**Overall Workflow for videoAndSidebar.tsx**

The VideoAndSidebar component is a top-level React component that combines the **video processing functionality** with an interactive **sidebar** for different modes, such as “record”, “play”, “upload”, and “broadcast”. It manages video detection, state synchronization, and UI rendering.

**1. Import Statements**

The file imports all the necessary components, utilities, hooks, and constants used in the component.

import { useRef, useState, useEffect } from "react"; *// React hooks for managing state and references*

import Video from "../common/video"; *// Core video processing component*

import { useOutletContext } from "react-router-dom"; *// React Router context for passing shared data*

import { useDispatch } from 'react-redux'; *// Redux hook for dispatching actions*

import { cornersReset, cornersSelect } from '../../slices/cornersSlice'; *// Redux actions and selectors for corners state*

import { Container } from "../common"; *// Layout container for the UI*

import LoadModels from "../../utils/loadModels"; *// Utility to load AI models*

import { CornersDict, Mode, ModelRefs, Study } from "../../types"; *// Type definitions for the app*

import RecordSidebar from "../record/recordSidebar"; *// Sidebar for "record" mode*

import UploadSidebar from "../upload/uploadSidebar"; *// Sidebar for "upload" mode*

import BroadcastSidebar from "../broadcast/broadcastSidebar"; *// Sidebar for "broadcast" mode*

import PlaySidebar from "../play/playSidebar"; *// Sidebar for "play" mode*

import { gameResetFen, gameResetMoves, gameResetStart, gameSelect } from "../../slices/gameSlice"; *// Redux actions and selectors for game state*

import { lichessPushRound } from "../../utils/lichess"; *// Utility for uploading data to Lichess*

import { userSelect } from "../../slices/userSlice"; *// Redux selector for user data*

import { START\_FEN } from "../../utils/constants"; *// Default chessboard FEN*

import { useMediaQuery } from 'react-responsive'; *// Media query hook for detecting orientation*

**Key Dependencies:**

1. Video: Handles video processing and piece detection.

2. **Redux**: Manages state updates for chessboard corners, game status, and user information.

3. **Sidebars**: Dynamically renders mode-specific sidebars (RecordSidebar, UploadSidebar, etc.).

4. **AI Models**: Loaded via LoadModels for detecting pieces and board transformations.

**2. Component Setup**

The VideoAndSidebar component initializes key variables and hooks for managing state, UI, and references.

const VideoAndSidebar = ({ mode }: { mode: Mode }) => {

  const context = useOutletContext<ModelRefs>(); *// Access shared AI model references*

  const dispatch = useDispatch(); *// Dispatch Redux actions*

  const corners: CornersDict = cornersSelect(); *// Get current chessboard corners from Redux*

  const token: string = userSelect().token; *// Get user authentication token*

  const moves: string = gameSelect().moves; *// Get current game moves from Redux*

  const isPortrait = useMediaQuery({ orientation: 'portrait' }); *// Check if the device is in portrait mode*

  const [text, setText] = useState<string[]>([]); *// State for displaying informational text*

  const [playing, setPlaying] = useState<boolean>(false); *// State for video playback*

  const [study, setStudy] = useState<Study | null>(null); *// State for managing Lichess studies*

  const [boardNumber, setBoardNumber] = useState<number>(-1); *// State for managing the chessboard number*

*// Refs for managing DOM elements and state*

  const videoRef = useRef<any>(null);

  const playingRef = useRef<boolean>(playing);

  const canvasRef = useRef<any>(null);

  const sidebarRef = useRef<any>(null);

  const cornersRef = useRef<CornersDict>(corners);

**3. Lichess Broadcasting (Broadcast Mode)**

When in “broadcast” mode, the component prepares and pushes game data to Lichess.

useEffect(() => {

  if (mode !== "broadcast" || study === null || boardNumber === -1) {

    return;

  }

  const broadcastPgn = [

    `[Result "\*"]`,

    `[FEN "${START\_FEN}"]`,

    `[Board "${boardNumber}"]`,

    `[Site "${boardNumber}"]`,

    `[White "White ${boardNumber}"]`,

    `[Black "Black ${boardNumber}"]`,

    `[Annotator "Chess Academy"]`,

    "",

    moves

  ].join("\r");

  lichessPushRound(token, broadcastPgn, study.id); *// Push game data to Lichess*

}, [moves]);

**What This Does:**

• Monitors game moves in Redux (gameSelect().moves).

• Formats them into PGN (Portable Game Notation).

• Sends them to Lichess via the lichessPushRound utility.

**4. Synchronizing State and Models**

The component keeps local and Redux states synchronized.

useEffect(() => {

  playingRef.current = playing; *// Update playing state reference*

}, [playing]);

useEffect(() => {

  cornersRef.current = corners; *// Update corners state reference*

}, [corners]);

useEffect(() => {

  LoadModels(context.piecesModelRef, context.xcornersModelRef); *// Load AI models*

  dispatch(cornersReset()); *// Reset chessboard corners*

  dispatch(gameResetStart()); *// Reset game state*

  dispatch(gameResetMoves());

  dispatch(gameResetFen());

}, []);

**What This Does:**

• **Model Loading**: Ensures the required AI models are loaded for piece detection.

• **State Reset**: Resets Redux states (corners and game) to their initial values.

**5. Dynamic Sidebar Rendering**

The Sidebar function dynamically selects the appropriate sidebar component based on the mode prop.

const Sidebar = () => {

  switch(mode) {

    case "record": return <RecordSidebar {...props} />;

    case "upload": return <UploadSidebar {...props} />;

    case "play": return <PlaySidebar {...props} />;

    case "broadcast": return <BroadcastSidebar {...props} />;

  }

};

**What This Does:**

• Renders different sidebars (RecordSidebar, UploadSidebar, etc.) depending on the active mode.

**6. Combining Video and Sidebar**

The main return statement combines the Video component with the dynamically selected sidebar.

return (

  <Container>

    {isPortrait ? (

      <PortraitWarning />

    ) : (

      <>

        {Sidebar()} {/\* Render the selected sidebar \*/}

        <Video {...props} /> {/\* Render the video component \*/}

      </>

    )}

  </Container>

);

**Key Features:**

• Displays a warning message if the device is in portrait mode.

• Integrates video processing with a sidebar for enhanced interactivity.

**7. Key Relationships**

• Video **Component**:

• Processes video frames and detects chess pieces.

• Shares state with VideoAndSidebar via props.

• Redux State:

• Syncs game state (gameSlice) and corner state (cornersSlice) with the sidebar and video components.

• Sidebars:

• Provide mode-specific functionality (e.g., recording moves, uploading data, broadcasting games).

This expanded workflow describes how videoAndSidebar.tsx manages video processing, state updates, and UI rendering. Let me know if you’d like further clarifications or adjustments!