Chess Game Project Documentation

1. Project Overview
This project is a chess game implemented using React with Vite as the build tool. The main objective is to create an interactive and visually appealing chess game that tracks scores, allows two players to play against each other on a single device, and provides feedback on game status. This documentation explains the key aspects of the implementation, including game logic, user interactions, and challenges encountered during development.

2. Game Logic

The core of the game logic revolves around handling chessboard rendering, movement validation, turn-based gameplay, and score tracking for captured pieces. The following sections describe the main components and functions used in the game.

Components and State Management
The game is divided into multiple components:
- App.jsx: The main component that manages the overall state of the game.
- ChessBoard.jsx*: Handles the board setup, piece placement, and move logic.
- ChessPiece.jsx: Displays individual pieces with Unicode symbols based on type and color.
- GameUpdates.jsx: Displays player information, current turn, and score updates.

Key Functions

1. handleMove: This function handles player moves and validates if the move is allowed for the chosen piece. It also checks if a piece is captured and, if so, triggers score updates.

2. handlePieceCapture: Called within `handleMove` when a piece is captured. This function updates the score for the capturing player by identifying the current turn's player and increasing their score in the game state.

3. setCurrentTurn: Toggles the turn between the two players after each move, ensuring that players alternate turns correctly.

Score Tracking

The `handlePieceCapture` function in `App.jsx` updates player scores in the `GameUpdates` component. Each captured piece increases the capturing player's score by 1, which is reflected in the UI in real time.

3. User Interactions

The user interface is designed to be intuitive and responsive, ensuring an enjoyable experience across devices.

Interface Components

1. ChessBoard: Displays an 8x8 grid where users can interact with each square to make moves. Pieces are displayed as Unicode characters for simplicity and styled based on their color.

2. GameUpdates: Provides live information on player scores, the current player’s turn, and highlights the active player’s box for clarity.

3. Turn Indicators: Each player box in `GameUpdates` highlights the current turn with a glow effect. This makes it clear which player is to make the next move.

Interactivity
1. Piece Selection and Movement: Users can select a piece and move it to an eligible square. Movement validation is basic, with a future enhancement planned for implementing specific piece rules.

2. Score Update on Capture: When a player captures an opponent's piece, their score immediately updates in `GameUpdates`, providing visual feedback for gameplay progression.

3. Responsive Design: The layout adjusts for smaller screens by stacking components vertically, ensuring usability on mobile devices.

4. Challenges Faced and Solutions

During development, several challenges arose, including layout adjustments, turn management, and score updates. Below are the main challenges and solutions implemented:

Challenge 1: Board Layout and Responsiveness
Initially, the board and `GameUpdates` container would overlap or go out of view on certain screen sizes.

Challenge 2: Score Tracking on Piece Capture
Implementing score updates when a piece was captured required a system to identify the capturing player and dynamically update their score.

Challenge 3: Turn Management and Player Indication
It was necessary to alternate turns correctly and visually indicate the active player.

Challenge 4: Movement Validation
Enforcing movement rules for each piece type is complex and requires a detailed algorithm.