# **Project Proposal**

**Artificial Intelligence**  
 **Section: F**

### **Group Members**

* **22k-4596 Waleed Haroon**
* **Ahmed Iltaf**

### **Introduction**

The goal of this project is to create an intelligent agent capable of playing the game of **Checkers** at a competitive level. This agent will leverage advanced **Artificial Intelligence** techniques, combining **Reinforcement Learning** (specifically Q-learning) with traditional **Alpha-Beta Pruning** algorithms to make optimal strategic decisions. The system will be paired with an intuitive **Pygame-based interactive interface**, allowing users to play against the AI while observing its decision-making process.

### **Existing Systems**

There are several existing implementations of AI-driven checkers games, each offering different levels of performance and user experience:

* **Simple-Checkers-AI**: A basic Python-based implementation utilizing Pygame and the easyAI library. While functional, it lacks scalability and experiences occasional performance issues.
* **Checkers-AI**: Built using HTML, CSS, and JavaScript, this project incorporates AI algorithms for gameplay but has limitations in terms of decision-making depth and efficiency.
* **Checkers-Reinforcement-Learning**: Focuses on reinforcement learning to train the AI. It includes tools for both training and evaluating AI performance but lacks a polished user interface.

These systems provide valuable foundational ideas but fall short in scalability, efficiency, and user engagement.

### **Problem Statement**

Current AI-driven checkers implementations often face the following challenges:

* **Limited Learning Capabilities**: Many systems rely on predefined strategies and algorithms, limiting the AI's ability to improve and adapt based on gameplay experiences.
* **Performance Constraints**: Some implementations experience significant delays, especially when exploring deeper search depths in decision-making algorithms.
* **User Interface Limitations**: Existing projects often lack intuitive and engaging interfaces, diminishing the overall user experience.

### **Proposed Solution**

Our project intends to address the aforementioned issues by:

* **Integrating Reinforcement Learning**: We will implement **Q-learning** to allow the AI to learn and adapt its strategies through self-play, continuously improving over time.
* **Optimizing Alpha-Beta Pruning**: To improve decision-making speed and efficiency, we will apply depth-limited search with alpha-beta pruning, reducing computational complexity.
* **Developing a User-Friendly Interface**: Using **Pygame**, we will design a visually appealing and responsive UI to facilitate smooth interaction between the human player and the AI.

### **Salient Features**

The project will provide the following key features:

* **Adaptive AI Opponent**: The AI will learn from gameplay, improving its strategy and becoming a more competitive opponent over time.
* **Efficient Decision-Making**: We will implement optimized search algorithms to ensure that the AI can make quick and strategic moves.
* **Engaging User Interface**: A visually appealing and intuitive interface will enhance user interaction, making the game more enjoyable.
* **Performance Analytics**: Tools for analyzing game performance will offer insights into both AI decision-making and user gameplay patterns.

### **Tools & Technologies**

* **Programming Language**: **Python** will be used for backend logic and AI algorithms.
* **Frameworks**: **Pygame** for the game interface, and **TensorFlow** or **PyTorch** for reinforcement learning models.
* **Operating System**: Cross-platform compatibility, supporting both **Windows** and **Linux** environments.

### **References**

* [Python Checkers AI by TechWithTim](https://github.com/techwithtim/Python-Checkers-AI)
* [Checkers AI by Sramakrishnan247](https://github.com/sramakrishnan247/Checkers-AI)