Formalized App Flow 2/9/2026

Below is a formalized and structured description of how the application's pages, variables and inputs communicate with one another throughout gameplay and analysis.

App:

- ➢ Home Page
- ➢ Game Page
- ➢ Analysis Page

Game Page:

- ➢ Input (Video)
- ➢ Store:
  - ○ List of moves made in chess notation
  - ○ 2 ints for time remaining for each player
  - ○ Current board in stored in string in FEN
- ➢ Move detection service
- ➢ Eval of current board through UCI engine

Move Detection:

- ➢ Receives frames from video input
- ➢ Detects piece movement by comparing the video input board to the current board (in FEN)
- ➢ Validate moves through chess.js
- ➢ Update current board if we get a valid move returned through chess.js

Video Input:

- ➢ Live feed from phone camera
- ➢ Captures frames at intervals (undecided how often)
- ➢ Sends frames to move detection

Chess.js (Rules Engine)

- ➢ Validates moves
- ➢ Updates internal board state
- ➢ Generates FEN for current board after move

UCI Engine (Stockfish)

- ➢ Receives FEN from current board
- ➢ Returns current eval during game
  - ○ Need to decide how long/at what depth we would like to evaluate, as I am assuming that while sending this to the UCI engine, we will temporarily increase

CPU usage, potentially slowing down the app or affecting other tasks. Evaluating at higher depths will give more accurate evaluations but will consume more processing power and may introduce lag, whereas shallow evaluations are faster but less precise. Therefore, we need to balance performance and accuracy when choosing the depth or duration of engine analysis during gameplay (later problem)

- ➢ Returns best/key moves throughout the game as well as the eval of each position in analysis page