# Connect Four with AI - Project Documentation

---

### Title Page

**Project Title**: Connect Four with AI (Minimax Algorithm + Alpha-Beta Pruning)
**Student Name**: Abdelrahman Ahmed Mahmoud Elsayed (فرقة رابعة)
**University**: Benha University – Faculty of Computer Science
**GitHub Link:** https://github.com/Apdo-Elhogaraty/Connect-Four-with-AI

---

### Abstract

This project presents a digital implementation of the classic Connect Four game, integrated with an artificial intelligence component that utilizes the Minimax algorithm with Alpha-Beta Pruning. The AI decision-making allows the computer to play competitively against a human player, simulating strategic thinking and enhancing the gameplay experience. The project is built using Python and offers a structured approach to understanding game AI fundamentals.

---

### Introduction

Connect Four is a two-player connection game in which the players first choose a color and then take turns dropping colored discs from the top into a vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs.

This project implements the Connect Four game with a graphical interface and an AI component that can play against a human player. The AI is built using the Minimax algorithm with Alpha-Beta pruning for efficient decision-making.

---

### Problem Statement

Traditional board games lack an intelligent system for single-player interaction. This project solves that by implementing a game AI that simulates a human-like opponent capable of making smart decisions and planning ahead.

---

**Objectives**

- Implement the classic Connect Four game.
- Develop a computer player using the Minimax algorithm.
- Optimize the AI using Alpha-Beta pruning to reduce computation.
- Allow human vs AI interaction in a user-friendly way.

---

**System Overview**

- The game board is a 6x7 grid.
- Each turn, a player chooses a column to drop their piece.
- The game continues until a player wins or the board is full.
- The AI uses Minimax + Alpha-Beta pruning to choose the best move.

---

**How the Game Works**

1. Players alternate turns (Human vs AI).
2. On each turn, the current player drops a disc into one of the 7 columns.
3. The disc occupies the next available space from the bottom.
4. The game checks if a player has 4 connected pieces (horizontal, vertical, or diagonal).
5. If a player wins or the board is full, the game ends.

---

**AI Algorithm Used: Minimax with Alpha-Beta Pruning**

**Minimax** is a decision-making algorithm used in two-player games. The AI assumes the opponent plays optimally and evaluates all possible moves to choose the best one.

**Alpha-Beta Pruning** is an optimization that cuts off branches in the game tree that don't need to be explored because they can't possibly affect the final decision.

Steps:

1. At each level of the game tree:

    - Maximize the AI's score when it's its turn.
    - Minimize the opponent's score when it's their turn.
2. The algorithm continues until a maximum depth is reached or the game ends.
3. Alpha stores the best score the maximizer can guarantee.
4. Beta stores the best score the minimizer can guarantee.
5. If beta <= alpha, that branch is pruned.

---

**Main Functions & Their Roles**

- minimax(board, depth, alpha, beta, maximizingPlayer): The core AI function. Uses recursion and evaluates possible moves to choose the optimal one.
- score_position(board, piece): Evaluates the score of the current board for a given piece.
- get_valid_locations(board): Returns all columns where a move can be made.
- get_next_open_row(board, col): Finds the first empty row in a given column.
- drop_piece(board, row, col, piece): Places a piece in the board.
- is_terminal_node(board): Checks if the game is over (win or full board).
- winning_move(board, piece): Checks if the given piece has a winning line.
- pick_best_move(board, piece): Picks the best move without recursion (used for baseline comparison).

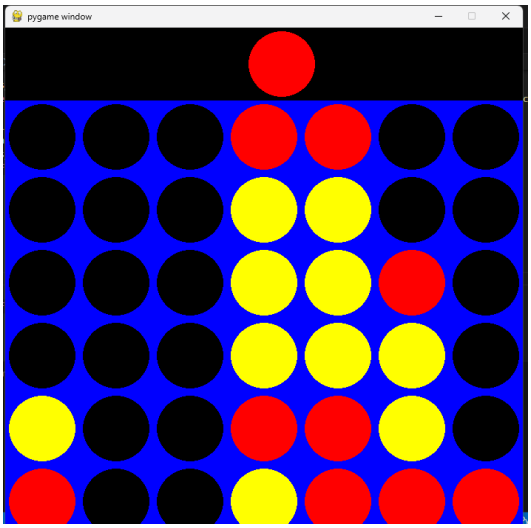
---

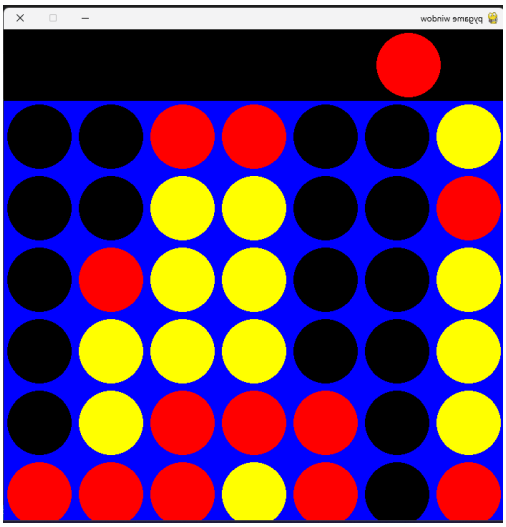
**Technologies Used**

- **Programming Language**: Python
- **Libraries**: NumPy (for board structure), Math, Random , Pygame
- **Game Logic**: Implemented manually using Python logic and loops

---

### Screenshots



---

### Conclusion

This project successfully demonstrates the use of artificial intelligence in a traditional board game. Using the Minimax algorithm with Alpha-Beta pruning allows for smart decision-making in a reasonable time frame, providing an engaging opponent for human players. The system can be extended to include more advanced heuristics or visual enhancements.

---

### Future Work

- Improve the evaluation function for better AI decisions.
- Support multiplayer over network.
- Add difficulty levels for the AI.