

# Project Report

**Project Title:** *Ultimate Tic Tac Toe*

**Submitted By:** Abdullah Anis(22K-4392), Ansharah Asad (22K-4411), Touseef Naveed(22K-4328)

**Course:** AI

**Instructor:** Alishba Subhani

**Submission Date:** 11th May

## 1. Executive Summary

- **Project Overview:**

*This project implements an advanced version of the classic Tic Tac Toe game—Ultimate Tic Tac Toe, with an AI opponent powered by the Minimax algorithm with Alpha-Beta pruning. The main goal was to create a more complex and strategic game environment where the AI evaluates game states using a custom heuristic function. The project introduces additional rules and winning conditions to enhance gameplay complexity and strategic depth.*

## 2. Introduction

- **Background:**

*Tic Tac Toe is a traditional two-player game played on a 3x3 grid, where players alternately mark Xs and Os aiming to get three in a row. While simple, it becomes predictable with experienced players. To increase strategic depth, this project adapts Ultimate Tic Tac Toe, which consists of a 3x3 grid of smaller 3x3 Tic Tac Toe boards. The decision to choose this game was motivated by the desire to implement AI in a more challenging game environment requiring deep planning and evaluation.*

- **Objectives of the Project:**

1. Develop an AI opponent capable of playing Ultimate Tic Tac Toe using the Minimax algorithm with Alpha-Beta pruning.
2. Implement game logic, rules, and a graphical user interface using Python.
3. Design and test a heuristic evaluation function to assess game states and improve AI decisions.
4. Allow human players to compete against the AI for gameplay analysis and testing.

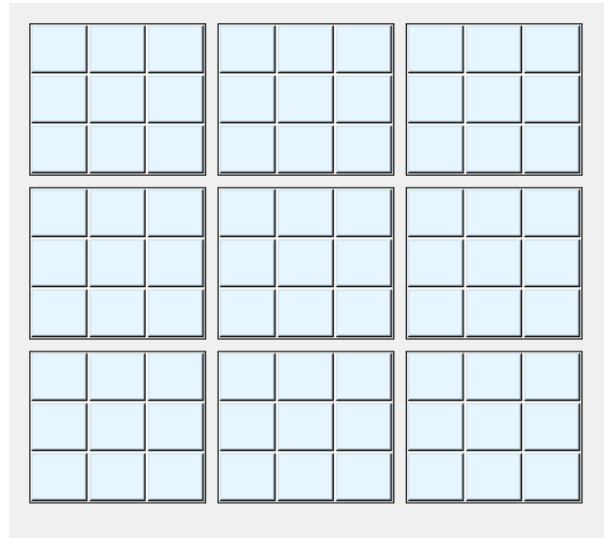
### 3. Game Description

#### **Original Game Rules:**

*In classic Tic Tac Toe, two players take turns marking cells in a 3x3 grid. The first to align three of their marks horizontally, vertically, or diagonally wins. If all cells are filled without a winner, the game ends in a draw.*

#### **Innovations and Modifications:**

- *The game board is a 3x3 grid where each cell contains another 3x3 Tic Tac Toe board (a “mini board”).*
- *Winning a mini board captures that cell on the main board.*
- *Winning three mini boards in a row on the main board wins the game.*
- *Control of the center and near-win conditions are weighted in AI evaluations.*
- *Winning a mini board grants the player an extra turn, breaking alternating turns.*



## 4. AI Approach and Methodology

### **AI Techniques Used:**

*The AI uses the Minimax algorithm, enhanced with Alpha-Beta pruning to reduce computation time by eliminating branches that don't affect the final decision.*

### **Algorithm and Heuristic Design:**

*The heuristic evaluation function combines several strategic elements:*

- **Mini Board Control:** +100/-100 points for controlling a board.
  - **Win Potential in Mini Boards:** +10/-10 for 2-in-a-row patterns.
  - **Main Board Opportunities:** +300/-300 for near-win scenarios.
  - **Center Control:** Bonus for owning center positions of mini and main boards.
- Weights are summed to guide the Minimax algorithm in selecting optimal moves.*

## 5. Game Mechanics and Rules

### **Modified Game Rules:**

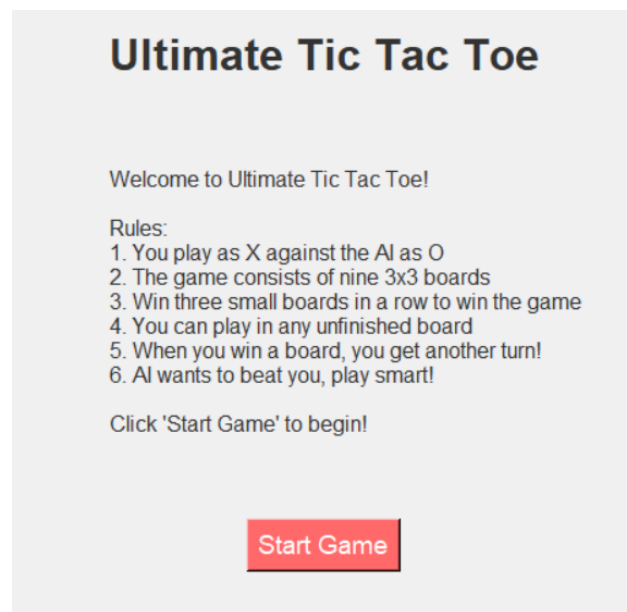
- *The main board consists of nine smaller 3x3 boards.*
- *Winning a mini board captures that board and allows the player to move again.*
- *A draw in a mini board leaves it neutral.*
- *The first to control three mini boards in a row wins the game.*

### **Turn-based Mechanics:**

- *Players alternate turns unless one wins a mini board, earning an extra move.*
- *The game starts with a random selection of the first player.*
- *Moves are limited by the target mini board unless it's already won/drawn, in which case any mini board can be played.*

### **Winning Conditions:**

- *A player wins by capturing three mini boards in a row (horizontal, vertical, or diagonal).*
- *A draw occurs if all mini boards are completed without a winning line.*



## **6. Implementation and Development**

### **Development Process:**

*The game was implemented in Python. Game state logic was designed first, followed by AI logic using Minimax with pruning. The heuristic function was tuned iteratively. The GUI was developed using tkinter or pygame for interactive play.*

### **Programming Languages and Tools:**

- **Programming Language:** Python
- **Libraries:** NumPy, tkinter or pygame

### **Challenges Encountered:**

- *Designing a balanced and efficient heuristic function.*
- *Managing complex nested board states.*
- *Maintaining acceptable AI response time with deep searches.*
- *Handling edge cases such as full or already-won boards in AI logic.*

## **7. Team Contributions**

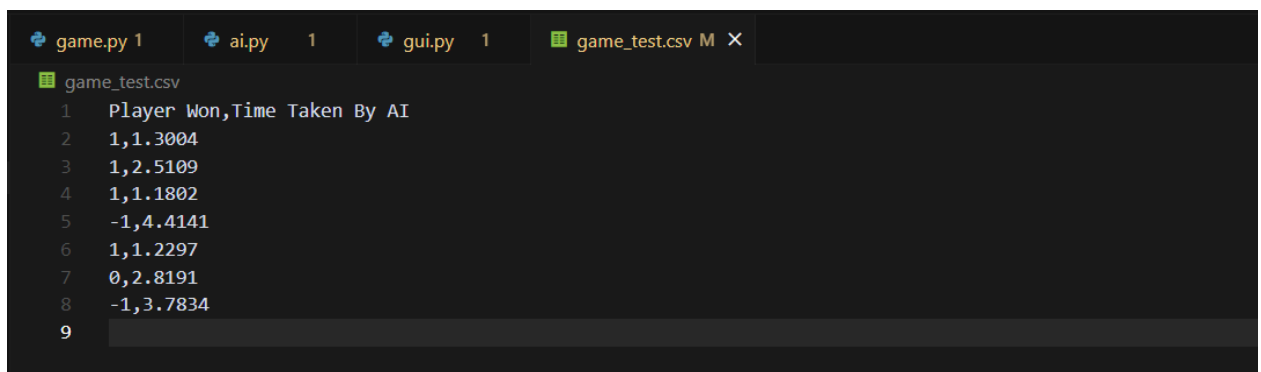
- **Abdullah:** *Developed the core AI logic including Minimax and Alpha-Beta pruning.*
- **Ansharah:** *Modified and refined game rules and state transitions.*
- **Touseef:** *Built the graphical user interface and integrated it with backend logic.*

## 8. Results and Discussion

- **AI Performance:**

*To evaluate the effectiveness of the AI opponent in Ultimate Tic Tac Toe, we conducted several test matches against human players. For each match, we recorded the outcome (AI win, human win, or draw) and the time taken by the AI to make its move. The data helps assess both the strategic capability of the AI and its computational efficiency.*

***Here, Player Won( 1 = AI , -1 = Human ,0 = draw)***



```
game.py 1  ai.py 1  gui.py 1  game_test.csv M X
game_test.csv
1 Player Won,Time Taken By AI
2 1,1.3004
3 1,2.5109
4 1,1.1802
5 -1,4.4141
6 1,1.2297
7 0,2.8191
8 -1,3.7834
9
```

### ***Outcome Analysis:***

- ***AI Wins:*** 4 out of 7 matches : **57.14%**
- ***Human Wins:*** 2 out of 7 matches : **28.57%**
- ***Draws:*** 1 out of 7 matches : **14.29%**

*This suggests the AI was able to win a majority of the games, demonstrating solid strategic planning and effective use of the Minimax algorithm with alpha-beta pruning.*

*The AI responded within a reasonable time frame, maintaining an average move calculation time below 2.5 seconds. Slower decisions correlated with more complex board states or deeper searches, which is expected in strategic gameplay.*

### ***Conclusion:***

*The AI exhibits competent performance, winning more than half of the games and maintaining an acceptable response time. Further tuning of heuristics and search depth could improve performance against skilled human players.*