|  |  |
| --- | --- |
| The British College Kathmandu | Top IT ...  Documentation  Chess Game | Link github:  Sweta Kumari Das  LEVEL 4 |

Chess Game Documentation

# 1. Overview

This project is a console-based 2D chess game implemented in C++. It features:  
- A human player (White) vs a basic AI (Black)  
- Unicode and ANSI-colored board rendering  
- Move validation, check/checkmate detection  
- Basic AI logic with piece capture prioritization

# 2. Technologies Used

- Language: C++  
- Libraries:  
 - Standard Library (iostream, vector, cstdlib, etc.)  
 - chrono, thread: Used for AI "thinking delay"  
- No external dependencies: The entire project is console-based and self-contained

# 3. Game Features

- Full move validation for all standard chess pieces  
- AI uses a simple scoring system for choosing moves  
- Check and checkmate detection  
- Colored and symbolic board display using ANSI + Unicode  
- Move input via algebraic notation (e.g., e2e4)  
- Resignation and game exit options

# 4. How the AI Works

- Generates all legal moves for the AI  
- Scores moves based on captured piece value (e.g., Queen > Knight)  
- Picks randomly among best-scoring moves  
- Includes a delay (AI\_THINKING\_MS) for realism

# 5. Project Structure

Everything is implemented in a single file: main.cpp, including:

- ChessGame class: Core game logic  
- Move struct: Represents a move  
- main(): Initializes and starts the game loop

# 6. Key Classes and Methods

ChessGame

- printBoard(): Draws the board in terminal  
- makeMove(): Executes a move and updates board state  
- isMoveValid(): Checks if a move is legal  
- isKingInCheck(): Determines if a player's king is under threat  
- makeAIMove(): AI logic for move selection  
- generateValidMoves(): Gathers all legal moves for current player  
- play(): Runs the game loop

# 7. Running the Game

Compile

g++ main.cpp -o chess\_game

Run

./chess\_game

# 8. Limitations

- No castling, en passant, or pawn promotion  
- AI logic is basic (not Stockfish or Minimax)  
- Single-file project (no modular separation)

# 9. Future Enhancements

- Add castling, en passant, and promotion  
- Introduce a GUI using SFML  
- Integrate Stockfish engine for stronger AI  
- Split code into multiple files (OOP modular design)