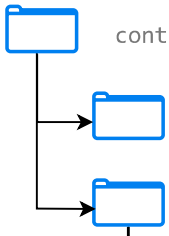

index.ts

```ts
export * from './UIContext'
export * from './UIProvider'
export * from './uiReducer'
```

UIContext.tsx

```tsx
import { createContext } from 'react';

interface ContextProps {
    sidemenuOpen: boolean;
    isAddingEntry: boolean;
    isDragging: boolean;

    // Methods
    openSideMenu: () => void;
    closeSideMenu: () => void

    setIsAddingEntry: (isAdding: boolean) => void

    startDragging: () => void
    endDragging: () => void
}

export const UIContext = createContext({} as ContextProps);
```

UIProvider.tsx

```tsx
import { FC, useReducer, ReactNode  } from 'react';
import { UIContext, uiReducer } from './';

export interface UIState {
    sidemenuOpen: boolean;
    isAddingEntry: boolean;
    isDragging: boolean;
}

const UI_INITIAL_STATE: UIState = {
    sidemenuOpen: false,
    isAddingEntry: false,
    isDragging: false,
}

interface UIProviderProps {
    children: ReactNode;
}

export const UIProvider: FC<UIProviderProps> = ({ children }) => {
    const [state, dispatch] = useReducer(uiReducer, UI_INITIAL_STATE)
```

```
    const openSideMenu = () => {
        dispatch({ type: 'UI - Open Sidebar'})
    }

    const closeSideMenu = () => {
        dispatch({ type: 'UI - Close Sidebar'})
    }

    const setIsAddingEntry = ( isAdding: boolean ) => {
        dispatch({ type:'UI - Set isAddingEntry', payload: isAdding })
    }

    const startDragging = () => {
        dispatch({ type: 'UI - Start Dragging'});
    }

    const endDragging = () => {
        dispatch({ type: 'UI - End Dragging'});
    }

    return (
        <UIContext.Provider value={{
            ...state,

            //Methods
            closeSideMenu,
            openSideMenu,

            setIsAddingEntry,

            startDragging,
            endDragging,
        }} >
            { children }
        </UIContext.Provider>
    )
}
```

uiReducer.ts

```
import { UIState } from "./";

type UIActionType =
    | { type: 'UI - Open Sidebar' }
    | { type: 'UI - Close Sidebar' }
    | { type: 'UI - Set isAddingEntry', payload: boolean }
    | { type: 'UI - Start Dragging'  }
    | { type: 'UI - End Dragging'  }


export const uiReducer = ( state: UIState, action:UIActionType ): UIState => {

    switch (action.type) {
        case 'UI - Open Sidebar':
            return {
                ...state,
                sidemenuOpen: true,
            }

        case 'UI - Close Sidebar':
```

```
        case 'UI - Close Sidebar':
            return {
                ...state,
                sidemenuOpen: false,
            }

        case 'UI - Set isAddingEntry':
            return {
                ...state,
                isAddingEntry: action.payload,
            }

        case 'UI - Start Dragging':
            return {
                ...state,
                isDragging: true
            }

        case 'UI - End Dragging':
            return {
                ...state,
                isDragging: false
            }

        default:
            return state;
    }

}
```