Chess Trainer App

Design Architecture

Overview

- The app follows a simple **Flutter + Services + Models** structure.
- No full dependency injection implemented yet.
- Main components:
  - **Models** → represent data
  - **Services** → perform analysis & detection
  - **Widget** → renders UI and orchestrates the workflow

Models

1️⃣ EngineAnalysis

- Represents the result of a Stockfish evaluation.
- Contains:
  - `List<PvLine>` → principal variations.
- Each `PvLine` contains:
  - Moves sequence (`pv`)
  - Evaluation (`score` , `scoreType`)
  - Depth and multipv

2️⃣ Motif

- Represents motifs in a position (FEN).
- Contains:
  - Pawn structures
  - Pawn islands
  - Queenside majority
  - Endgame type
- Returned by `MotifDetector.detectMotifs()`

Services

1️⃣ StockfishHelper

- Abstraction of **multistockfish** (Stockfish in Dart).
- Responsible for:
  - Analyzing FEN positions (`analyzeFen(String fen, {depth=3, pv=3})`)
  - Returning an `EngineAnalysis` object
- **Current limitations:**
  - No dependency injection
  - Default depth/pv is low for fast testing

2️⃣ MotifDetector

- Takes an `EngineAnalysis` object.
- Detects motifs in each PV line.
- Returns:

```
Map<int, List<Motif>>
```

Widget

1️⃣ ChessGUI

- Stateful widget that orchestrates everything:
- Accepts a FEN string (single position).
- Calls StockfishHelper.analyzeFen().
- Runs MotifDetector.detectMotifs().
- Stores results in _analysis and _motifResults.

- Renders:
  - Chessboard
  - PV lines with evaluations
  - Motif cards
  - Navigation (next/prev)

Notes / Future Improvements

- Dependency Injection:
  - Currently, StockfishHelper and MotifDetector are instantiated directly.
  - DI could improve testability and flexibility.
- PGN support:
  - Currently disabled; could allow multiple FENs in future.
- ViewModel migration:
  - ChessGUI currently handles all state.
  - Could move _analysis, _motifResults, and navigation logic to a ViewModel.