Project Report

**Course: Artificial Intelligence**

**Project Title: AI-based Tic-Tac-Toe AI Agent**

**Submitted By: Faizan Ali (22I-2496), Zaid Vohra (22K-4195), Hamail Rehman (22K-4443)**

**Instructor: Mr. Khalid Khan**

**Submission Date: 11th May 2025**

# 1. Executive Summary

This project aims to create an AI agent to play a modified version of the conventional 5x5 Tic-Tac-Toe game, where four marks in a row are required to win. The AI uses the Minimax algorithm with heuristics, move ordering, and caching to optimize its decision-making process.

# 2. Introduction

Background:  
The original Tic-Tac-Toe game is a 3x3 grid game, where two players take turns marking a space on the grid with their respective symbols (X or O). The project aims to develop an AI capable of playing a modified 5x5 grid Tic-Tac-Toe game, with the objective of getting four in a row. The AI utilizes an optimized version of the Minimax algorithm to make strategic decisions.

Objectives of the Project:  
The goal is to develop an AI capable of playing Tic-Tac-Toe with 5x5 grids and four marks in a row to win, using Minimax with a heuristic evaluation function. Additionally, the project explores techniques like Alpha-Beta pruning, move ordering, and caching to optimize performance and decision-making.

# 3. Game Description

Original Game Rules:  
Tic-Tac-Toe is a turn-based game played on a 3x3 grid. Two players take turns marking a cell with their symbol, with the goal of placing three symbols in a row, column, or diagonal.

Innovations and Modifications:  
This version introduces a 5x5 grid, where four in a row is required to win. The game includes advanced AI decision-making, using the Minimax algorithm with heuristics for evaluation and move ordering to prioritize strategic moves.

# 4. AI Approach and Methodology

AI Techniques Used:  
The AI uses the Minimax algorithm to evaluate potential moves and select the best option. Alpha-Beta pruning is applied to improve efficiency by reducing the number of evaluated positions. Caching is used to store previously evaluated game states, and move ordering is implemented to prioritize moves that are more likely to lead to favorable outcomes.

Algorithm and Heuristic Design:  
The evaluation function assigns scores based on the number of marks in a row, column, or diagonal. Four-in-a-row is given the highest score, followed by three-in-a-row, two-in-a-row, and one-in-a-row, with block weights applied to prevent the opponent from winning.

AI Performance Evaluation:  
The AI's performance was evaluated by running test matches against human players and measuring win rates, decision times, and the overall effectiveness of its strategy.

# 5. Game Mechanics and Rules

Modified Game Rules:  
In this version, players take turns marking a space on a 5x5 grid. A player wins by placing four of their marks in a row, either horizontally, vertically, or diagonally. The game ends when a player wins or no valid moves are left.

Turn-based Mechanics:  
Players alternate turns, and the AI is designed to play as one of the players in the game. The game state is evaluated at each step using the Minimax algorithm to decide on the best move.

Winning Conditions:  
The game ends when one player achieves four marks in a row, or the board is filled without any player winning, in which case it results in a draw.

# 6. Implementation and Development

Development Process:  
The game was implemented using Python, utilizing the Pygame library for the graphical user interface (GUI). The AI was integrated with the Minimax algorithm, enhanced with heuristics for move evaluation and optimizations such as Alpha-Beta pruning and caching.

Programming Languages and Tools:  
Programming Language: Python  
Libraries: Pygame, NumPy  
Tools: GitHub for version control

Challenges Encountered:  
Challenges included optimizing the Minimax algorithm to handle larger board sizes and ensuring efficient computation of the AI's move evaluations.

# 7. Team Contributions

Faizan Ali: Responsible for AI algorithm development (Minimax, Alpha-Beta Pruning, and Heuristics).

Zaid Vohra: Handled game rule modifications, board design, and integration of the AI with the game.

Hamail Rehman: Focused on implementing the user interface, testing, and optimizing AI performance.

# 8. Results and Discussion

The AI successfully played the modified 5x5 Tic-Tac-Toe game, demonstrating a high win rate against human players. The decision-making time per move was optimized using move ordering and caching, which reduced the time complexity of the Minimax algorithm. The AI performed effectively even when playing against human players in a competitive setting.

# 9. References

List of references here:

Minimax Algorithm: https://en.wikipedia.org/wiki/Minimax

Alpha-Beta Pruning: https://en.wikipedia.org/wiki/Alpha%E2%80%93beta\_pruning

Tic Tac Toe Game Theory: https://en.wikipedia.org/wiki/Tic-tac-toe

Pygame Documentation: https://www.pygame.org/docs/