# Tic Tac Toe – Game Documentation

**Project Type:** Game – Tic Tac Toe using Python & Pygame
**Student Name:** Zeyad Mohamed Ahmed Abdelwahab
**University:** Arab Academy for Science, Technology & Maritime Transport (AAST)
**Faculty:** Computer Science
**Project Name:** Tic Tac Toe – Unbeatable Using MiniMax

## Project Summary

The Tic Tac Toe project is a Python-based game built using Pygame, where two players (or a player vs. AI) take turns marking spaces on a 3x3 grid with either "X" or "O". The player who first aligns three of their marks in a row, column, or diagonal wins the game. If the board is full and no one wins, it's a draw.

This version includes an AI opponent that is unbeatable, achieved using the **Minimax Algorithm**, ensuring that the AI always plays optimally.

---

## Core Classes & Functions

### *Main (in `main.py`)*

- **draw_lines()** – Draws the grid lines of the Tic Tac Toe board.
  - **Parameters:** `color` (default: white)
  - **Function:** Draws the lines separating the rows and columns of the grid.
- **draw_figures()** – Draws the "X" and "O" figures on the board based on the game state.
  - **Parameters:** `color` (default: white)
  - **Function:** Draws circles for "O" and crosses for "X".
- **mark_square()** – Marks a square on the board with the current player's symbol.
  - **Parameters:** `row`, `col` (coordinates on the board), `player` (1 for X, 2 for O)
  - **Function:** Updates the board with the player's symbol.
- **available_square()** – Checks if a square is available to be marked.

    - **Parameters:** `row`, `col` (coordinates of the square)
    - **Function:** Returns `True` if the square is empty, `False` otherwise.
- **is_board_full()** – Checks if the board is completely filled with marks.
    - **Parameters:** `check_board` (default: `board`)
    - **Function:** Returns `True` if there are no empty spaces on the board, `False` otherwise.
- **check_win()** – Checks if the current player has won the game.
    - **Parameters:** `player` (1 for X, 2 for O), `check_board` (default: `board`)
    - **Function:** Returns `True` if the player has won, `False` otherwise.
- **minimax()** – Implements the Minimax Algorithm for the AI to calculate the optimal move.
    - **Parameters:** `minimax_board` (current state of the board), `depth` (recursion depth), `is_max` (boolean to determine whether it's the AI's or player's turn)
    - **Function:** Recursively calculates the best move for the AI.
- **best_move()** – Finds the best move for the AI by evaluating all possible moves.
    - **Function:** Uses the Minimax algorithm to select the best move for the AI.
- **restart_game()** – Restarts the game, clearing the board and resetting the game state.

#### *Game Loop (in `main.py`)*

- **Game Flow:**
    - The game alternates turns between the player and the AI.
    - The player makes a move by clicking on an available square.
    - The AI calculates its move using the Minimax algorithm.
    - The game ends when one player wins or the board is full (draw).
    - The player can restart the game by pressing the "R" key.

#### *AI Logic (in `minimax()` and `best_move()`)*

- The AI uses the **Minimax Algorithm** to evaluate every possible move and select the one that maximizes its chances of winning.
- The algorithm recursively explores the game tree, evaluating potential future moves and their outcomes based on whether the AI is maximizing or minimizing its score.

---

## Game Features

1. **Two-Player Mode:** Players can take turns marking the board with X and O.
2. **Unbeatable AI:** The AI uses the Minimax algorithm to make the best possible move.
3. **Game Reset:** The game can be reset by pressing the "R" key.
4. **Visual Feedback:** Players can see the current state of the board with colored lines and figures.
5. **Game End:** The game announces the winner or declares a draw when the board is full.

---

## Libraries Used

- **Pygame:** For creating the game window, handling user input, and rendering graphics.
- **NumPy:** Used for creating and manipulating the game board as a 2D array.

---

## Visuals

- **Board Grid:** A 3x3 grid with lines separating the rows and columns.
- **Player Symbols:** "X" (cross) and "O" (circle) represent the two players.
- **Winning Line:** The winning player's line will be drawn in green (if player 1 wins) or red (if player 2 wins).
- **Game Over Colors:** The board changes color to indicate the game’s outcome: red for player 2’s victory, green for player 1’s victory, or gray for a draw.