# **Project Proposal: AI-Powered Connect Four with Minimax Algorithm**

# Submitted By:

# M.Soubanullah(22k-5002)

M.Ismail Memon(22k-5005)

Asadullah Vohra(22k-5121)

# Course: AI

# Instructor: Ms. Almas Ayesha Ansari

# Submission Date: 25-03-2025

**1. Project Overview**

**Project Topic**

Implementation of Connect Four with an AI opponent using the Minimax algorithm with Alpha-Beta pruning for optimal decision-making.

**Objective**

* Develop a fully functional Connect Four game where a human player can compete against an AI.
* Implement Minimax with Alpha-Beta pruning to create a challenging AI opponent.
* Design a heuristic evaluation function to assess board states.
* Provide adjustable difficulty levels (Easy, Medium, Hard) by varying search depth.

**2. Game Description**

Original Game Rules

* 6 rows × 7 columns grid.
* Players take turns dropping discs (🔴 for human, 🔴 for AI).
* The first player to connect 4 discs (horizontally, vertically, or diagonally) wins.

**Innovations Introduced**

**1. AI Opponent**

* Uses Minimax algorithm for decision-making.
* Alpha-Beta pruning optimizes performance.
* Heuristic function evaluates board strength (e.g., center control, potential threats).

**2. Difficulty Levels**

* Easy: AI makes random but valid moves.
* Medium: Minimax with depth=3.
* Hard: Minimax + Alpha-Beta pruning with depth=5.

**3. AI Approach & Methodology**

**Algorithms Used**

|  |  |
| --- | --- |
| **Technique** | **Purpose** |
| Minimax | Searches possible moves to maximize AI advantage. |
| Alpha-Beta Pruning | Reduces unnecessary branches in Minimax. |
| Heuristic Evaluation | Scores board states for AI decision-making. |

**Heuristic Function Design**

The AI evaluates board positions using:

1. Immediate Wins (+∞ if AI wins, -∞ if opponent wins).

2. Potential 3-in-a-Row (high score for near-wins).

3. Center Control (higher weight for central columns).

Complexity Analysis

* + Minimax without pruning: O(bᵈ) (b = branching factor, d = depth).
  + With Alpha-Beta pruning: Reduces complexity by ~50%.

**4. Implementation Plan**

**Tech Stack**

* Language: Python
* Libraries:
* `pygame` (GUI)
* `numpy` (board state handling)

Milestones & Timeline

|  |  |
| --- | --- |
| **Week** | **Task** |
|  |  |
| 1 | Design board, implement basic game logic. |
| 2 | Develop Minimax algorithm. |
| 3 | Add Alpha-Beta pruning and heuristics. |
| 4 | Integrate GUI and final testing. |

**5. 5-10% Working Demo**

Task:

* + Display the Connect Four board.
  + Allow the human player to make a move.
  + Show AI’s random move (Easy mode).

Code Snippet (Python + Pygame)

```python

import pygame

import numpy as np

Initialize Pygame

pygame.init()

screen = pygame.display.set\_mode((700, 600))

ROWS, COLS = 6, 7

board = np.zeros((ROWS, COLS))

def draw\_board():

for row in range(ROWS):

for col in range(COLS):

pygame.draw.rect(screen, (0, 0, 255), (col\*100, row\*100+100, 100, 100))

color = (255, 255, 255) if board[row][col] == 0 else (255, 0, 0)

pygame.draw.circle(screen, color, (col\*100+50, row\*100+150), 45)

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

if event.type == pygame.MOUSEBUTTONDOWN:

col = event.pos[0] // 100

for row in reversed(range(ROWS)):

if board[row][col] == 0:

board[row][col] = 1 Human move

break

draw\_board()

pygame.display.update()

pygame.quit()

```

**6. Next Steps**

1. Implement Minimax for AI decision-making.

2. Add Alpha-Beta pruning for optimization.

3. Design difficulty levels (adjust search depth).

**7. References**

* + Russell & Norvig Artificial Intelligence: A Modern Approach (Minimax theory).
  + [Pygame Documentation](<https://www.pygame.org/docs/>)