Backend-Driven Chess Web Application with Dynamic Backend Processing

Overview

This is a web application designed for a two-player chess game. The focus of this application is on the backend processing, with a simple static front-end page for user interaction. The front-end interface allows players to send their chess moves to the server, where the game logic is processed, and the updated game state is rendered, stored and sent back to the client-side.

Components

1. **Client-Side Interface**:
    - **index.html**: The single static web page that serves as the interface for players to interact with the game. It sends chess piece moves to the server and displays the updated game state received from the backend.
2. **Server-Side Interface**:
    - **app.py**: A Python server file that acts as the bridge between the client-side interface and the backend. It handles requests from index.html, processes them, and sends back the rendered HTML Code with the updated game state.
3. **Backend Components**:
    - **control.py**: The central control file that manages the game's data flow and processing. It receives data from app.py, fetches necessary information, coordinates with other backend components, and updates the game state.
    - **database_access.py**: A file that simulates a DBMS, responsible for accessing and manipulating game-related data stored in database.txt.
    - **chess.py**: The core logic file containing the chess game rules and mechanics. It processes the moves received from control.py and returns the updated game state.
    - **database.txt**: A text file that serves as the database, storing game-related information such as player ID, game state, and other relevant data.

Key Features

- **Backend-Loaded Architecture**: The application relies heavily on backend processing to handle game logic and state management.
- **Simple Front-End Interface**: The single static page (index.html) serves as the user interface, making the application lightweight and easy to use.

- **Central Control**: control.py control and coordinates the backend operations, ensuring smooth data flow and processing between components.
- **Database Management**: database_access.py handles data access, storing and manipulation, ensuring efficient retrieval and updating of game-related data from database.txt.

Summary

This two-player chess web application leverages a backend-loaded architecture. The focus on backend processing ensures that game logic is separately handled, while the simple front-end interface offers a seamless and light-weight user experience. The modular design of the backend components allows for easy maintenance and scalability.