# Project Proposal: Chess960 (Fischer Random Chess) Implementation

**Submitted By**: Mustafa Ahmed (22K-5056)
**Course**: AI
**Instructor**: Ms. Alishba Subhani
**Submission Date**: May 11, 2025

## 1. Project Overview

**Project Topic**:
This project implements Chess960, also known as Fischer Random Chess, a chess variant with 960 possible starting positions for the back-rank pieces. The implementation includes a graphical user interface (GUI) using Pygame, an AI opponent powered by the Negamax algorithm with Alpha-Beta pruning, sound effects, and game data analysis to evaluate AI performance.

**Objective**:
The main goal is to develop a fully functional Chess960 game with a strategic AI opponent that challenges human players. The project demonstrates advanced AI techniques (Negamax with Alpha-Beta pruning), robust game logic for Chess960-specific rules (e.g., randomized setups, flexible castling), and a user-friendly GUI. Additionally, it analyzes AI performance through statistical reports, showcasing skills in data processing and visualization.

## 2. Game Description

**Original Game Background**:
Chess960 is a variant of standard chess invented by Bobby Fischer. The game uses the same 8x8 board and pieces, but the back-rank pieces (king, queen, rooks, knights, bishops) are randomized at the start, subject to constraints: bishops must be on opposite-colored squares, and the king must be between the rooks. Pawns remain on their standard ranks (2 for Black, 7 for White). The rules for movement, capturing, check, checkmate, and stalemate are identical to standard chess, except castling is adapted to accommodate the randomized positions.

**Innovations Introduced**:

- **Randomized Starting Positions**: Implemented a generator that produces valid Chess960 setups, ensuring compliance with the variant's constraints (bishops on opposite colors, king between rooks). This increases replayability and strategic depth by eliminating reliance on memorized openings.
- **Customizable GUI Themes**: Five color themes (blue, bw, green, wood, purple) for light/dark squares, highlights, and last-move indicators, enhancing user experience.
- **AI Opponent with Chess960 Optimizations**: The AI uses Negamax with Alpha-Beta pruning, tailored with heuristics like a bishop pair bonus (+0.5 points) and a penalty for undeveloped rooks (-0.3 points after move 10).
- **Game Analysis**: Saves game data (e.g., outcome, move count, AI decision times) to a CSV file and generates a performance report (e.g., AI win rate, average decision time), demonstrating data analysis skills.
- **Sound Effects and Animations**: Includes sound effects for moves, captures, checks, and game end, along with smooth piece movement animations, improving immersion.

These innovations make the game more engaging, strategically complex, and analytically rich compared to standard chess implementations. The randomized setups challenge players to rely on tactical understanding, while the AI and analysis features provide insights into gameplay dynamics.

## 3. AI Approach and Methodology

**AI Techniques to be Used**:

- **Minimax Algorithm (Negamax Variant)**: Implemented as the core AI decision-making algorithm, evaluating game states to a depth of 3 to balance performance and strategy. The Negamax variant simplifies the code by unifying the perspective of both players.
- **Alpha-Beta Pruning**: Applied to reduce the number of nodes evaluated in the game tree, significantly improving AI efficiency without sacrificing accuracy.
- **Heuristic Design**:
    - **Material Evaluation**: Assigns standard piece values (King: 20,000, Queen: 9, Rook: 5, Bishop: 3.25, Knight: 3, Pawn: 1) to prioritize material advantage.
    - **Chess960-Specific Bonuses**: Adds a +0.5-point bonus for a bishop pair, reflecting their increased value in randomized setups, and a -0.3-point penalty for undeveloped rooks after move 10 to encourage early development.
    - **Checkmate and Stalemate**: Assigns a high score (±20,001) for checkmate and 0 for stalemate to prioritize decisive outcomes.
- **Move Randomization**: Randomizes the order of evaluated moves to make the AI less predictable, enhancing replayability.

**Complexity Analysis**:

- **Time Complexity**: The Negamax algorithm with Alpha-Beta pruning has a worst-case time complexity of $O(b^{d})$, where `b` is the branching factor (average ~35 in chess) and `d` is the depth (3). Alpha-Beta pruning reduces the effective branching factor, achieving $O(b^{(d/2)})$ in optimal cases. For depth 3, this results in evaluating thousands of positions per move, manageable on modern hardware (decision times ~0.1–1 second).
- **Challenges**: Implementing Chess960-specific castling required custom logic to handle variable king and rook positions, increasing complexity. Optimizing the AI to balance strength and speed was challenging, addressed by limiting depth to 3 and using efficient pruning.

## 4. Game Rules and Mechanics

**Modified Rules**:

- **Randomized Starting Positions**: The back-rank pieces are shuffled to create one of 960 possible setups, ensuring bishops are on opposite-colored squares and the king is between the rooks. Pawns remain on ranks 2 (Black) and 7 (White).
- **Castling**: Allowed if the king and rook have not moved, the path between them is clear, and the king does not move through or land in check. The king moves two squares toward the rook, and the rook moves to the king's opposite side, adapting to the initial positions.
- **Other Rules**: Standard chess rules apply for pawn movement (including en passant), piece captures, check, checkmate, and stalemate (including the 50-move rule for draws).

**Winning Conditions**:

- **Checkmate**: A player wins by placing the opponent's king in checkmate, where the king cannot escape check.
- **Stalemate**: The game is a draw if a player has no legal moves and is not in check.
- **50-Move Rule**: A draw is declared if 50 consecutive moves occur without a capture or pawn move.

**Turn Sequence**:

- Players alternate turns, with White moving first. A turn consists of moving one piece to a legal square or castling. The AI takes the opponent's turn in human vs. computer mode, with a brief delay (200ms) to simulate thinking.

## 5. Implementation Plan

**Programming Language**: Python 3.8+
**Libraries and Tools**:

- **Pygame (2.6.1)**: Used for the GUI, rendering the 8x8 board, piece animations, and sound playback (move, capture, check, game end).
- **NumPy (2.2.5)**: Manages the board as a 2D array of `Square` objects for efficient state updates.
- **Pandas (2.2.3)**: Processes game data in `game_data.csv` to generate statistical reports (e.g., AI win rate, decision times).
- **Tkinter**: Standard Python library for the game menu, allowing color and theme selection.

**Milestones and Timeline**:

- **Week 1-2: Game Design and Rule Finalization**
  - Defined Chess960 rules and constraints for starting positions.
  - Designed the GUI layout (board, sidebar, move history) and selected themes.
- **Week 3-4: AI Strategy Development**
  - Implemented Negamax with Alpha-Beta pruning and Chess960-specific heuristics.
  - Tested AI for move validity and strategic depth.
- **Week 5-6: Coding and Testing Game Mechanics**
  - Developed board representation (`chess_board.py`), piece logic (`chess_pieces.py`), and game state management (`chess_engine.py`).
  - Implemented Chess960 castling, en passant, and pawn promotion.
  - Created the Pygame GUI with animations and sound effects (`chess_main.py`).
- **Week 7: AI Integration and Testing**
  - Integrated the AI into the game loop, ensuring seamless human vs. computer play.
  - Tested edge cases (e.g., castling under check, stalemate conditions).
- **Week 8: Final Testing and Report Preparation**
  - Conducted gameplay testing to verify rules and AI performance.
  - Developed game data analysis (`analyze_game_data.py`) and generated reports.
  - Prepared this proposal and documentation.

## 6. References

- Chess960 Rules: https://www.chess.com/terms/fischer-random-chess
- Pygame Documentation: https://www.pygame.org/docs/
- NumPy Documentation: https://numpy.org/doc/
- Pandas Documentation: https://pandas.pydata.org/docs/
- Negamax Algorithm: https://www.chessprogramming.org/Negamax

- Alpha-Beta Pruning: https://www.chessprogramming.org/Alpha-Beta
- Fischer, Bobby. *My 60 Memorable Games*. Batsford, 2008 (for chess variant inspiration).