SAAS Automation Documentation

WORKFLOWS PAGE AND COMPONENTS

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\page.tsx`:

1. **Purpose**:

   `WorkflowPage` → Workflow management UI.

2. **Key Functions/Props**:

   - `WorkflowPage`: Renders UI.
   - `WorkflowButton`: Action button.
   - `Workflows`: Main content.
   - `Props`: Empty (extendable).

3. **Dependencies**:

   - **Imports**:
     - `React` (`'react'`).
     - `WorkflowButton` (`'./_components/workflow-button'`).
     - `Workflows` (`'./_components'`).
   - **Exports**: `WorkflowPage` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\_actions\\workflow-connections.tsx`:

1. **Purpose**:

   Server-side actions for workflow management (Google, Discord, Slack, Notion integrations).

2. **Key Functions**:

   - `getGoogleListener`: Fetches Google resource ID for authenticated user.
   - `onFlowPublish`: Updates workflow publish state.
   - `onCreateNodeTemplate`: Saves templates for Discord, Slack, Notion.
   - `onGetWorkflows`: Fetches workflows for current user.
   - `onCreateWorkflow`: Creates a new workflow.
   - `onGetNodesEdges`: Retrieves nodes and edges for a workflow.

3. **Dependencies**:

   - **Imports**:
     - `Option` (`@/components/ui/multiple-selector`).
     - `db` (`@/lib/db`).
     - `auth`, `currentUser` (`@clerk/nextjs/server`).
   - **Exports**: All functions (`getGoogleListener`, `onFlowPublish`, etc.).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\_components\\index.tsx`:

1. **Purpose**:

   `Workflows` → Displays workflows or "No Workflows" message.

2. **Key Functions/Props**:

   - `Workflows`: Fetches and renders workflows using `onGetWorkflows`.
   - `Workflow`: Renders individual workflow details.
   - `Props`: Empty (unused).

3. **Dependencies**:

- **Imports**:
  - `React` (`'react'`).
  - `Workflow` (`'./workflow'`).
  - `onGetWorkflows` (`'../_actions/workflow-connections'`).
- **Exports**: `Workflows` (default).

⇒ File Name and path:

`src\\app\\(main)\\(pages)\\workflows\\_components\\workflow-button.tsx`:

1. **Purpose**:

   `WorkflowButton` → Button to open a modal for creating workflows.

2. **Key Functions/Props**:
   - `WorkflowButton`: Opens a modal with `Workflowform` on click.
   - `handleClick`: Triggers modal opening with workflow creation form.
   - `Props`: Empty (unused).
3. **Dependencies**:
   - **Imports**:
     - `Workflowform` (`@/components/forms/workflow-form`).
     - `CustomModal` (`@/components/global/custom-modal`).
     - `Button` (`@/components/ui/button`).
     - `useModal` (`@/providers/modal-provider`).
     - `Plus` (`lucide-react`).
     - `React` (`'react'`).
   - **Exports**: `WorkflowButton` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\_components\\workflow.tsx`:

1. **Purpose**:

   `Workflow` → Displays workflow details with a toggle for publish status.

2. **Key Functions/Props**:

   - `Workflow`: Renders workflow card with name, description, and publish toggle.
   - `onPublishFlow`: Placeholder for publish/unpublish functionality (commented out).
   - **Props**:
     - `name`: Workflow name.
     - `description`: Workflow description.
     - `id`: Workflow ID.
     - `publish`: Publish status (boolean or null).

3. **Dependencies**:

   - **Imports**:
     - `React` (`'react'`).
     - `Card`, `CardDescription`, `CardHeader`, `CardTitle` (`@/components/ui/card`).
     - `Link` (`next/link`).
     - `Image` (`next/image`).
     - `Label` (`@/components/ui/label`).
     - `Switch` (`@/components/ui/switch`).
   - **Exports**: `Workflow` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\page.tsx`:

1. **Purpose**:

   `Page` → Workflow editor page with canvas and provider contexts.

2. **Key Functions/Props**:

   - `Page`: Renders workflow editor with `EditorCanvas` wrapped in providers.
   - **Providers**:
     - `EditorProvider`: Manages editor state.
     - `ConnectionsProvider`: Manages connection state.
   - `Props`: Empty (unused).

3. **Dependencies**:

   - **Imports**:
     - `ConnectionsProvider` (`@/providers/connections-provider`).
     - `EditorProvider` (`@/providers/editor-provider`).
     - `React` (`'react'`).
     - `EditorCanvas` (`./_components/editor-canvas`).
   - **Exports**: `Page` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_actions\\workflow-connections.tsx`:

1. **Purpose**:

   Server-side actions for saving and publishing workflows.

2. **Key Functions**:

   - `onCreateNodesEdges`: Updates workflow nodes, edges, and flow path in the database.
   - `onFlowPublish`: Updates workflow publish state in the database.

3. **Dependencies**:

- **Imports**:
  - `db` (`@/lib/db`).
- **Exports**:
  - `onCreateNodesEdges`: Saves workflow nodes and edges.
  - `onFlowPublish`: Updates workflow publish status.

⇒ File Type: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\action-button.tsx`:

1. **Purpose**:

   `ActionButton` → Renders action buttons for testing and saving templates for Discord, Notion, and Slack integrations.

2. **Key Functions/Props**:
   - `ActionButton`: Dynamically renders buttons based on the current service (Discord, Notion, Slack).
   - **Functions**:
     - `onSendDiscordMessage`: Sends a test message to Discord.
     - `onStoreNotionContent`: Tests and stores Notion content.
     - `onStoreSlackContent`: Sends a message to Slack.
     - `onCreateLocalNodeTempate`: Saves templates for the current service.
   - **Props**:
     - `currentService`: Current integration service (Discord, Notion, Slack).
     - `nodeConnection`: Connection details for the service.
     - `channels`: Slack channels (optional).
     - `setChannels`: Function to update Slack channels (optional).
3. **Dependencies**:

- **Imports**:
  - `React`, `useCallback` (`'react'`).
  - `Option` (`./content-based-on-title`).
  - `ConnectionProviderProps` (`@/lib/types`).
  - `usePathname` (`next/navigation`).
  - `Button` (`@/components/ui/button`).
  - `onCreateNodeTemplate` (`../../../_actions/workflow-connections`).
  - `toast` (`sonner`).
  - `onCreateNewPageInDatabase` (`@/app/(main)/(pages)/connections/_actions/notion-connection`).
  - `postMessageToSlack` (`@/app/(main)/(pages)/connections/_actions/slack-connection`).
  - `postContentToWebHook` (`@/app/(main)/(pages)/connections/_actions/discord-connection`).
- **Exports**: `ActionButton` (default).

⇒ File and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\content-based-on-title.tsx`:

1. **Purpose**:

   `ContentBasedOnTitle` → Renders content and UI based on the selected node's title (e.g., Discord, Slack, Notion, Google Drive).

2. **Key Functions/Props**:
   - `ContentBasedOnTitle`: Dynamically renders UI based on the selected node's title.
   - **Props**:
     - `nodeConnection`: Connection details for the selected service.
     - `newState`: Current editor state.

- `file`: Google Drive file details.
- `setFile`: Function to update the file state.
- `selectedSlackChannels`: Selected Slack channels.
- `setSelectedSlackChannels`: Function to update selected Slack channels.

- **Functions**:
  - `useEffect`: Fetches Google Drive file data on mount.
  - `onContentChange`: Updates content for the selected service.

3. **Dependencies**:
   - **Imports**:
     - `AccordionContent` (`@/components/ui/accordion`).
     - `ConnectionProviderProps` (`@/providers/connections-provider`).
     - `EditorState` (`@/providers/editor-provider`).
     - `nodeMapper` (`@/lib/types`).
     - `React`, `useEffect` (`'react'`).
     - `Card`, `CardContent`, `CardDescription`, `CardHeader`, `CardTitle` (`@/components/ui/card`).
     - `Input` (`@/components/ui/input`).
     - `onContentChange` (`@/lib/editor-utils`).
     - `GoogleFileDetails`, `GoogleDriveFiles` (`./google-file-details`, `./google-drive-files`).
     - `ActionButton` (`./action-button`).
     - `axios` (`axios`).
     - `toast` (`sonner`).
     - `getFileMetaData` (`@/app/(main)/(pages)/connections/_actions/google-connection`).
   - **Exports**: `ContentBasedOnTitle` (default).

⇒ File and path:

`src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\custom-handle.tsx`:

1. **Purpose**:

   `CustomHandle` → Custom handle component for node connections with validation logic.

2. **Key Functions/Props**:

   - `CustomHandle`: Renders a custom handle with connection validation logic.
   - **Props**:
     - `props`: Extends `HandleProps` with optional `style`.
   - **Validation Logic**:
     - Ensures valid connections based on source and target node states.
     - Prevents multiple connections to the same target.
     - Allows unlimited connections for "Condition" type nodes.

3. **Dependencies**:

   - **Imports**:
     - `useEditor` (`@/providers/editor-provider`).
     - `Handle`, `HandleProps`, `useStore` (`@xyflow/react`).
     - `React`, `CSSProperties` (`'react'`).
   - **Exports**: `CustomHandle` (default).

⇒ File and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\editor-canvas-card-icon-helper.tsx`:

1. **Purpose**:

`EditorCanvasIconHelper` → Renders icons for different workflow node types.

2. **Key Functions/Props**:
   - `EditorCanvasIconHelper`: Returns an icon based on the `type` prop.
   - **Props**:
     - `type`: Type of the workflow node (`EditorCanvasTypes`).
3. **Dependencies**:
   - **Imports**:
     - `React` (`'react'`).
     - Icons (`Calendar`, `CircuitBoard`, `Database`, etc.) from `lucide-react`.
     - `SlackIcon` (`@/lib/icons/slack-icon`).
     - `EditorCanvasTypes` (`@/lib/types`).
   - **Exports**: `EditorCanvasIconHelper` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\editor-canvas-card-single.tsx`:

1. **Purpose**:

   `EditorCanvasCardSingle` → Renders a single workflow node card with handles, icons, and status indicators.

2. **Key Functions/Props**:
   - `EditorCanvasCardSingle`: Displays a node card with title, description, and dynamic status badge.
   - **Props**:
     - `data`: Node data of type `EditorCanvasCardType`.
   - **Handles**:
     - `CustomHandle`: Renders connection handles for source and target.
   - **Dynamic Status**:

    - Randomly assigns a status color (green, orange, red) to the node.
3. **Dependencies**:
    - **Imports**:
        - `EditorCanvasCardType` (`@/lib/types`).
        - `useEditor` (`@/providers/editor-provider`).
        - `React`, `useMemo` (`'react'`).
        - `Position`, `useNodeId` (`@xyflow/react`).
        - `EditorCanvasIconHelper` (`./editor-canvas-card-icon-helper`).
        - `CustomHandle` (`./custom-handle`).
        - `Badge` (`@/components/ui/badge`).
        - `Card`, `CardDescription`, `CardHeader`, `CardTitle` (`@/components/ui/card`).
        - `clsx` (`clsx`).
    - **Exports**: `EditorCanvasCardSingle` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\editor-canvas-sidebar.tsx`:

1. **Purpose**:

   `EditorCanvasSidebar` → Sidebar for workflow editor with tabs for actions and settings.

2. **Key Functions/Props**:
    - `EditorCanvasSidebar`: Displays draggable action cards and node settings.
    - **Props**:
        - `nodes`: Array of `EditorNodeType` for filtering actions.
    - **Tabs**:
        - **Actions**: Displays draggable cards for triggers and actions.

    - **Settings**: Displays account connections and expected output for selected nodes.
- **Dynamic Behavior**:
    - Fetches Slack channels and updates connections based on node state.

3. **Dependencies**:
    - **Imports**:
        - `EditorCanvasTypes`, `EditorNodeType` (`@/lib/types`).
        - `useNodeConnections` (`@/providers/connections-provider`).
        - `useEditor` (`@/providers/editor-provider`).
        - `Tabs`, `TabsContent`, `TabsList`, `TabsTrigger` (`@/components/ui/tabs`).
        - `React`, `useEffect`, `useState` (`'react'`).
        - `Separator` (`@/components/ui/separator`).
        - `CONNECTIONS`, `EditorCanvasDefaultCardTypes` (`@/lib/constants`).
        - `Card`, `CardDescription`, `CardHeader`, `CardTitle` (`@/components/ui/card`).
        - `fetchBotSlackChannels`, `onConnections`, `onDragStart` (`@/lib/editor-utils`).
        - `EditorCanvasIconHelper` (`./editor-canvas-card-icon-helper`).
        - `Accordion`, `AccordionContent`, `AccordionItem`, `AccordionTrigger` (`@/components/ui/accordion`).
        - `RenderConnectionAccordion` (`./render-connection-accordion`).
        - `RenderOutputAccordion` (`./render-output-accordion`).
        - `SailthreadStore` (`@/store`).
    - **Exports**: `EditorCanvasSidebar` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\editor-canvas.tsx`:

1. **Purpose**:

`EditorCanvas` → Main canvas for workflow editor with drag-and-drop nodes, connections, and a sidebar.

2. **Key Functions/Props**:
   - **State Management**:
     - `nodes`, `edges`: Tracks workflow nodes and connections.
     - `reactFlowInstance`: Manages React Flow instance.
   - **Event Handlers**:
     - `onDragOver`, `onDrop`: Handles node drag-and-drop.
     - `onNodesChange`, `onEdgesChange`, `onConnect`: Updates nodes and edges.
     - `handleClickCanvas`: Resets selected element on canvas click.
   - **Dynamic Behavior**:
     - Fetches workflow data (`onGetNodesEdges`) on mount.
     - Syncs nodes and edges with editor state (`useEffect`).
3. **Dependencies**:
   - **Imports**:
     - `EditorCanvasCardType`, `EditorNodeType` (`@/lib/types`).
     - `useEditor` (`@/providers/editor-provider`).
     - `React`, `useCallback`, `useEffect`, `useMemo`, `useState` (`'react'`).
     - `ReactFlow`, `Background`, `Connection`, `Controls`, `Edge`, `EdgeChange`, `MiniMap`, `NodeChange`, `ReactFlowInstance`, `applyNodeChanges`, `applyEdgeChanges`, `addEdge` (`@xyflow/react`).
     - `EditorCanvasCardSingle` (`./editor-canvas-card-single`).
     - `ResizableHandle`, `ResizablePanel`, `ResizablePanelGroup` (`@/components/ui/resizable`).
     - `toast` (`sonner`).
     - `usePathname` (`next/navigation`).
     - `v4` (`uuid`).

    - `EditorCanvasDefaultCardTypes` (`@/lib/constants`).
    - `FlowInstance` (`./flow-instance`).
    - `EditorCanvasSidebar` (`./editor-canvas-sidebar`).
    - `onGetNodesEdges` (`../../../_actions/workflow-connections`).
- **Exports**: `EditorCanvas` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\flow-instance.tsx`:

1. **Purpose**:

   `FlowInstance` → Manages saving and publishing workflows, and renders child components.

2. **Key Functions/Props**:
   - **Props**:
     - `children`: Child components to render.
     - `edges`: Array of workflow edges.
     - `nodes`: Array of workflow nodes.
   - **Functions**:
     - `onFlowAutomation`: Saves workflow nodes, edges, and flow data.
     - `onPublishWorkflow`: Publishes the workflow.
     - `onAutomateFlow`: Identifies connected nodes and updates flow state.
3. **Dependencies**:
   - **Imports**:
     - `Button` (`@/components/ui/button`).
     - `useNodeConnections` (`@/providers/connections-provider`).
     - `usePathname` (`next/navigation`).

    - `React`, `useCallback`, `useEffect`, `useState` (`'react'`).
    - `onCreateNodesEdges`, `onFlowPublish` (`../_actions/workflow-connections`).
    - `toast` (`sonner`).
  - **Exports**: `FlowInstance` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\google-drive-files.tsx`:

1. **Purpose**:

   `GoogleDriveFiles` → Manages Google Drive file listener and displays its status.

2. **Key Functions/Props**:
   - **State**:
     - `loading`: Tracks loading state during API calls.
     - `isListening`: Tracks whether the listener is active.
   - **Functions**:
     - `reqGoogle`: Initiates Google Drive listener and updates state.
     - `onListener`: Checks if a listener is already active.
3. **Dependencies**:
   - **Imports**:
     - `React`, `useEffect`, `useState` (`'react'`).
     - `toast` (`sonner`).
     - `axios` (`axios`).
     - `getGoogleListener` (`../../../_actions/workflow-connections`).
     - `Button` (`@/components/ui/button`).
     - `Card`, `CardDescription` (`@/components/ui/card`).
     - `CardContainer` (`@/components/global/3d-card`).
   - **Exports**: `GoogleDriveFiles` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\google-file-details.tsx`:

1. **Purpose**:

   `GoogleFileDetails` → Displays details of a Google Drive file and allows adding file data to templates.

2. **Key Functions/Props**:

   - **Props**:
     - `nodeConnection`: Connection details for Google Drive.
     - `title`: Title of the node (e.g., "Google Drive").
     - `gFile`: Google Drive file data.
   - **Functions**:
     - `isGoogleFileNotEmpty`: Checks if the Google Drive file data is valid.
     - `onAddTemplate`: Adds file details to the template.

3. **Dependencies**:

   - **Imports**:
     - `Card`, `CardContent`, `CardDescription` (`@/components/ui/card`).
     - `onAddTemplate` (`@/lib/editor-utils`).
     - `ConnectionProviderProps` (`@/providers/connections-provider`).
     - `React` (`'react'`).
   - **Exports**: `GoogleFileDetails` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\render-connection-accordion.tsx`:

1. **Purpose**:

   `RenderConnectionAccordion` → Renders connection details and Slack channel selection for workflow nodes.

2. **Key Functions/Props**:
    - **Props**:
        - `connection`: Connection details (e.g., title, image, description).
        - `state`: Current editor state.
    - **State Management**:
        - `slackChannels`, `selectedSlackChannels`: Manages Slack channel selection.
        - `isConnected`: Checks if the connection is active.
    - **Dynamic Behavior**:
        - Displays Slack channel selection if the connection is Slack-specific and active.
3. **Dependencies**:
    - **Imports**:
        - `ConnectionCard` (`@/app/(main)/(pages)/connections/_components/connection-card`).
        - `AccordionContent` (`@/components/ui/accordion`).
        - `MultipleSelector` (`@/components/ui/multiple-selector`).
        - `Connection` (`@/lib/types`).
        - `useNodeConnections` (`@/providers/connections-provider`).
        - `EditorState` (`@/providers/editor-provider`).
        - `SailthreadStore` (`@/store`).
        - `Command`, `CommandEmpty`, `CommandGroup`, `CommandInput`, `CommandItem` (`@/components/ui/command`).
        - `Popover`, `PopoverContent`, `PopoverTrigger` (`@/components/ui/popover`).
        - `CheckIcon`, `ChevronsUpDown` (`lucide-react`).
        - `Button` (`@/components/ui/button`).
        - `cn` (`@/lib/utils`).
    - **Exports**: `RenderConnectionAccordion` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\workflows\\editor\\[editorId]\\_components\\render-output-accordion.tsx`:

1. **Purpose**:

   `RenderOutputAccordion` → Renders output content based on the selected node's title and connection details.

2. **Key Functions/Props**:

   - **Props**:
     - `state`: Current editor state.
     - `nodeConnection`: Connection details for the selected node.
   - **State Management**:
     - `googleFile`, `setGoogleFile`: Manages Google Drive file data.
     - `selectedSlackChannels`, `setSelectedSlackChannels`: Manages Slack channel selection.

3. **Dependencies**:

   - **Imports**:
     - `ConnectionProviderProps` (`@/lib/types`).
     - `EditorState` (`@/providers/editor-provider`).
     - `SailthreadStore` (`@/store`).
     - `React` (`'react'`).
     - `ContentBasedOnTitle` (`./content-based-on-title`).
   - **Exports**: `RenderOutputAccordion` (default).

Providers

⇒ File name and path: `src\\providers\\connections-provider.tsx`:

1. **Purpose**:

   `ConnectionsProvider` → Manages state and connections for workflow nodes (Discord, Google, Notion, Slack).

2. **Key Functions/Props**:

   - **State Management**:
     - `discordNode`, `googleNode`, `notionNode`, `slackNode`: Stores connection details for each service.
     - `isLoading`: Tracks loading state.
     - `workflowTemplate`: Stores workflow templates.
   - **Context**:
     - `ConnectionsContext`: Provides global access to connection state.
   - **Hooks**:
     - `useNodeConnections`: Retrieves connection state from context.

3. **Dependencies**:

   - **Imports**:
     - `createContext`, `useContext`, `useState` (`'react'`).
     - `ConnectionProviderProps` (`@/lib/types`).
   - **Exports**:
     - `ConnectionsProvider`: Context provider for connection state.
     - `useNodeConnections`: Hook to access connection state.

⇒ File name and path: `src\\providers\\editor-provider.tsx`:

1. **Purpose**:

   `EditorProvider` → Manages state and actions for the workflow editor, including nodes, edges, and history.

2. **Key Functions/Props**:
    - **State Management**:
        - `editor`: Tracks nodes, edges, and the selected node.
        - `history`: Manages undo/redo history.
    - **Reducer**:
        - `editorReducer`: Handles actions like `UNDO`, `REDO`, `LOAD_DATA`, and `SELECTED_ELEMENT`.
    - **Context**:
        - `EditorContext`: Provides global access to editor state and dispatch function.
    - **Hooks**:
        - `useEditor`: Retrieves editor state and dispatch function from context.
3. **Dependencies**:
    - **Imports**:
        - `EditorActions`, `EditorNodeEdge`, `EditorNodeType` (`@/lib/types`).
        - `createContext`, `useContext`, `useReducer`, `Dispatch` (`'react'`).
    - **Exports**:
        - `EditorProvider`: Context provider for editor state.
        - `useEditor`: Hook to access editor state and dispatch.

⇒ File name and path: `src\\providers\\modal-provider.tsx`:

1. **Purpose**:

    `ModalProvider` → Manages modal state and provides functionality to open and close modals with optional data fetching.
2. **Key Functions/Props**:
    - **State Management**:
        - `isOpen`: Tracks whether the modal is open.

    - `data`: Stores modal data.
    - `showingModal`: Tracks the currently displayed modal.
  - **Functions**:
    - `setOpen`: Opens a modal and optionally fetches data.
    - `setClose`: Closes the modal and resets data.
  - **Context**:
    - `ModalContext`: Provides global access to modal state and functions.
  - **Hooks**:
    - `useModal`: Retrieves modal state and functions from context.
3. **Dependencies**:
  - **Imports**:
    - `createContext`, `useContext`, `useEffect`, `useState` (`'react'`).
  - **Exports**:
    - `ModalProvider`: Context provider for modal state.
    - `useModal`: Hook to access modal state and functions.

⇒ File name and path: `src\\providers\\theme-provider.tsx`:

1. **Purpose**:

   `ThemeProvider` → Wraps the application with `next-themes` to enable theme switching (light/dark mode).
2. **Key Functions/Props**:
  - **Props**:
    - `children`: Child components to be wrapped.
    - `...props`: Additional props for `NextThemesProvider`.
  - **State Management**:

- `isMounted`: Ensures the component is mounted before rendering to avoid SSR issues.

3. **Dependencies**:
   - **Imports**:
     - `React`, `useEffect`, `useState` (`'react'`).
     - `NextThemesProvider`, `ThemeProviderProps` (`next-themes`).
   - **Exports**:
     - `ThemeProvider`: Wrapper for theme management.

CONNECTIONS AND COMPONENTS

⇒ FIle name and path: `src\\app\\(main)\\(pages)\\connections\\page.tsx`:

1. **Purpose**:

   `Connections` → Displays and manages connections for various apps (Discord, Notion, Slack, Google Drive).

2. **Key Functions/Props**:
   - **Props**:
     - `searchParams`: Query parameters for connection details (e.g., `webhook_id`, `access_token`).
   - **Functions**:
     - `onUserConnections`: Handles connecting user accounts to Discord, Notion, and Slack.
     - `getUserData`: Retrieves user connection data.
   - **Dynamic Behavior**:
     - Fetches and displays connection status for each app.
3. **Dependencies**:

- **Imports**:
  - `CONNECTIONS` (`@/lib/constants`).
  - `React` (`'react'`).
  - `ConnectionCard` (`./_components/connection-card`).
  - `currentUser` (`@clerk/nextjs/server`).
  - `onDiscordConnect`, `onNotionConnect`, `onSlackConnect` (`./_actions`).
  - `getUserData` (`./_actions/get-user`).
- **Exports**: `Connections` (default).

⇒ File name and path: `src\\app\\(main)\\(pages)\\connections\\_components\\connection-card.tsx`:

1. **Purpose**:

   `ConnectionCard` → Displays connection details (Discord, Notion, Slack) and allows users to connect or view connection status.

2. **Key Props**:
   - `type`: Connection type (e.g., Discord, Notion, Slack).
   - `icon`: Icon for the connection.
   - `title`: Connection title.
   - `description`: Connection description.
   - `connected`: Object indicating connection status.
3. **Behavior**:
   - Shows "Connected" if linked; otherwise, a "Connect" button redirects to the app's auth URL.
4. **Dependencies**:
   - Imports: `ConnectionTypes`, React, UI components (`Card`, `CardTitle`, etc.), `Image`, `Link`.

- Environment variables: `NEXT_PUBLIC_DISCORD_REDIRECT`, `NEXT_PUBLIC_NOTION_AUTH_URL`, `NEXT_PUBLIC_SLACK_REDIRECT`.

5. **Exports**:
   - `ConnectionCard` (default).
6. **Styling**:
   - Uses Tailwind CSS for layout and conditional rendering.

File name and path: `src\app\(main)\(pages)\connections\_actions\discord-connection.tsx`:

1. **Purpose**:

   Handles Discord connection logic, including creating webhooks, retrieving connection details, and posting content to Discord via webhooks.
2. **Key Functions**:
   - `onDiscordConnect`:
     - Connects a user to Discord by creating or updating a webhook in the database.
     - Checks for existing webhooks to avoid duplicates.
     - Links the webhook to the user and stores details like `channel_id`, `webhook_id`, `guild_name`, etc.
   - `getDiscordConnectionUrl`:
     - Retrieves the Discord webhook URL and details for the current user.
   - `postContentToWebHook`:
     - Posts content to a Discord webhook using the provided URL.
3. **Dependencies**:
   - `db`: Database instance for querying and storing webhook data.
   - `currentUser`: Retrieves the current authenticated user.
   - `axios`: Sends HTTP requests to post content to Discord.
4. **Behavior**:

- Ensures no duplicate webhooks are created for the same user or channel.
- Returns success/failure messages for webhook creation and content posting.

5. **Exports**:
   - `onDiscordConnect`, `getDiscordConnectionUrl`, `postContentToWebHook`.

This module manages Discord integration, enabling users to connect, retrieve, and interact with Discord webhooks seamlessly.

⇒ File Name and Path: `src\app\(main)\(pages)\connections\_actions\get-user.tsx`**:**

1. **Purpose**:

   Retrieves user data, including their connections, from the database.

2. **Key Function**:
   - `getUserData`:
     - Fetches user information and associated connections using the user's `clerkId`.
     - Returns the user object with included connection details.

3. **Dependencies**:
   - `db`: Database instance for querying user data.

4. **Exports**:
   - `getUserData`: The function to fetch user data.

This module provides a simple way to access user information and their connections for use in the `Connections` page.

⇒ File name and path: `src\\app\\(main)\\(pages)\\connections\\_actions\\google-connection.tsx`**:**

1. **Purpose**:

   Fetches file metadata from Google Drive using OAuth2 authentication.

2. **Key Function**:
    - `getFileMetaData`:
        - Initializes Google OAuth2 client with environment variables (`GOOGLE_CLIENT_ID`, `GOOGLE_CLIENT_SECRET`, `OAUTH2_REDIRECT_URI`).
        - Retrieves the authenticated user's Google OAuth access token via Clerk.
        - Uses the access token to authenticate with Google Drive API and fetch file metadata.
        - Returns the file metadata if successful; otherwise, returns an error message.
3. **Dependencies**:
    - `@clerk/nextjs/server`: For retrieving the authenticated user's ID.
    - `googleapis`: For interacting with Google Drive API.
    - Environment variables: `GOOGLE_CLIENT_ID`, `GOOGLE_CLIENT_SECRET`, `OAUTH2_REDIRECT_URI`.
4. **Behavior**:
    - Requires an authenticated user with Google OAuth access.
    - Fetches and returns file metadata from the user's Google Drive.
5. **Exports**:
    - `getFileMetaData`: The function to retrieve Google Drive file metadata.

This module enables integration with Google Drive, allowing users to fetch and manage their file data securely.

⇒ File name and path: `app\(main)\(pages)\connections\_actions\notion-connection.tsx`:

1. **Purpose**:

    Manages Notion integration, including connecting to Notion, retrieving connection details, fetching databases, and creating new pages in a Notion database.

2. **Key Functions**:
    - `onNotionConnect`:
        - Connects a user to Notion by storing their access token, workspace details, and database ID in the database.
        - Ensures no duplicate connections for the same access token.
    - `getNotionConnection`:
        - Retrieves the Notion connection details for the current user.
    - `getNotionDatabase`:
        - Fetches details of a specific Notion database using the Notion API.
    - `onCreateNewPageInDatabase`:
        - Creates a new page in a specified Notion database with the provided content.
3. **Dependencies**:
    - `db`: Database instance for storing and querying Notion connection data.
    - `@clerk/nextjs/server`: For retrieving the current authenticated user.
    - `@notionhq/client`: Notion SDK for interacting with the Notion API.
4. **Behavior**:
    - Ensures unique Notion connections per user.
    - Enables fetching database details and creating new pages programmatically.
5. **Exports**:
    - `onNotionConnect`, `getNotionConnection`, `getNotionDatabase`, `onCreateNewPageInDatabase`.

➔ File Name and path: `src\\app\\(main)\\(pages)\\connections\\_actions\\slack-connection.tsx`:

1. **Purpose**:

Manages Slack integration, including connecting to Slack, retrieving connection details, listing bot-accessible channels, and posting messages to Slack channels.

2. **Key Functions**:
   - `onSlackConnect`:
     - Connects a user to Slack by storing their access token, app details, and team information in the database.
     - Ensures no duplicate connections for the same Slack access token.
   - `getSlackConnection`:
     - Retrieves the Slack connection details for the current user.
   - `listBotChannels`:
     - Fetches a list of Slack channels (public and private) where the bot is a member.
     - Returns channels as `Option[]` (label-value pairs).
   - `postMessageToSlack`:
     - Posts a message to one or multiple Slack channels using the Slack API.
     - Wraps `postMessageInSlackChannel` for handling multiple channels.
3. **Dependencies**:
   - `db`: Database instance for storing and querying Slack connection data.
   - `@clerk/nextjs/server`: For retrieving the current authenticated user.
   - `axios`: For making HTTP requests to the Slack API.
   - `Option`: Type definition for channel selection (label-value pairs).
4. **Behavior**:
   - Ensures unique Slack connections per user.
   - Enables fetching bot-accessible channels and posting messages programmatically.
   - Handles errors gracefully and logs relevant information.

5. **Exports**:
    - `onSlackConnect`, `getSlackConnection`, `listBotChannels`, `postMessageToSlack`.

This module facilitates seamless integration with Slack, allowing users to connect their accounts, manage channels, and send messages programmatically.