**Chess Engine**

A Synopsis Submitted

in Partial Fulfilment of the Requirements

for the Degree of

**BACHELOR OF TECHNOLOGY**

in

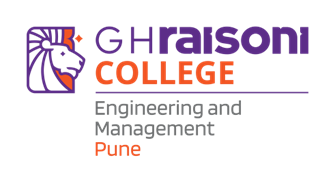
# Department of Artificial Intelligence & Machine Learning

by

# Atharva Bansod A20

# Devesh Mhaske A26

# Devashish Murai A27



**G H Raisoni College of Engineering and Management**

**Wagholi, Pune**

**Department of Artificial Intelligence and Artificial Intelligence & Machine Learning**

**(An Empowered Autonomous Institute (NAAC A+ Grade) Affiliated to SPPU, Pune)**

# June, 2024

**Introduction**

The beginner-level chess engine project aims to create an interactive chess application where users can play against a computer opponent. Chess is a strategic game with centuries of history, and this project seeks to translate this complexity into a user-friendly, responsive web-based platform. Leveraging modern technologies such as Python, React.js, and Flask, the system will provide a seamless chess-playing experience. The significance of the project lies in its educational value for both novice chess players looking to improve and developers interested in AI-based game logic.

**Objectives**

The main goals of this project are:

1. To create a chess engine that allows users to play chess against an AI opponent.

2. To implement basic chess rules and offer game features such as move validation, undo options, and detection of end-game conditions.

3. To develop a responsive and intuitive web-based GUI for easy user interaction.

4. To provide different levels of AI difficulty to challenge users of varying skill levels.

5. To ensure the game is accessible across different operating systems (Windows, macOS, and Linux).

6. To maintain a modular architecture that allows for easy updates, including scalability to multiplayer modes.

**Literature Review**

Many chess engines have been developed over the years, such as Stockfish and AlphaZero, each employing different algorithms like minimax and Monte Carlo Tree Search (MCTS). Existing engines often focus on highly complex AI opponents and advanced user interfaces. However, there is a gap in beginner-level engines designed for casual players that provide an intuitive experience with different AI difficulty levels. This project seeks to fill that gap by offering an educational, easy-to-use chess engine with a simplified AI system. Unlike more advanced engines, this project prioritizes usability and scalability, making it suitable for casual users while also being a learning tool for developers interested in chess AI.

**Methodology**

The project will be developed using Python for the backend, React.js for the front-end, and Flask for communication between the client and server. The chess logic will be managed using the `Self Build Chess Logic Package` library, while the AI moves will be implemented using algorithms such as Minimax or NegaMax with alpha-beta pruning for optimal performance at different levels. The key steps in the methodology include:

1. **Game Initialization**: Implement a user interface that allows starting a new chess game and displaying an 8x8 board.

2. **User Interaction**: Enable piece movement with drag-and-drop functionality.

3. **Move Validation**: Incorporate chess rules using `Self Build Chess Logic Package` to ensure legality of user moves.

4. **AI Moves**: Develop AI functionality using depth-limited search algorithms, ensuring responsiveness.

5. **Game State Management**: Maintain the current game state and allow users to undo moves.

6. **Multiplayer**: Plan and implement multiplayer functionality for scalability in later phases.

7. **Testing and Validation**: Conduct unit tests for game logic and integrate user feedback for GUI usability.

**Tools and Technologies**

- **Frontend**: React.js for the graphical user interface.

- **Backend**: Python with Flask for handling logic and state management.

- **Chess Logic**: `Self Build Chess Logic Package` for move validation and game rules.

- **AI Engine**: Custom depth-based search algorithms for generating AI moves.

- **Data Storage**: JSON for maintaining game state and history, with an option for database integration in the multiplayer version.

**Project Plan and Timeline**

The project will be developed over a six-month period, divided into the following phases:

1. **Phase 1** - Requirement Analysis and Design (0.5 month)

- Detailed requirement gathering and finalizing architecture.

- Design of GUI wireframes and AI move planning.

2. **Phase 2** - Frontend Development (0.5 months)

- Building the basic React.js components for displaying the chessboard, pieces, and user interactions.

3. **Phase 3** - Backend Development (0.5 month)

- Setting up the Flask server, game logic, and `Self Build Chess Logic Package` integration.

- Implementing basic AI functionality for beginner difficulty.

4. **Phase 4** - AI Implementation and Scaling (1 month)

- Developing AI difficulty levels using depth-limited search algorithms.

- Ensuring responsive AI moves across different difficulty settings.

5. **Phase 5** - Integration and Testing (0.5 month)

- Integrating frontend with backend for seamless gameplay.

- Conducting rigorous testing for move validation, AI accuracy, and performance.

6. **Phase 6** - Final Adjustments and Deployment (0.5 month)

- Implementing final user feedback.

- Preparing for deployment and documentation.

**Expected Outcomes**

The primary deliverable is a functional chess engine that allows users to play against an AI opponent through a web-based interface. The system will be modular, allowing for future scalability, such as multiplayer functionality. The AI will provide a range of difficulty levels to cater to beginners, with room for optimization in subsequent iterations. The project aims to deliver a stable, user-friendly product with the potential to be expanded into a multiplayer platform.

The expected impacts include:

- Helping beginners improve their chess skills through practice against a well-designed AI.

- Offering developers an example of how to integrate game logic, AI, and web-based GUIs using Python and JavaScript.

- Providing a modular and maintainable codebase for future updates and scaling.

**References**

1. Chess Rules: (https://www.chess.com/learn-how-to-play-chess)

2. Python Chess Library Documentation: https://Self Build Chess Logic Package.readthedocs.io/en/latest/)

3. Stockfish: (https://stockfishchess.org/)

4. AlphaZero: Silver, D., Schrittwieser, J., et al. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm."