4x4 Tic-Tac-Toe with AI

Submitted to: **Mr Talha Shahid**

1. Executive Summary

Project Overview:

This project implements a 4x4 variant of the traditional Tic-Tac-Toe game with an AI opponent. The core goals were to scale up the game to a 4x4 board, build a responsive GUI using Pygame, and develop an AI opponent using the Minmax algorithm with Alpha-Beta pruning. The project ensures a playable and visually appealing game, where the AI makes strategic decisions based on a limited-depth search.

2. Introduction

Background:

Tic-Tac-Toe is a classical two-player game, typically played on a 3x3 grid. This project extends the game to a 4x4 grid, increasing strategic complexity. The AI leverages classic decision-making techniques to simulate a competitive opponent.

Objectives of the Project:

- Create a 4x4 grid-based Tic-Tac-Toe using Python.
- Implement an AI using Minimax with Alpha-Beta pruning.
- Build a modern GUI using Pygame.
- Make the game interactive and responsive for human users.

3. Game Description

Original Game Rules:

Players take turns placing X or O on a 3x3 grid. The first player to align three marks in a row wins.

Modifications:

- Grid expanded to 4x4.
- Winning condition adjusted to 4 consecutive marks.
- AI integrated with depth-limited search for performance.
- Enhanced UI with modern button styling and replay/quit options.

4. AI Approach and Methodology

AI Techniques Used:

Minimax algorithm with Alpha-Beta pruning for optimal move prediction.

Algorithm and Heuristic Design:

- Decision trees explore future states up to a fixed depth (depth = 3).

- Pruning reduces redundant evaluation, improving efficiency.
- Evaluation function based on game termination state (win/loss/tie).

AI Performance Evaluation:
- AI responds in under 0.5 seconds on average.
- Plays optimally within the limits of the search depth.
- Can win or draw against most players, depending on move order.

5. Game Mechanics and Rules

Modified Game Rules:
- Players alternate turns placing X or O on a 4x4 board.
- The game ends when a player lines up 4 symbols or all cells are filled.
- No diagonal/twist modifications were added (as in Connect 4 variants).

Turn-based Mechanics:
- Player selects a symbol (X or O) at start.
- Human and AI alternate moves until terminal state is reached.

Winning Conditions:
- A player wins by placing 4 same symbols in a row (horizontal, vertical, or diagonal).
- A tie occurs if the board is filled with no winner.

6. Implementation and Development

Development Process:
- GUI designed using Pygame with custom fonts and 3D-style buttons.
- Game logic separated into gamelogic.py, GUI in main.py.
- AI logic implemented using recursive functions with pruning.

Programming Languages and Tools:
- Python
- Pygame (GUI)

Challenges Encountered:
- Optimizing Minimax for a larger board without long delays.
- Managing turn transitions between human and AI.
- Creating an intuitive and attractive GUI.

7. Team Contributions

Team Members and Responsibilities:

SHOAIB: Developed AI logic using Minimax with pruning and finalized UI interactions and testing.

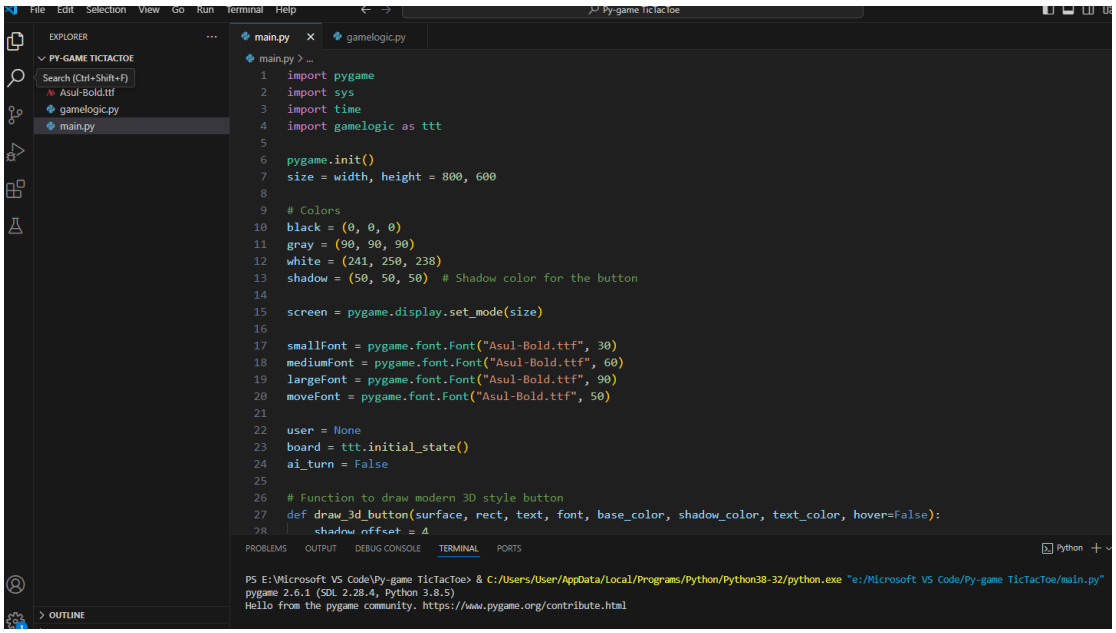
IQRA: - Integrated game logic with GUI and managed player turns and finalized UI interactions and testing.

8. Results and Discussion

AI Performance:

The AI performs well on a 4x4 board with limited depth (3), ensuring quick response times. While a deeper search may result in better late-game moves, performance trade-offs are well balanced. Human players find the AI challenging yet beatable, providing engaging gameplay.

CODE:



9. References

1. Pygame Documentation: https://www.pygame.org
2. GeeksforGeeks – Minimax Algorithm
3. Stack Overflow – Evaluation Functions for Board Games
4. Wikipedia – Tic-Tac-Toe: https://en.wikipedia.org/wiki/Tic-tac-toe