```
import React from 'react'
import { useSelector, useDispatch } from 'react-redux'
import { Radio, Switch, Button, message } from 'antd';
import {
        toggleCollapsedNav,
        onNavTypeChange,
        onNavStyleChange,
        onTopNavColorChange,
        onHeaderNavColorChange,
        onSwitchTheme,
        onDirectionChange
} from 'store/slices/themeSlice';
import { CopyOutlined } from '@ant-design/icons';
import ColorPicker from 'components/shared-components/ColorPicker';
import CopyToClipboard from 'react-copy-to-clipboard';
import NavLanguage from './NavLanguage';
import {
        SIDE_NAV_LIGHT,
        NAV_TYPE_SIDE,
        NAV_TYPE_TOP,
        SIDE_NAV_DARK,
        DIR_RTL,
        DIR_LTR
} from 'constants/ThemeConstant';
import { useThemeSwitcher } from "react-css-theme-switcher";
import utils from 'utils/index';

const colorOptions = [
        '#3e82f7',
        '#24a772',
        '#de4436',
        '#924aca',
        '#193550'
]

const ListOption = ({name, selector, disabled, vertical}) => (
        <div className={`my-4 ${vertical? '' : 'd-flex align-items-center justify-content-between'}`}>
                <div className={`${disabled ? 'opacity-0-3' : ''} ${vertical? 'mb-3' : ''}`}>{name}</div>
                <div>{selector}</div>
        </div>
)

export const ThemeConfigurator = () => {


        const dispatch = useDispatch()

        const {
                navType,
                sideNavTheme,
                navCollapsed,
                topNavColor,
                headerNavColor,
                locale,
                currentTheme,
                direction
        } = useSelector(state => state.theme)

        const isNavTop = navType === NAV_TYPE_TOP
        const isCollapse = navCollapsed


        const { switcher, themes } = useThemeSwitcher();

        const toggleTheme = (isChecked) => {
                onHeaderNavColorChange('')
                const changedTheme = isChecked ? 'dark' : 'light'
                dispatch(onSwitchTheme(changedTheme))
                switcher({ theme: themes[changedTheme] });
        };

        const ontopNavColorClick = (value) => {
                dispatch(onHeaderNavColorChange(''))
                const { rgb } = value
                const rgba = `rgba(${rgb.r}, ${rgb.g}, ${rgb.b}, ${rgb.a})`
                const hex = utils.rgbaToHex(rgba)
                dispatch(onTopNavColorChange(hex))
        }

        const onHeaderNavColorClick = (value) => {
                const { rgb } = value
                const rgba = `rgba(${rgb.r}, ${rgb.g}, ${rgb.b}, ${rgb.a})`
                const hex = utils.rgbaToHex(rgba)
                dispatch(onHeaderNavColorChange(hex))
        }

        const onNavTypeClick = (value) => {
                dispatch(onHeaderNavColorChange(''))
                if(value === NAV_TYPE_TOP) {
                        onTopNavColorChange(colorOptions[0])
                        toggleCollapsedNav(false)
                }
                dispatch(onNavTypeChange(value))
        }

        const haddleNavStyleChange = (style) => {
                dispatch(onNavStyleChange(style))
        }

        const genCopySettingJson = (configState) => JSON.stringify(configState, null, 2);

        const handleToggleCollapseNav = () => {
                dispatch(toggleCollapsedNav(!navCollapsed))
        }

        const handleDirectionChange = (dir) => {
                dispatch(onDirectionChange(dir))
        }
```

```
	return (
		<>
			<div className="mb-5">
				<h4 className="mb-3 font-weight-bold">Navigation</h4>
				{
					isNavTop ?
					<ListOption
						name="Top Nav Color:"
						vertical
						selector={
							<ColorPicker color={topNavColor} colorChange={ontopNavColorClick}/>
						}
					/>
					:
					<ListOption
						name="Header Nav Color:"
						vertical
						selector={
							<ColorPicker color={headerNavColor} colorChange={onHeaderNavColorClick}/>
						}
					/>
				}

				<ListOption
					name="Navigation Type:"
					selector={
						<Radio.Group
							size="small"
							onChange={e => onNavTypeClick(e.target.value)}
							value={navType}
						>
							<Radio.Button value={NAV_TYPE_SIDE}>Side</Radio.Button>
							<Radio.Button value={NAV_TYPE_TOP}>Top</Radio.Button>
						</Radio.Group>
					}
				/>
				<ListOption
					name="Side Nav Color:"
					selector={
						<Radio.Group
							disabled={isNavTop}
							size="small"
							onChange={e => haddleNavStyleChange(e.target.value)}
							value={sideNavTheme}
						>
							<Radio.Button value={SIDE_NAV_LIGHT}>Light</Radio.Button>
							<Radio.Button value={SIDE_NAV_DARK}>Dark</Radio.Button>
						</Radio.Group>
					}
					disabled={isNavTop}
				/>
				<ListOption
					name="Side Nav Collapse:"
					selector={
						<Switch
							disabled={isNavTop}
							checked={isCollapse}
							onChange={handleToggleCollapseNav}
						/>
					}
					disabled={isNavTop}
				/>
				<ListOption
					name="Dark Theme:"
					selector={
						<Switch checked={currentTheme === 'dark'} onChange={toggleTheme} />
					}
				/>
				<ListOption
					name="Direction:"
					selector={
						<Radio.Group
							size="small"
							onChange={e => handleDirectionChange(e.target.value)}
							value={direction}
						>
							<Radio.Button value={DIR_LTR}>LTR</Radio.Button>
							<Radio.Button value={DIR_RTL}>RTL</Radio.Button>
						</Radio.Group>
					}
				/>
			</div>
			<div className="mb-5">
				<h4 className="mb-3 font-weight-bold">Locale</h4>
				<ListOption
					name="Language:"
					selector={
						<NavLanguage configDisplay/>
					}
				/>
			</div>
			<div>
				<CopyToClipboard
					text={genCopySettingJson({ navType, sideNavTheme, navCollapsed, topNavColor, headerNavColor, locale, currentTheme, direction})}
					onCopy={() => message.success('Copy Success, please paste it to src/configs/AppConfig.js THEME_CONFIG variable.') }
				>
					<Button icon={<CopyOutlined /> } block>
						<span>Copy Setting</span>
					</Button>
				</CopyToClipboard>
			</div>
		</>
	)
}

export default ThemeConfigurator
```