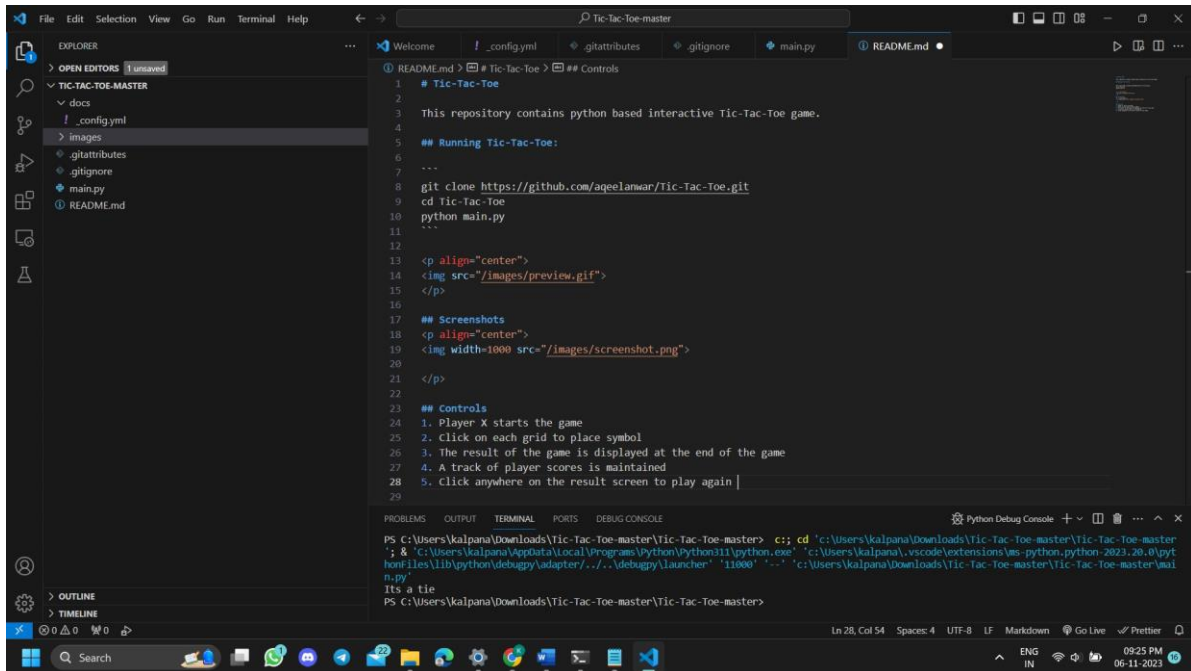
1. The game is played on a 3x3 grid.
2. Two players take turns, one marking "X" and the other marking "O" in empty cells.
3. The game continues until there is a winner or all cells are filled without a winning combination.
4. A player wins if they have three of their symbols in a row, column, or diagonal.
5. If no player achieves a winning combination and all cells are filled, the game ends in a draw.

The GUI should provide an intuitive and visually appealing interface to represent the game board and allow players to make their moves easily. It should update the game state and display the outcome of each game, offering options to restart the game or exit the application.

Overall, the problem involves integrating the game logic, GUI design, event handling, and game flow to create an interactive and enjoyable Tic Tac Toe game using a GUI in Python

Software Used: Visual Studio Code

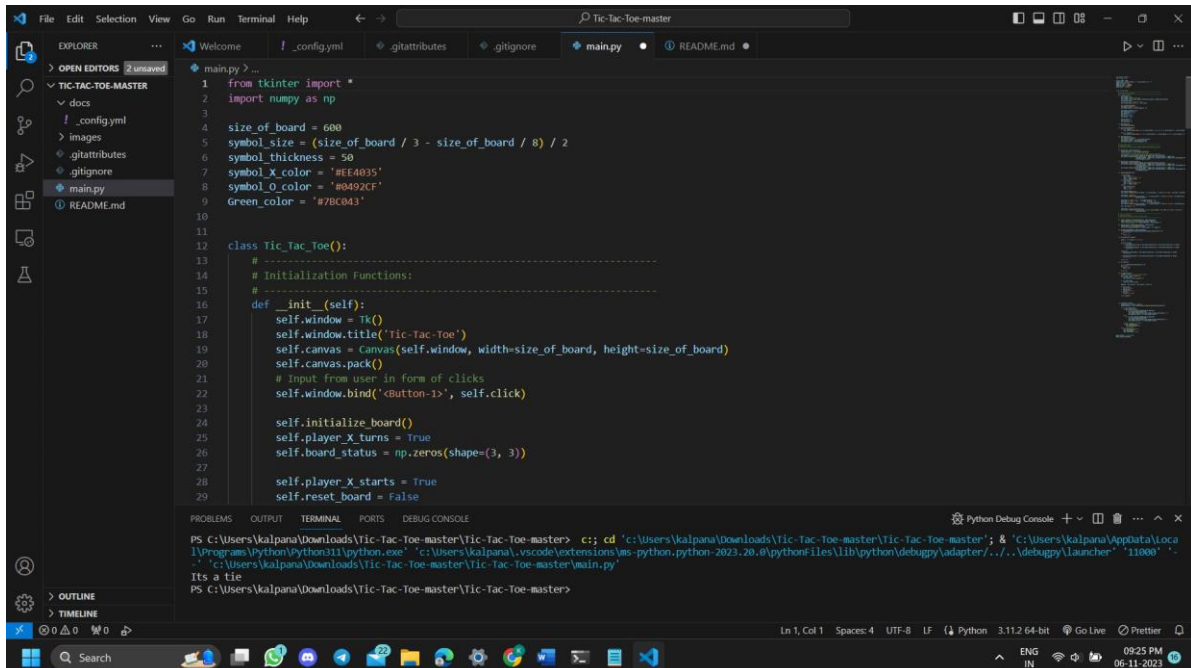
Screen Shots:



This screenshot shows the Visual Studio Code interface with the README.md file open. The file contains instructions for cloning the repository and running the game. The terminal at the bottom shows the command to run the game, which results in a tie.

```
1  # Tic-Tac-Toe
2
3  This repository contains python based interactive Tic-Tac-Toe game.
4
5  ## Running Tic-Tac-Toe:
6  ...
7
8  git clone https://github.com/ageelanwar/Tic-Tac-Toe.git
9  cd Tic-Tac-Toe
10 python main.py
11 ...
12
13 <p align="center">
14 
15 </p>
16
17 ## Screenshots
18 <p align="center">
19 
20 </p>
21
22 ## Controls
23 1. Player X starts the game
24 2. Click on each grid to place symbol
25 3. The result of the game is displayed at the end of the game
26 4. A track of player scores is maintained
27 5. Click anywhere on the result screen to play again |
28
29
```

```
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master> c;; cd 'c:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master'; & 'c:\Users\kalpana\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\kalpana\vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '11000' '-.' 'c:\Users\kalpana\Downloads\Tic-Tac-Toe-master\main.py'
It's a tie
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master>
```



This screenshot shows the Visual Studio Code interface with the main.py file open. The file contains the Python code for the Tic-Tac-Toe game, including imports, constants, and the main game logic.

```
1  from tkinter import *
2  import numpy as np
3
4  size_of_board = 600
5  symbol_size = (size_of_board / 3 - size_of_board / 8) / 2
6  symbol_thickness = 50
7  symbol_X_color = '#EE4835'
8  symbol_O_color = '#0492CF'
9  Green_color = '#7BC043'
10
11
12  class Tic_Tac_Toe():
13      #
14      # Initialization Functions:
15      # -----
16      def __init__(self):
17          self.window = Tk()
18          self.window.title('Tic-Tac-Toe')
19          self.canvas = Canvas(self.window, width=size_of_board, height=size_of_board)
20          self.canvas.pack()
21          # Input from user in form of clicks
22          self.window.bind('<Button-1>', self.click)
23
24          self.initialize_board()
25          self.player_X_turns = True
26          self.board_status = np.zeros(shape=(3, 3))
27
28          self.player_X_starts = True
29          self.reset_board = False
```

```
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master> c;; cd 'c:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master'; & 'c:\Users\kalpana\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\kalpana\vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '11000' '-.' 'c:\Users\kalpana\Downloads\Tic-Tac-Toe-master\main.py'
It's a tie
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master>
```

The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure: `TIC-TAC-TOE-MASTER` (root), `docs`, `!_config.yml`, `images`, `.gitattributes`, `.gitignore`, `main.py`, and `README.md`. The `main.py` file is open in the editor, showing the following code:

```
28 self.player_X_starts = True
29 self.reset_board = False
30 self.gameover = False
31 self.tie = False
32 self.X_wins = False
33 self.O_wins = False
34
35 self.X_score = 0
36 self.O_score = 0
37 self.tie_score = 0
38
39 def mainloop(self):
40     self.window.mainloop()
41
42 def initialize_board(self):
43     for i in range(2):
44         self.canvas.create_line((i + 1) * size_of_board / 3, 0, (i + 1) * size_of_board / 3, size_of_board)
45
46     for i in range(2):
47         self.canvas.create_line(0, (i + 1) * size_of_board / 3, size_of_board, (i + 1) * size_of_board / 3)
48
49 def play_again(self):
50     self.initialize_board()
51     self.player_X_starts = not self.player_X_starts
52     self.player_X_turns = self.player_X_starts
53     self.board_status = np.zeros(shape=(3, 3))
54
55 # -----
56 # Drawing Functions:
```

The bottom panel shows the terminal with the following command and output:

```
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master> c:: cd 'c:\Users\kalpana\Downloads\Tic-Tac-Toe-master'; & 'c:\Users\kalpana\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\kalpana\.vscode\extensions\ms-python.python-2023.20.0\pythonfiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '11000' '-
Its a tie
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master>
```

The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure: `TIC-TAC-TOE-MASTER` (root), `docs`, `!_config.yml`, `images`, `.gitattributes`, `.gitignore`, `main.py`, and `README.md`. The `main.py` file is open in the editor, showing the following code:

```
55 # -----
56 # Drawing Functions:
57 # The modules required to draw required game based object on canvas
58 # -----
59
60 def draw_O(self, logical_position):
61     logical_position = np.array(logical_position)
62     # logical_position = grid value on the board
63     # grid_position = actual pixel values of the center of the grid
64     grid_position = self.convert_logical_to_grid_position(logical_position)
65     self.canvas.create_oval(grid_position[0] - symbol_size, grid_position[1] - symbol_size,
66                             grid_position[0] + symbol_size, grid_position[1] + symbol_size, width=symbol_thickness,
67                             outline=symbol_O_color)
68
69 def draw_X(self, logical_position):
70     grid_position = self.convert_logical_to_grid_position(logical_position)
71     self.canvas.create_line(grid_position[0] - symbol_size, grid_position[1] - symbol_size,
72                             grid_position[0] + symbol_size, grid_position[1] + symbol_size, width=symbol_thickness,
73                             fill=symbol_X_color)
74     self.canvas.create_line(grid_position[0] - symbol_size, grid_position[1] + symbol_size,
75                             grid_position[0] + symbol_size, grid_position[1] - symbol_size, width=symbol_thickness,
76                             fill=symbol_X_color)
77
78 def display_gameover(self):
79
80     if self.X_wins:
81         self.X_score += 1
82         text = 'Winner: Player 1 (X)'
83         color = symbol_X_color
84
85     if self.O_wins:
```

The bottom panel shows the terminal with the following command and output:

```
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master> c:: cd 'c:\Users\kalpana\Downloads\Tic-Tac-Toe-master'; & 'c:\Users\kalpana\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\kalpana\.vscode\extensions\ms-python.python-2023.20.0\pythonfiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '11000' '-
Its a tie
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master>
```

The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure: `TIC-TAC-TOE-MASTER` with subfolders `docs`, `images`, `.gitattributes`, `.gitignore`, `main.py`, and `README.md`. The editor window shows the `display_gameover` method in `main.py`, which handles game over logic, including updating scores and displaying the final state. The terminal at the bottom shows the command `python main.py` being executed, resulting in the output: `Its a tie`.

```
def display_gameover(self):
    if self.X_wins:
        self.X_score += 1
        text = 'winner: Player 1 (X)'
        color = symbol_X_color
    elif self.O_wins:
        self.O_score += 1
        text = 'winner: Player 2 (O)'
        color = symbol_O_color
    else:
        self.tie_score += 1
        text = 'Its a tie'
        color = 'gray'

    self.canvas.delete("all")
    self.canvas.create_text(size_of_board / 2, size_of_board / 3, font="cmr 60 bold", fill=color, text=text)

    score_text = 'Scores \n'
    self.canvas.create_text(size_of_board / 2, 5 * size_of_board / 8, font="cmr 40 bold", fill=Green_color,
                           text=score_text)

    score_text = 'Player 1 (X) : ' + str(self.X_score) + '\n'
    score_text += 'Player 2 (O): ' + str(self.O_score) + '\n'
    score_text += 'Tie : ' + str(self.tie_score)
    self.canvas.create_text(size_of_board / 2, 3 * size_of_board / 4, font="cmr 30 bold", fill=Green_color,
                           text=score_text)
    self.reset_board = True
```

Terminal output:

```
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master> python main.py
Its a tie
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master>
```

The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure: `TIC-TAC-TOE-MASTER` with subfolders `docs`, `images`, `.gitattributes`, `.gitignore`, `main.py`, and `README.md`. The editor window shows the `reset_board` method and logical functions in `main.py`, including `convert_logical_to_grid_position`, `convert_grid_to_logical_position`, `is_grid_occupied`, and `is_winner`. The terminal at the bottom shows the command `python main.py` being executed, resulting in the output: `Its a tie`.

```
self.reset_board = True
score_text = 'Click to play again \n'
self.canvas.create_text(size_of_board / 2, 15 * size_of_board / 16, font="cmr 20 bold", fill="gray",
                        text=score_text)

# -----
# Logical Functions:
# The modules required to carry out game logic
# -----

def convert_logical_to_grid_position(self, logical_position):
    logical_position = np.array(logical_position, dtype=int)
    return (size_of_board / 3) * logical_position + size_of_board / 6

def convert_grid_to_logical_position(self, grid_position):
    grid_position = np.array(grid_position)
    return np.array(grid_position // (size_of_board / 3), dtype=int)

def is_grid_occupied(self, logical_position):
    if self.board_status[logical_position[0]][logical_position[1]] == 0:
        return False
    else:
        return True

def is_winner(self, player):
    player = -1 if player == 'X' else 1
```

Terminal output:

```
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master> python main.py
Its a tie
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master>
```

This screenshot shows the VS Code editor with the file `main.py` open. The code defines methods to check for a win or a tie in a Tic-Tac-Toe game. The `is_gameover` method checks for three in a row, three in a column, and two diagonals. The `is_tie` method checks if the board is full and no one has won. The `is_gameover` method returns `True` if there is a win or a tie, and `False` otherwise.

```
132 player = -1 if player == 'X' else 1
133
134 # Three in a row
135 for i in range(3):
136     if self.board_status[i][0] == self.board_status[i][1] == self.board_status[i][2] == player:
137         return True
138     if self.board_status[0][i] == self.board_status[1][i] == self.board_status[2][i] == player:
139         return True
140
141 # Diagonals
142 if self.board_status[0][0] == self.board_status[1][1] == self.board_status[2][2] == player:
143     return True
144
145 if self.board_status[0][2] == self.board_status[1][1] == self.board_status[2][0] == player:
146     return True
147
148 return False
149
150 def is_tie(self):
151
152     r, c = np.where(self.board_status == 0)
153     tie = False
154     if len(r) == 0:
155         tie = True
156
157     return tie
158
159 def is_gameover(self):
160     # Either someone wins or all grid occupied
```

The terminal output shows the command to run the game and the result:

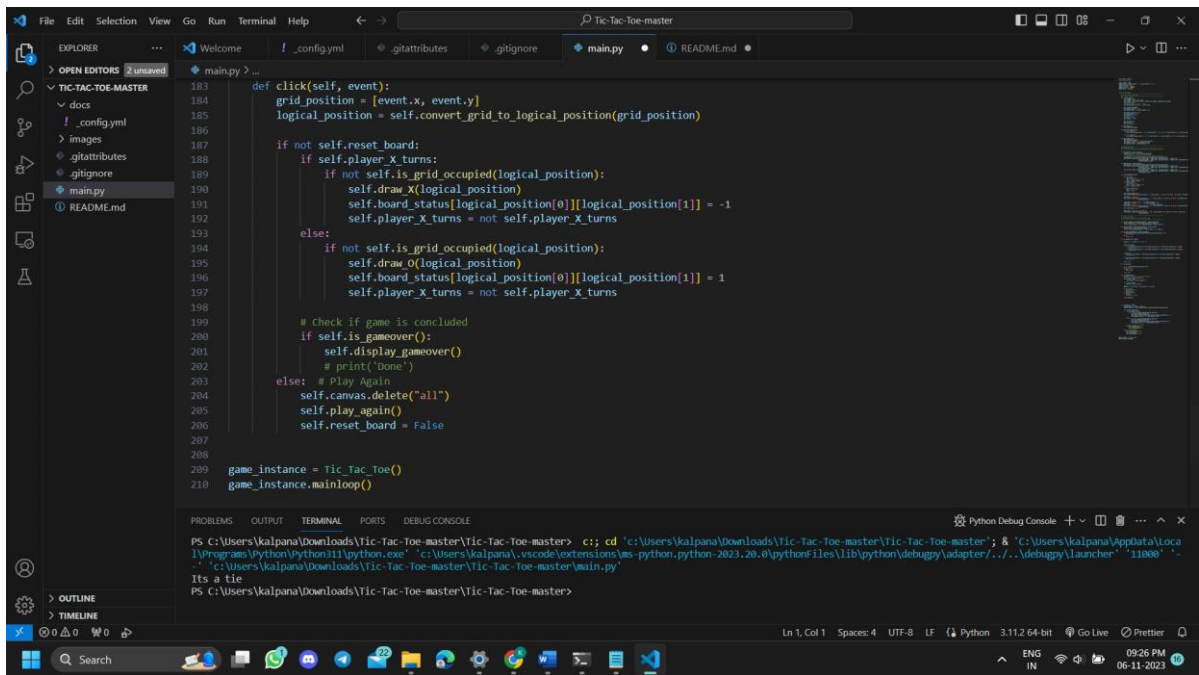
```
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master> c:\cd 'c:\Users\kalpana\Downloads\Tic-Tac-Toe-master'; c:\cd 'c:\Users\kalpana\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\kalpana\.vscode\extensions\ms-python.python-2023.20.0\pythonfiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '11000' '-
' 'c:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master\main.py'
Its a tie
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master>
```

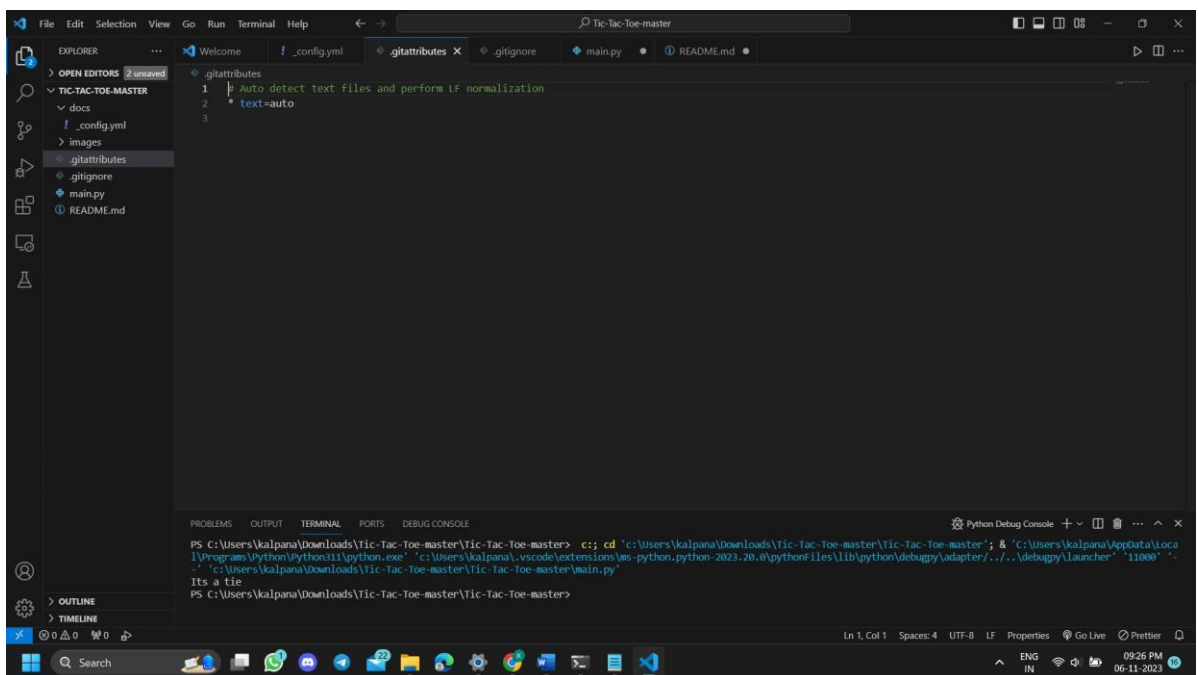
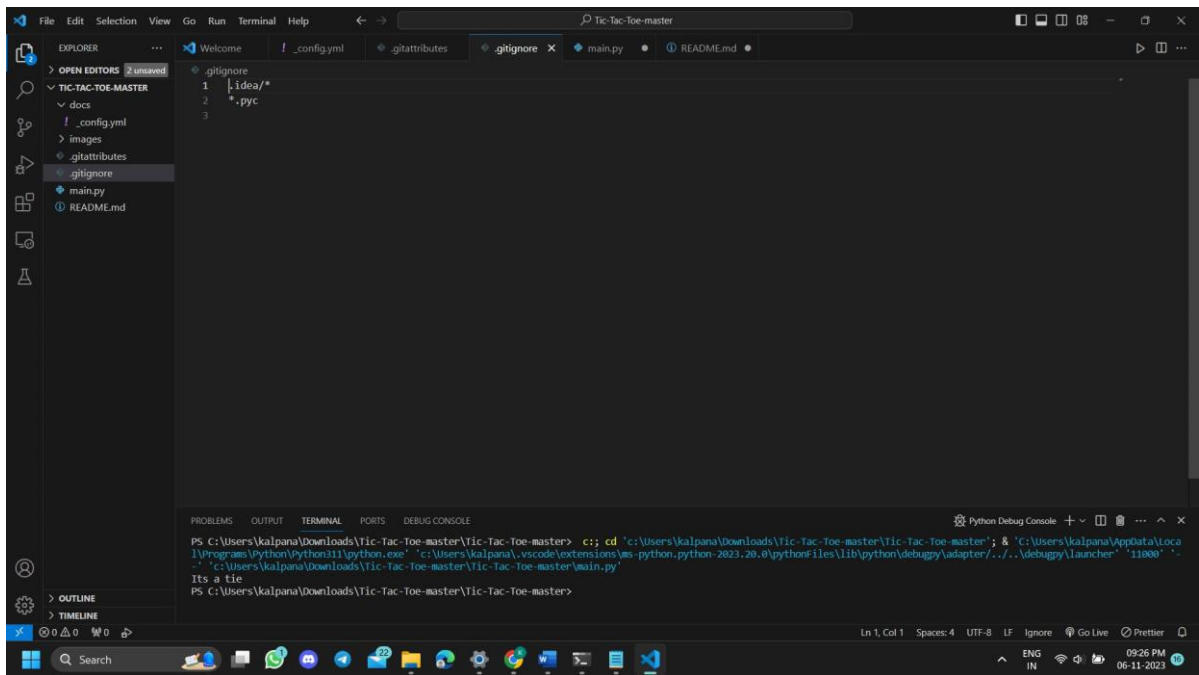
This screenshot shows the VS Code editor with the file `main.py` open. The code defines methods to handle a click and check for a game over. The `click` method updates the board status based on the click event. The `is_gameover` method checks for a win or a tie and returns the game over status. The `click` method prints the board status and the game over status.

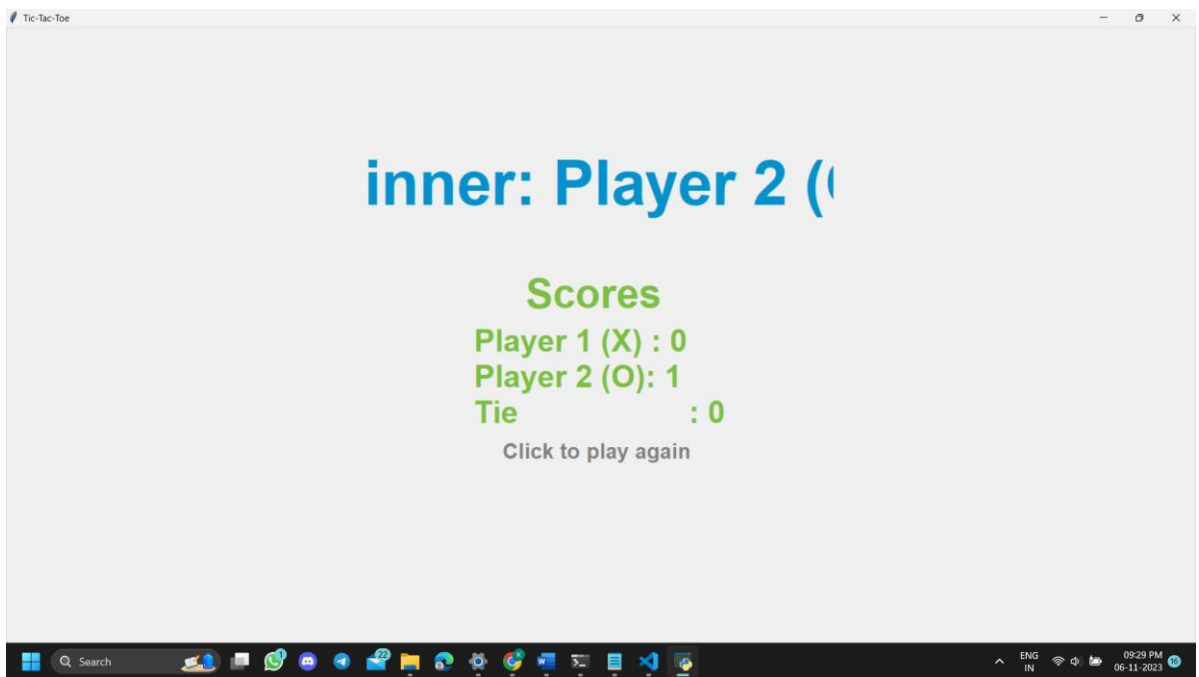
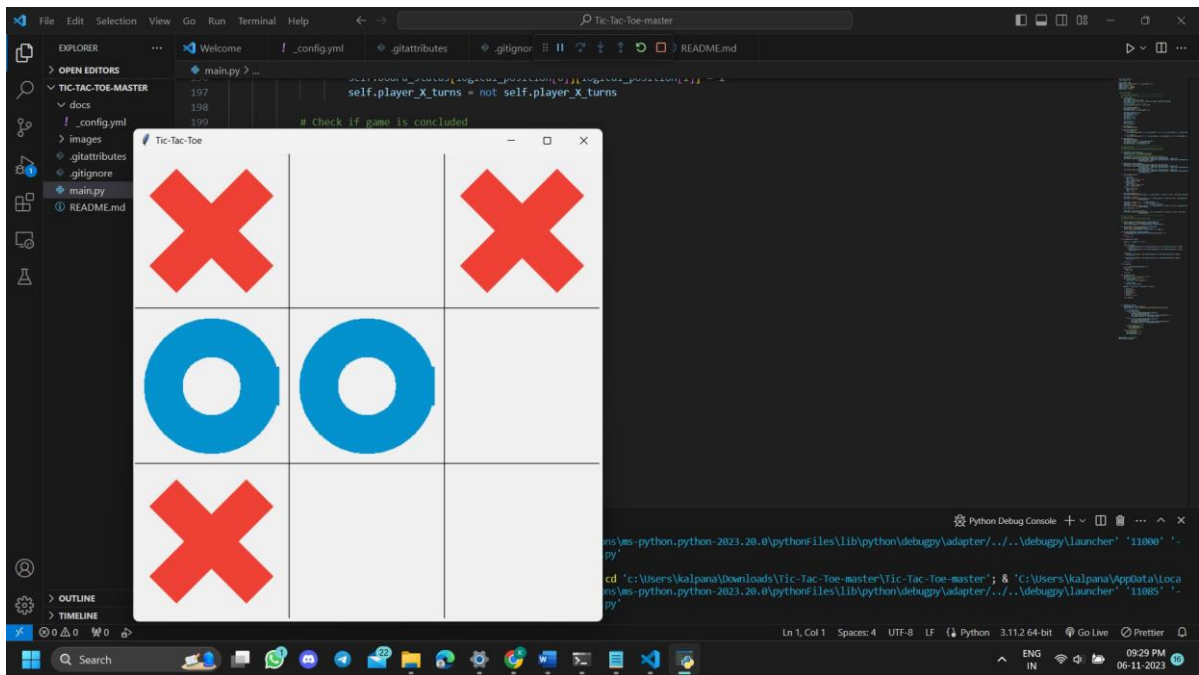
```
159 def is_gameover(self):
160     # Either someone wins or all grid occupied
161     self.X_wins = self.is_winner('X')
162     if not self.X_wins:
163         self.O_wins = self.is_winner('O')
164
165     if not self.O_wins:
166         self.tie = self.is_tie()
167
168     gameover = self.X_wins or self.O_wins or self.tie
169
170     if self.X_wins:
171         print('X wins')
172     if self.O_wins:
173         print('O wins')
174     if self.tie:
175         print('Its a tie')
176
177     return gameover
178
179
180
181
182
183 def click(self, event):
184     grid_position = [event.x, event.y]
185     logical_position = self.convert_grid_to_logical_position(grid_position)
186
187     if not self.nasat_board:
```

The terminal output shows the command to run the game and the result:

```
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master> c:\cd 'c:\Users\kalpana\Downloads\Tic-Tac-Toe-master'; & 'c:\Users\kalpana\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\kalpana\.vscode\extensions\ms-python.python-2023.20.0\pythonfiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '11000' '-
' 'c:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master\main.py'
Its a tie
PS C:\Users\kalpana\Downloads\Tic-Tac-Toe-master\Tic-Tac-Toe-master>
```





Video Link (Make it Public):

https://drive.google.com/file/d/1QbOSKF-ZlwFM8ZMl6omRFDZ_MKkmGCj0/view?usp=drivesdk

GitHub Link: <https://github.com/kattakalpana/tic-tac-toe>

Contribution of the author:

We've studied about the related topics and collected the data required and designed the analysis. We've developed the code and executed it in VSCode successfully. We also used geeksforgeeks, javatpoint and shiksha for reference and data collection.