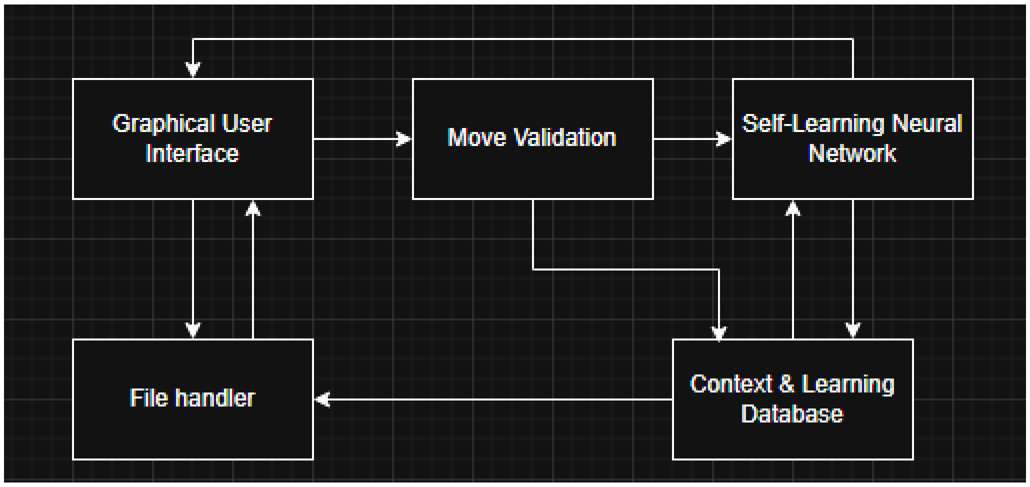
5.1 Introduction

The Chess Learning Engine (CLE) is a windows-based application designed to provide a competitive environment for players to engage with a self-learning artificial intelligence. The system components include a graphical user interface for move input and board visualization, a move validation engine, and a neural-network-based chess engine that improves its strategic evaluation through reinforcement learning.

The remainder of this document is structured as follows. Section 5.2 contains the functional requirements defining core gameplay and engine behavior; Section 5.3 contains performance requirements regarding processing speed and memory; and Section 5.4 details the environment requirements for development and execution.



5.1.2 Component Breakdown

Based on the architectural design, the system is separated into the following components:

- **Graphical User Interface (GUI)**: Manages all visual output and captures user inputs.
- **Move Validation**: Enforces the legal rules of chess for every submitted move.
- **Self-Learning Neural Network**: The core engine responsible for generating moves and reinforcement learning.
- **Context & Learning Database**: stores the engine's learned weights and historical game data.
- **File Handler**: Manages and stores current game state and outcome.

5.2 Functional Requirements

The functional requirements describe what the completed system will do, itemized into provable statement.

5.2.1 Gameplay Interface (GUI)

The GUI provides the primary interaction layer for the player.

- **5.2.1.1** The GUI shall display a standard 8x8 grid representing the chess board.
- **5.2.1.2** The GUI shall provide interactable chess pieces on the board.
- **5.2.1.3** The GUI shall accept mouse drag-and-drop inputs to initiate piece movement.
- **5.2.1.4** The GUI shall highlight legal move destinations when a piece is selected.
- **5.2.1.5** The GUI shall provide a menu system to load, save and start a new game.

5.2.2 Move Validation Engine

This component acts as the logic gate for all gameplay actions.

- **5.2.2.1** The system shall verify that every user-initiated move adheres to standard FIDE chess rules.
- **5.2.2.2** The system shall update the internal game state only after a move is validated as legal.
- **5.2.2.3** The system shall detect and signal end-game conditions, including checkmate, stalemate, and the fifty-move rule.

5.2.3 Self-Learning Neural Network

The engine calculates optimal responses and refines its strategy over time.

- **5.2.3.1** The engine shall generate a legal counter-move for any valid player move.
- **5.2.3.2** The engine shall update its internal evaluation weights at the conclusion of every game.

5.2.4 Database

- **5.2.4.1** The Database shall store the neural network's weight layers to prevent loss of learning progress.
- **5.2.4.2** The Database shall store and provide context for each decision making process.

5.2.5 File Handler

These components ensure data persistence and integrity.

- **5.2.5.1** The File Handler shall save the current board state to a local file in PGN (Portable Game Notation) format.
- **5.2.5.2** The File Handler shall load previously saved PGN files to resume a game.
- **5.2.5.3** The system shall auto-save the game state every five minutes of active play.

5.3 Performance Requirements

Performance requirements specify how the project must operate to be effective and grouped by component.

5.3.1 Graphical User Interface (GUI)

- **5.3.1.1** The GUI shall refresh the board display in less than 100 milliseconds following a validated move.
- **5.3.1.2** The system shall take no longer than 1 second to start a new game.
- **5.3.1.3** The system shall take no longer than 3 seconds to load a saved game.

5.3.2 Self Learning Engine

- **5.3.2.1** The engine shall return a calculated move within 10 seconds of the player's turn completion during standard matches.

5.3.3 Move Validation Engine

- 5.3.3.1 The engine shall return possible moves when a chess piece is selected in less than 100 milliseconds.

5.4 Environment Requirements

This section lists the resources needed for development and execution.

5.4.1 Hardware Requirements

- **Processor:** Quad-core 2.5 GHz or higher
- **Ram:** 16GB minimum to optimally support neural network
- **Storage:** 5GB available disk space.

5.4.2 Software Requirements

- **Operating System:** Windows 10/11
- **Programming Language:** Python 3.12
- **Library:** TensorFlow, Tkinter
- **IDE:** VS code
- **Version Control:** Git and GitHub