**Chess Game with AI - Coursework Report**

**Created by Eimantas Putna EIF-24**

This is either a two player chess game on the same computer or ai vs human chess game built with Python and Pygame. The AI uses the NegaMax algorithm for move evaluation. The application focuses on core chess mechanics, rule enforcement, move generation, and basic animation. The sprites were inspired by fantasy chess pieces.

How to run the program

1. Make sure Python 3 is installed.
2. Install Pycharm: pip install pygame
3. Run the main program: python ChessMain.py

How to use the program

* Use the mouse to select and move pieces.
* Press Z to undo moves (undoes both human and AI moves).
* Press R to reset the game.

**2. Body/Analysis**

All standard chess rules are built into the GameState class found in ChessEngine.py. This includes generating only legal moves and handling special cases like castling, en passant, and pawn promotion.

The computer opponent is built using the Strategy design pattern. There's a general MoveStrategy interface and a specific implementation called NegaMaxStrategy that makes the AI work.

Players use the mouse to select and move pieces. The interface provides visual feedback and includes simple animations to show moves clearly.

There's a feature that lets you undo the last move—reversing both your move and the AI's, so you can rethink your strategy.

**Code Snippet (Strategy Pattern - Abstraction & Polymorphism):**

**Paveikslėlis, kuriame yra tekstas, ekrano kopija, meniu

Dirbtinio intelekto sugeneruotas turinys gali būti neteisingas.**

**OOP Pillars:**

* **Abstraction:** MoveStrategy exposes only findMove().
* **Polymorphism:** ChessMain.py uses different strategies via the same interface.
* **Inheritance:** NegaMaxStrategy inherits from MoveStrategy.
* **Encapsulation:** GameState encapsulates board data and operations like makeMove() and getValidMoves().

**3. Results and Summary**

* The core game logic and rules are fully functional.
* The AI performs reasonably well using the NegaMax strategy with material scoring.
* Undo logic required handling both player‘s turns properly.
* Pygame's rendering and animation logic were implemented cleanly.
* Design patterns and OOP principles improved structure and readability.

**4. Conclusions**

* This project successfully delivered a functioning chess game with AI support.
* The AI logic correctly evaluates and plays moves based on a depth-limited NegaMax algorithm.
* The use of object-oriented programming and the Strategy pattern made the codebase maintainable and extensible.
* In the future, the AI can be upgraded with Minimax + Alpha-Beta pruning or machine learning for better move prediction.
* Multiplayer networking, move timers, or persistent game state saving could also be integrated to expand functionality.