**AIES Mini Project Report**

**Subject – Artificial Intelligence and Expert Systems**

**Prepared by:**

**Group members – PC-35 Rohan Vayase**

**PC-36 Navjot Singh Sidhu**

**PC-38 Ujjval Kulkarni**

**PC-39 Abhishek Singh**

**Project Report: Chess Bot using Minimax Algorithm**

**Abstract**:

Artificial Intelligence (AI) and Expert Systems play a crucial role in developing intelligent agents capable of making strategic decisions. In this mini project, we delve into the realm of chess programming, implementing a Chess Bot using the Minimax algorithm in Python. Notably, the project incorporates the p5.js library to enhance the user interface and create an interactive and visually appealing experience. This report provides a comprehensive overview of the design, implementation, challenges faced, and outcomes achieved during the development of our Chess Bot.

**1.** **Introduction**:

Chess, as a game of strategy and foresight, has long been a testing ground for AI algorithms. The Minimax algorithm, a fundamental approach in two-player, aids in decision-making by searching through possible moves. The integration of the p5.js library in Python adds a dynamic visual dimension to our Chess Bot, creating a seamless and engaging user experience.

**2. Literature Review:**

Prior research has extensively explored the application of AI algorithms in chess, with Minimax being a prominent choice for decision-making. The use of p5.js in Python for interactive graphics and visualization is gaining popularity due its versatility and ease of use. Our project aims to build on these foundations, providing a unique combination of intelligent gameplay and an immersive user interface.

**3.** **Methodology:**

**3.1 Chess Bot Architecture:**

Our Chess Bot follows a modular architecture, consisting of distinct components for game logic, AI decision-making, and user interface. The minimax algorithm forms the core of the decision-making module, evaluating possible moves and selecting the optimal one base on a scoring mechanism.

**3.2 Implementation Details:**

The Chess Bot is implementing in Python, utilizing the object-oriented paradigm to model game entities. Integration with the p5.js library facilitates real-time visualization, allowing users to interact with the chessboard and pieces. The Minimax algorithm, enhanced with alpha-beta pruning for efficiency, enables intelligent decision-making.

**3.3** **User Interface Design:**

The p5.js library provides a dynamic canvas for the chessboard and pieces, allowing for smooth user interface. Visual cues, such as highlighting legal moves and animating piece movements, enhance the user experience. The integration of p5.js in a Python environment is achieved through appropriate libraries and wrappers.

**4. Results and Evaluation:**

**4.1** **Performance Metrics:**

Evaluation of the Chess Bot includes metrics such as win rate against human players, average move time, and adaptability to different playing styles. The visual appeal and responsiveness of the user interface are also considered in the evaluation.

**4.2** **Planning:**

* **Integration of p5.js in Python:** Bridging the gap between the p5.js library (commonly used with JavaScript) and Python presented challenges. We overcame by utilizing existing wrappers and ensuring compatibility.
* **Optimizing Visual Feedback:** Balancing the need for engaging visual elements with maintaining optimal performance posed challenges. Techniques such as asynchronous updates were implemented to address this.
* **User Interface and User Experience (UI/UX):** Ensuring an intuitive and user-friendly interface required iterative design and testing. Incorporating user feedback was crucial in refining the UI/UX.

**5. Conclusion:**

In conclusion, our project successfully implemented a Chess Bot using the Minimax algorithm and the p5.js library in Python. The integration of intelligent decision-making with an interactive and visually appealing user interface provides a holistic gaming experience. Challenges encountered during the implementation process highlight the complexity of creating AI agents for strategic games. Future work may involve further optimization, exploring alternative visualization libraries, and incorporating additional AI techniques for enhanced gameplay.

**6. Acknowledgments:**

We express our gratitude to our project supervisor for guidance and support. Additionally, we acknowledge the open-source contributions from the chess programming community and the developers behind the p5.js library for making our project possible.