Project Report

Student Name: Manpreet Kaur **UID:** 24MCA20186
Branch: MCA(UIC) **Section/Group :** 24MCA7A
Semester: 2nd **Date of Performance:** 02/04/2025
Subject Name: Design and analysis of algorithms lab **Subject Code:** 24CAP-612

1. **Aim of the practical : Create an interactive GUI-based Tic-Tac-Toe game** where a human player can compete against an AI opponent.

2. **Hardware and Software Requirements :**

➢ **Hardware Requirements:**

1) **Processor (CPU) :**
   - **Minimum**: Any modern dual-core processor (e.g., Intel i3, AMD Ryzen 3).
   - **Recommended:** Quad-core processor (e.g., Intel i5, AMD Ryzen 5) or better for smoother performance, especially when running multiple extensions or working on large projects.

2) **Memory (RAM) :**
   - **Minimum:** 4 GB of RAM. This will work for basic coding tasks.
   - **Recommended:** 8 GB or more, especially if you are running multiple instances of Jupyter, using extensions, or working with large datasets or projects.

3) **Storage:**
   - **Minimum:** 128 GB of storage. An SSD is preferable for faster access and better performance.
   - **Recommended:** 256 GB SSD or more, providing sufficient space for your OS, Jupyter, Python libraries, and project files.

4). Display:

- **Minimum:** A display with at least 1366x768 resolution.
- **Recommended:** Full HD (1920x1080) or higher for better clarity and screen real estate, especially when working with multiple code windows or side-by-side views.

➢ Software Requirements:

1) Operating System:

- **Windows:** Windows 10 pro .

2) Python Installation:

- **Python Version:** Python 3.12.5 . Download the latest version from the official Python website\

3) Install Anaconda and Jupyter Notebook:

- Downlods and install Anaconda from https://repo.anaconda.com/archive/Anaconda3-2022.05-Windows-x86_64.exe
- Open "Anaconda Prompt" by finding it in the windows (start) Menu.
- Type the command in (python - -version) Anaconda was installed.

4) Start Jupyter Notebook:

- Type the command ("Jupyter Notebook") to Start Jupyter Notebook

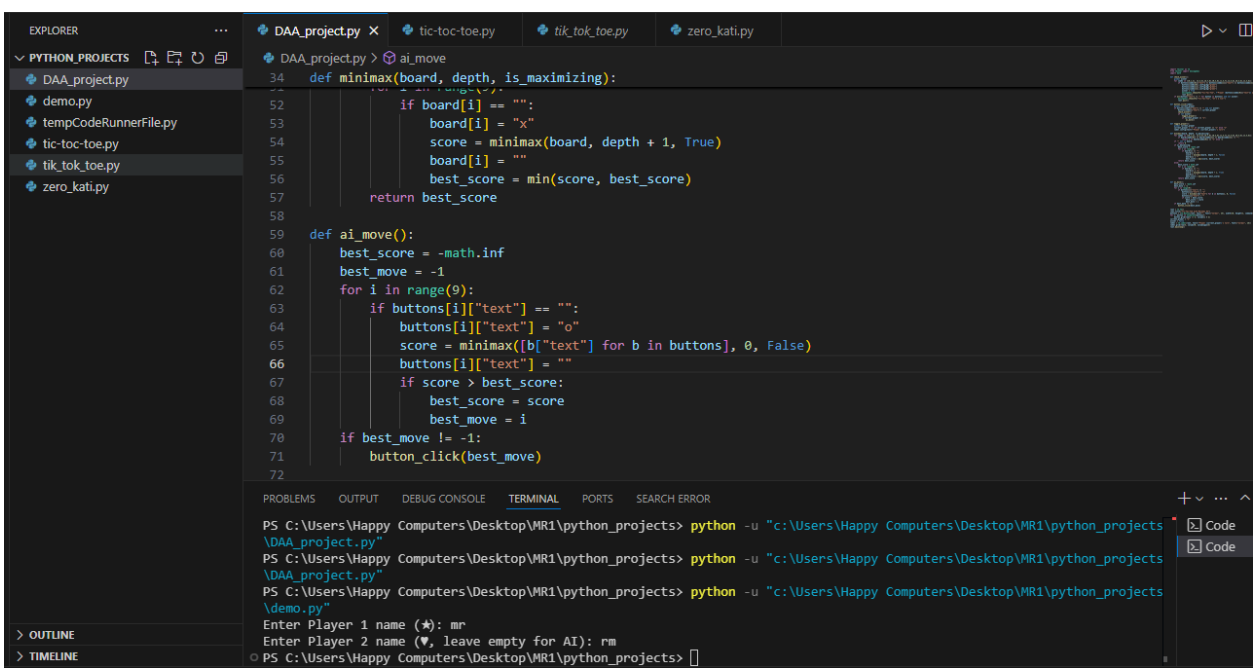
3) Code :

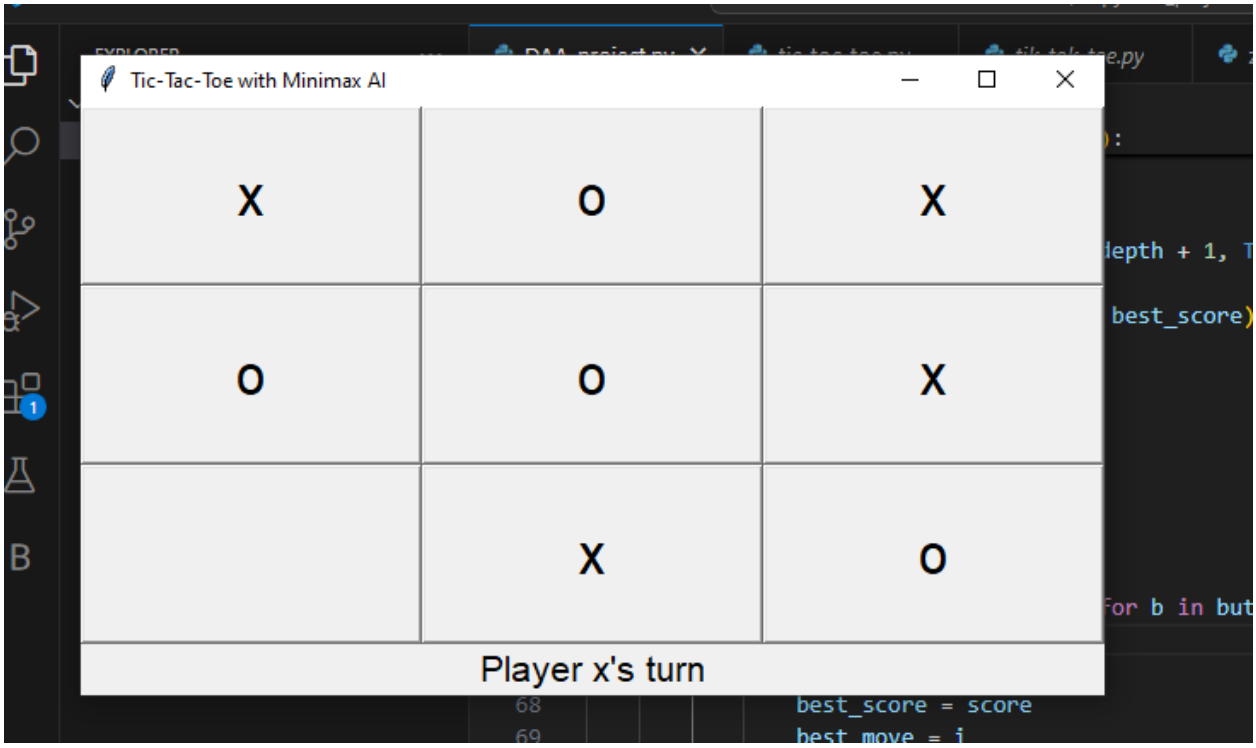
```
import tkinter as tk
from tkinter import messagebox
import math

def check_winner():
    global winner
    for combo in [[0,1,2], [3,4,5],[6,7,8],[0,3,6],[1,4,7],[2,5,8],[0,4,8],[2,4,6]]:
        if buttons[combo[0]]["text"] == buttons[combo[1]]["text"] == buttons[combo[2]]["text"] != "":
            buttons[combo[0]].config(bg="green")
            buttons[combo[1]].config(bg="green")
            buttons[combo[2]].config(bg="green")
            winner = True
            messagebox.showinfo("Tic-Tac-Toe", f"Player {buttons[combo[0]]["text"]} wins!")
            root.quit()
    if all(button["text"] != "" for button in buttons) and not winner:
```

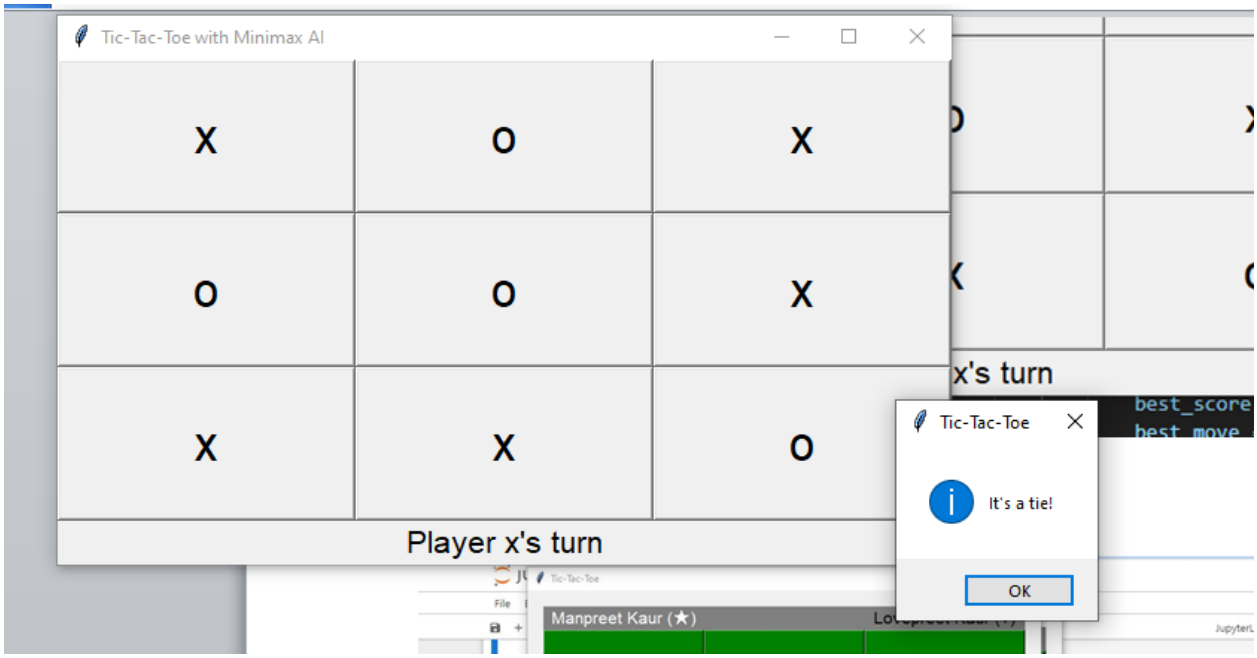
```
        messagebox.showinfo("Tic-Tac-Toe", "It's a tie!")
        root.quit()

def button_click(index):
    global current_player
    if buttons[index]["text"] == "" and not winner:
        buttons[index]["text"] = current_player
        check_winner()
        if not winner:
            toggle_player()
            if current_player == "o":
                ai_move()

def toggle_player():
    global current_player
    current_player = "x" if current_player == "o" else "o"
    label.config(text=f"Player {current_player}'s turn")

def minimax(board, depth, is_maximizing):
    for combo in [[0,1,2], [3,4,5],[6,7,8],[0,3,6],[1,4,7],[2,5,8],[0,4,8],[2,4,6]]:
        if board[combo[0]] == board[combo[1]] == board[combo[2]] != "":
            return 1 if board[combo[0]] == "o" else -1
    if "" not in board:
        return 0
    if is_maximizing:
        best_score = -math.inf
        for i in range(9):
            if board[i] == "":
                board[i] = "o"
                score = minimax(board, depth + 1, False)
                board[i] = ""
                best_score = max(score, best_score)
        return best_score
    else:
        best_score = math.inf
        for i in range(9):
            if board[i] == "":
                board[i] = "x"
                score = minimax(board, depth + 1, True)
                board[i] = ""
                best_score = min(score, best_score)
        return best_score

def ai_move():
    best_score = -math.inf
    best_move = -1
    for i in range(9):
```

```
        if buttons[i]["text"] == "":
            buttons[i]["text"] = "o"
            score = minimax([b["text"] for b in buttons], 0, False)
            buttons[i]["text"] = ""
            if score > best_score:
                best_score = score
                best_move = i
    if best_move != -1:
        button_click(best_move)

root = tk.Tk()
root.title("Tic-Tac-Toe with Minimax AI")
buttons = [tk.Button(root, text="", font=("normal", 25), width=10, height=2, command=lambda i=i:
    button_click(i)) for i in range(9)]
for i, button in enumerate(buttons):
    button.grid(row=i // 3, column=i % 3)
current_player = "x"
winner = False
label = tk.Label(root, text=f"Player {current_player}'s turn", font=("normal", 16))
label.grid(row=3, column=0, columnspan=3)
root.mainloop()
```

4) Result:




Tic-Tac-Toe with Minimax AI
X
O
X
O
O
X
X
O
Player x's turn


Tic-Tac-Toe with Minimax AI
X
O
X
O
O
X
X
X
O
Player x's turn
Tic-Tac-Toe
It's a tie!
OK

5) Learning outcomes (What I have learnt):

- **Making a Simple Game Interface**: Learn how to create a basic window with buttons and labels using `tkinter`.
- Applied **Greedy Algorithm** for AI decision-making
- **Switching Player Turns**: Learn how to make the game switch between Player 1 and Player 2 each turn.
- **Checking for a Win or Draw**: Understand how to check if a player has won or if the game is a tie.
- **Using Labels to Show Messages**: Learn how to display messages like "Player 1's turn" or "Player 2's turn.
- Understood **game theory concepts** like maximizing and minimizing moves.
- Analyzed **time complexity** of Minimax algorithm (**O(9!) in worst case, optimized to O(9) in practice**).