|  |
| --- |
| Instructor: Miss Mehak Mazhar |
| Project Proposal |
| Topic: Strategic AI Checkers |

Submitted by :

Muhammad Umer Qureshi 22K-4706

Adeen Ali 22K-4679

Arsalan 22K-4694

Umaima abeer 22K-4738

# PROJECT PROPOSAL

# Strategic AI Checkers: Human vs AI with Minimax and Alpha-Beta Pruning

## 1. Project Overview

### Project Topic:

This project is based on the classic game of Checkers but enhanced with an AI opponent using the Minimax algorithm with Alpha-Beta pruning. The goal is to allow a human player to compete against an intelligent AI opponent that makes optimal decisions in real-time.

### Objective:

The primary goal is to design and implement a strategic AI capable of playing Checkers using the Minimax algorithm, optimized with Alpha-Beta pruning. The system includes a graphical interface for human interaction using Pygame, promoting engagement and real-time decision-making.

## 2. Game Description

### Original Game Background:

Checkers is a two-player game played on an 8x8 board with alternating dark and light squares. Players move diagonally and capture opponent pieces by jumping over them. The objective is to eliminate all opponent pieces or block them from making a legal move.

### Innovations Introduced:

- Implemented a fully interactive GUI using Pygame for human vs AI gameplay.  
- Integrated the Minimax algorithm with Alpha-Beta pruning for optimal AI performance.  
- Enhanced visual feedback for selected pieces and legal moves.  
- Added turn-based interaction with real-time rendering of moves and captures.  
  
These innovations introduce a layer of complexity and allow players to engage with a strategic AI, rather than a fixed rule-based opponent.

## 3. AI Approach and Methodology

### AI Techniques to be Used:

- Minimax Algorithm: Core decision-making logic for the AI to evaluate possible moves.  
- Alpha-Beta Pruning: Optimizes the Minimax tree by cutting off unneeded branches, improving performance.  
- Future Scope : Potential for reinforcement learning to train the AI through self-play.

### Heuristic Design:

The AI evaluates game states based on:  
- Number of remaining pieces for each player  
- Number of kinged pieces  
- Positional advantage (e.g., control of the center)  
- Potential captures in future moves

### Complexity Analysis:

- Time Complexity: Exponential in nature (O(b^d)) where \*b\* is the branching factor and \*d\* is depth. Alpha-Beta pruning helps reduce this significantly.  
- Challenges: Balancing between search depth and real-time responsiveness for smooth user experience.

## 4. Game Rules and Mechanics

### Modified Rules:

- Standard Checkers rules apply.  
- Enhanced interaction through real-time GUI input for human players.  
- AI moves calculated using search algorithms rather than predefined strategies.

### Winning Conditions:

A player wins by eliminating all opponent pieces or blocking them from making any legal moves.

### Turn Sequence:

Players alternate turns.  
- Human inputs move via GUI.  
- AI evaluates board state and responds using Minimax + heuristics.

## 5. Implementation Plan

### Programming Language:

Python

### Libraries and Tools:

- Pygame – for rendering the game board and handling player input.  
- NumPy – for efficient board representation and computation.  
- No external AI libraries; custom implementation of algorithms.

### Milestones and Timeline:

- Week 1–2: Finalize Checkers game logic and GUI  
- Week 3–4: Implement and test Minimax with heuristics  
- Week 5–6: Integrate AI with GUI and optimize for performance  
- Week 7: Complete AI tuning and polish user interaction  
- Week 8: Final testing, debugging, and report preparation

## 6. References

- https://en.wikipedia.org/wiki/Checkers  
- https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/  
- https://www.pygame.org/docs/  
- GitHub repo: https://github.com/Gualor/checkers-minimax