File: client/src/components/ui/sidebar.tsx

Type: tsx | Size: 23567 bytes

```tsx
import * as React from "react"
import { Slot } from "@radix-ui/react-slot"
import { VariantProps, cva } from "class-variance-authority"
import { PanelLeft } from "lucide-react"

import { useIsMobile } from "@/hooks/use-mobile"
import { cn } from "@/lib/utils"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import { Separator } from "@/components/ui/separator"
import {
  Sheet,
  SheetContent,
  SheetDescription,
  SheetHeader,
  SheetTitle,
} from "@/components/ui/sheet"
import { Skeleton } from "@/components/ui/skeleton"
import {
  Tooltip,
  TooltipContent,
  TooltipProvider,
  TooltipTrigger,
} from "@/components/ui/tooltip"

const SIDEBAR_COOKIE_NAME = "sidebar_state"
const SIDEBAR_COOKIE_MAX_AGE = 60 * 60 * 24 * 7
const SIDEBAR_WIDTH = "16rem"
const SIDEBAR_WIDTH_MOBILE = "18rem"
const SIDEBAR_WIDTH_ICON = "3rem"
const SIDEBAR_KEYBOARD_SHORTCUT = "b"

type SidebarContextProps = {
  state: "expanded" | "collapsed"
  open: boolean
  setOpen: (open: boolean) => void
  openMobile: boolean
  setOpenMobile: (open: boolean) => void
  isMobile: boolean
  toggleSidebar: () => void
}

const SidebarContext = React.createContext<SidebarContextProps | null>(null)

function useSidebar() {
  const context = React.useContext(SidebarContext)
  if (!context) {
    throw new Error("useSidebar must be used within a SidebarProvider.")
  }

  return context
}

const SidebarProvider = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div"> & {
    defaultOpen?: boolean
    open?: boolean
    onOpenChange?: (open: boolean) => void
  }
>(
  (
    {
      defaultOpen = true,
      open: openProp,
      onOpenChange: setOpenProp,
      className,
      style,
      children,
      ...props
    },
    ref
  ) => {
    const isMobile = useIsMobile()
    const [openMobile, setOpenMobile] = React.useState(false)

    // This is the internal state of the sidebar.
    // We use openProp and setOpenProp for control from outside the component.
    const [_open, _setOpen] = React.useState(defaultOpen)
    const open = openProp ?? _open
    const setOpen = React.useCallback(
      (value: boolean | ((value: boolean) => boolean)) => {
        const openState = typeof value === "function" ? value(open) : value
        if (setOpenProp) {
          setOpenProp(openState)
        } else {
          _setOpen(openState)
        }

        // This sets the cookie to keep the sidebar state.
        document.cookie = `${SIDEBAR_COOKIE_NAME}=${openState}; path=/; max-age=${SIDEBAR_COOKIE_MAX_AGE}`
      },
      [setOpenProp, open]
    )

    // Helper to toggle the sidebar.
    const toggleSidebar = React.useCallback(() => {
      return isMobile
        ? setOpenMobile((open) => !open)
        : setOpen((open) => !open)
    }, [isMobile, setOpen, setOpenMobile])

    // Adds a keyboard shortcut to toggle the sidebar.
    React.useEffect(() => {
      const handleKeyDown = (event: KeyboardEvent) => {
        if (
          event.key === SIDEBAR_KEYBOARD_SHORTCUT &&
          (event.metaKey || event.ctrlKey)
        ) {
          event.preventDefault()
          toggleSidebar()
        }
      }

      window.addEventListener("keydown", handleKeyDown)
      return () => window.removeEventListener("keydown", handleKeyDown)
    }, [toggleSidebar])

    // We add a state so that we can do data-state="expanded" or "collapsed".
    // This makes it easier to style the sidebar with Tailwind classes.
    const state = open ? "expanded" : "collapsed"

    const contextValue = React.useMemo<SidebarContextProps>(
      () => ({
        state,
        open,
        setOpen,
        isMobile,
        openMobile,
        setOpenMobile,
        toggleSidebar,
      }),
      [state, open, setOpen, isMobile, openMobile, setOpenMobile, toggleSidebar]
    )

    return (
      <SidebarContext.Provider value={contextValue}>
        <TooltipProvider delayDuration={0}>
          <div
            style={
              {
                "--sidebar-width": SIDEBAR_WIDTH,
                "--sidebar-width-icon": SIDEBAR_WIDTH_ICON,
                ...style,
              } as React.CSSProperties
            }
            className={cn(
              "group/sidebar-wrapper flex min-h-svh w-full has-[[data-variant=inset]]:bg-sidebar",
              className
            )}
            ref={ref}
            {...props}
          >
            {children}
          </div>
        </TooltipProvider>
      </SidebarContext.Provider>
    )
  }
)
SidebarProvider.displayName = "SidebarProvider"

const Sidebar = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div"> & {
    side?: "left" | "right"
    variant?: "sidebar" | "floating" | "inset"
    collapsible?: "offcanvas" | "icon" | "none"
  }
>(
  (
    {
      side = "left",
      variant = "sidebar",
      collapsible = "offcanvas",
      className,
      children,
      ...props
    },
    ref
  ) => {
    const { isMobile, state, openMobile, setOpenMobile } = useSidebar()

    if (collapsible === "none") {
      return (
        <div
          className={cn(
            "flex h-full w-[--sidebar-width] flex-col bg-sidebar text-sidebar-foreground",
            className
          )}
          ref={ref}
          {...props}
        >
          {children}
        </div>
      )
    }

    if (isMobile) {
      return (
        <Sheet open={openMobile} onOpenChange={setOpenMobile} {...props}>
          <SheetContent
            data-sidebar="sidebar"
            data-mobile="true"
            className="w-[--sidebar-width] bg-sidebar p-0 text-sidebar-foreground [&>button]:hidden"
            style={
              {
                "--sidebar-width": SIDEBAR_WIDTH_MOBILE,
              } as React.CSSProperties
            }
            side={side}
          >
            <SheetHeader className="sr-only">
              <SheetTitle>Sidebar</SheetTitle>
              <SheetDescription>Displays the mobile sidebar.</SheetDescription>
            </SheetHeader>
            <div className="flex h-full w-full flex-col">{children}</div>
          </SheetContent>
        </Sheet>
      )
    }

    return (
      <div
        ref={ref}
        className="group peer hidden text-sidebar-foreground md:block"
        data-state={state}
        data-collapsible={state === "collapsed" ? collapsible : ""}
        data-variant={variant}
        data-side={side}
      >
        {/* This is what handles the sidebar gap on desktop */}
        <div
          className={cn(
            "relative w-[--sidebar-width] bg-transparent transition-[width] duration-200 ease-linear",
            "group-data-[collapsible=offcanvas]:w-0",
            "group-data-[side=right]:rotate-180",
            variant === "floating" || variant === "inset"
              ? "group-data-[collapsible=icon]:w-[calc(var(--sidebar-width-icon)_+_theme(spacing.4))]"
              : "group-data-[collapsible=icon]:w-[--sidebar-width-icon]"
          )}
        />
        <div
          className={cn(
            "fixed inset-y-0 z-10 hidden h-svh w-[--sidebar-width] transition-[left,right,width] duration-200 ...
            side === "left"
              ? "left-0 group-data-[collapsible=offcanvas]:left-[calc(var(--sidebar-width)*-1)]"
              : "right-0 group-data-[collapsible=offcanvas]:right-[calc(var(--sidebar-width)*-1)]",
            // Adjust the padding for floating and inset variants.
            variant === "floating" || variant === "inset"
              ? "p-2 group-data-[collapsible=icon]:w-[calc(var(--sidebar-width-icon)_+_theme(spacing.4)_+2px)]...
              : "group-data-[collapsible=icon]:w-[--sidebar-width-icon] group-data-[side=left]:border-r group-...
            className
          )}
          {...props}
        >
          <div
            data-sidebar="sidebar"
            className="flex h-full w-full flex-col bg-sidebar group-data-[variant=floating]:rounded-lg group-d...
          >
            {children}
          </div>
        </div>
      </div>
    )
  }
)
Sidebar.displayName = "Sidebar"

const SidebarTrigger = React.forwardRef<
  React.ElementRef<typeof Button>,
  React.ComponentProps<typeof Button>
>(({ className, onClick, ...props }, ref) => {
  const { toggleSidebar } = useSidebar()

  return (
    <Button
      ref={ref}
      data-sidebar="trigger"
      variant="ghost"
      size="icon"
      className={cn("h-7 w-7", className)}
      onClick={(event) => {
        onClick?.(event)
        toggleSidebar()
      }}
      {...props}
    >
      <PanelLeft />
      <span className="sr-only">Toggle Sidebar</span>
    </Button>
  )
})
SidebarTrigger.displayName = "SidebarTrigger"

const SidebarRail = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<"button">
>(({ className, ...props }, ref) => {
  const { toggleSidebar } = useSidebar()

  return (
    <button
      ref={ref}
      data-sidebar="rail"
      aria-label="Toggle Sidebar"
      tabIndex={-1}
      onClick={toggleSidebar}
      title="Toggle Sidebar"
      className={cn(
        "absolute inset-y-0 z-20 hidden w-4 -translate-x-1/2 transition-all ease-linear after:absolute after:i...
        "[[data-side=left]_&]:cursor-w-resize [[data-side=right]_&]:cursor-e-resize",
        "[[data-side=left][data-state=collapsed]_&]:cursor-e-resize [[data-side=right][data-state=collapsed]_&...
        "group-data-[collapsible=offcanvas]:translate-x-0 group-data-[collapsible=offcanvas]:after:left-full g...
        "[[data-side=left][data-collapsible=offcanvas]_&]:-right-2",
        "[[data-side=right][data-collapsible=offcanvas]_&]:-left-2",
        className
      )}
      {...props}
    />
  )
})
SidebarRail.displayName = "SidebarRail"

const SidebarInset = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"main">
>(({ className, ...props }, ref) => {
  return (
    <main
      ref={ref}
      className={cn(
        "relative flex w-full flex-1 flex-col bg-background",
        "md:peer-data-[variant=inset]:m-2 md:peer-data-[state=collapsed]:peer-data-[variant=inset]:ml-2 md:pee...
        className
      )}
      {...props}
    />
  )
})
SidebarInset.displayName = "SidebarInset"

const SidebarInput = React.forwardRef<
  React.ElementRef<typeof Input>,
  React.ComponentProps<typeof Input>
>(({ className, ...props }, ref) => {
  return (
    <Input
      ref={ref}
      data-sidebar="input"
      className={cn(
        "h-8 w-full bg-background shadow-none focus-visible:ring-2 focus-visible:ring-sidebar-ring",
        className
      )}
      {...props}
    />
  )
})
SidebarInput.displayName = "SidebarInput"

const SidebarHeader = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div">
>(({ className, ...props }, ref) => {
  return (
    <div
      ref={ref}
      data-sidebar="header"
      className={cn("flex flex-col gap-2 p-2", className)}
      {...props}
    />
  )
})
SidebarHeader.displayName = "SidebarHeader"

const SidebarFooter = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div">
>(({ className, ...props }, ref) => {
  return (
    <div
      ref={ref}
      data-sidebar="footer"
      className={cn("flex flex-col gap-2 p-2", className)}
      {...props}
    />
  )
})
SidebarFooter.displayName = "SidebarFooter"

const SidebarSeparator = React.forwardRef<
  React.ElementRef<typeof Separator>,
  React.ComponentProps<typeof Separator>
>(({ className, ...props }, ref) => {
  return (
    <Separator
      ref={ref}
      data-sidebar="separator"
      className={cn("mx-2 w-auto bg-sidebar-border", className)}
      {...props}
    />
  )
})
SidebarSeparator.displayName = "SidebarSeparator"

const SidebarContent = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div">
>(({ className, ...props }, ref) => {
  return (
    <div
      ref={ref}
      data-sidebar="content"
      className={cn(
        "flex min-h-0 flex-1 flex-col gap-2 overflow-auto group-data-[collapsible=icon]:overflow-hidden",
        className
      )}
      {...props}
    />
  )
})
SidebarContent.displayName = "SidebarContent"

const SidebarGroup = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div">
>(({ className, ...props }, ref) => {
  return (
    <div
      ref={ref}
      data-sidebar="group"
      className={cn("relative flex w-full min-w-0 flex-col p-2", className)}
      {...props}
    />
  )
})
SidebarGroup.displayName = "SidebarGroup"

const SidebarGroupLabel = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div"> & { asChild?: boolean }
>(({ className, asChild = false, ...props }, ref) => {
  const Comp = asChild ? Slot : "div"

  return (
    <Comp
      ref={ref}
      data-sidebar="group-label"
      className={cn(
        "flex h-8 shrink-0 items-center rounded-md px-2 text-xs font-medium text-sidebar-foreground/70 outline...
        "group-data-[collapsible=icon]:-mt-8 group-data-[collapsible=icon]:opacity-0",
        className
      )}
      {...props}
    />
  )
})
SidebarGroupLabel.displayName = "SidebarGroupLabel"

const SidebarGroupAction = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<"button"> & { asChild?: boolean }
>(({ className, asChild = false, ...props }, ref) => {
  const Comp = asChild ? Slot : "button"

  return (
    <Comp
      ref={ref}
      data-sidebar="group-action"
      className={cn(
        "absolute right-3 top-3.5 flex aspect-square w-5 items-center justify-center rounded-md p-0 text-sideb...
        // Increases the hit area of the button on mobile.
        "after:absolute after:-inset-2 after:md:hidden",
        "group-data-[collapsible=icon]:hidden",
        className
      )}
      {...props}
    />
  )
})
SidebarGroupAction.displayName = "SidebarGroupAction"

const SidebarGroupContent = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div">
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    data-sidebar="group-content"
    className={cn("w-full text-sm", className)}
    {...props}
  />
))
SidebarGroupContent.displayName = "SidebarGroupContent"

const SidebarMenu = React.forwardRef<
  HTMLUListElement,
  React.ComponentProps<"ul">
>(({ className, ...props }, ref) => (
  <ul
    ref={ref}
    data-sidebar="menu"
    className={cn("flex w-full min-w-0 flex-col gap-1", className)}
    {...props}
  />
))
SidebarMenu.displayName = "SidebarMenu"
```

```
const SidebarMenuItem = React.forwardRef<
  HTMLLIElement,
  React.ComponentProps<"li">
>(({ className, ...props }, ref) => (
  <li
    ref={ref}
    data-sidebar="menu-item"
    className={cn("group/menu-item relative", className)}
    {...props}
  />
))
SidebarMenuItem.displayName = "SidebarMenuItem"

const sidebarMenuButtonVariants = cva(
  "peer/menu-button flex w-full items-center gap-2 overflow-hidden rounded-md p-2 text-left text-sm outline-no...
  {
    variants: {
      variant: {
        default: "hover:bg-sidebar-accent hover:text-sidebar-accent-foreground",
        outline:
          "bg-background shadow-[0_0_0_1px_hsl(var(--sidebar-border))] hover:bg-sidebar-accent hover:text-side...
      },
      size: {
        default: "h-8 text-sm",
        sm: "h-7 text-xs",
        lg: "h-12 text-sm group-data-[collapsible=icon]:!p-0",
      },
    },
    defaultVariants: {
      variant: "default",
      size: "default",
    },
  }
)

const SidebarMenuButton = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<"button"> & {
    asChild?: boolean
    isActive?: boolean
    tooltip?: string | React.ComponentProps<typeof TooltipContent>
  } & VariantProps<typeof sidebarMenuButtonVariants>
>(
  (
    {
      asChild = false,
      isActive = false,
      variant = "default",
      size = "default",
      tooltip,
      className,
      ...props
    },
    ref
  ) => {
    const Comp = asChild ? Slot : "button"
    const { isMobile, state } = useSidebar()

    const button = (
      <Comp
        ref={ref}
        data-sidebar="menu-button"
        data-size={size}
        data-active={isActive}
        className={cn(sidebarMenuButtonVariants({ variant, size }), className)}
        {...props}
      />
    )

    if (!tooltip) {
      return button
    }
    if (typeof tooltip === "string") {
      tooltip = {
        children: tooltip,
      }
    }

    return (
      <Tooltip>
        <TooltipTrigger asChild>{button}</TooltipTrigger>
        <TooltipContent
          side="right"
          align="center"
          hidden={state !== "collapsed" || isMobile}
          {...tooltip}
        />
      </Tooltip>
    )
  }
)
SidebarMenuButton.displayName = "SidebarMenuButton"

const SidebarMenuAction = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<"button"> & {
    asChild?: boolean
    showOnHover?: boolean
  }
>(({ className, asChild = false, showOnHover = false, ...props }, ref) => {
  const Comp = asChild ? Slot : "button"

  return (
    <Comp
      ref={ref}
      data-sidebar="menu-action"
      className={cn(
        "absolute right-1 top-1.5 flex aspect-square w-5 items-center justify-center rounded-md p-0 text-sideb...
        // Increases the hit area of the button on mobile.
        "after:absolute after:-inset-2 after:md:hidden",
        "peer-data-[size=sm]/menu-button:top-1",
        "peer-data-[size=default]/menu-button:top-1.5",
        "peer-data-[size=lg]/menu-button:top-2.5",
        "group-data-[collapsible=icon]:hidden",
        showOnHover &&
          "group-focus-within/menu-item:opacity-100 group-hover/menu-item:opacity-100 data-[state=open]:opacit...
        className
      )}
      {...props}
    />
  )
})
SidebarMenuAction.displayName = "SidebarMenuAction"

const SidebarMenuBadge = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div">
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    data-sidebar="menu-badge"
    className={cn(
      "pointer-events-none absolute right-1 flex h-5 min-w-5 select-none items-center justify-center rounded-m...
      "peer-hover/menu-button:text-sidebar-accent-foreground peer-data-[active=true]/menu-button:text-sidebar-...
      "peer-data-[size=sm]/menu-button:top-1",
      "peer-data-[size=default]/menu-button:top-1.5",
      "peer-data-[size=lg]/menu-button:top-2.5",
      "group-data-[collapsible=icon]:hidden",
      className
    )}
    {...props}
  />
))
SidebarMenuBadge.displayName = "SidebarMenuBadge"
```

```
const SidebarMenuSkeleton = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div"> & {
    showIcon?: boolean
  }
>(({ className, showIcon = false, ...props }, ref) => {
  // Random width between 50 to 90%.
  const width = React.useMemo(() => {
    return `${Math.floor(Math.random() * 40) + 50}%`
  }, [])

  return (
    <div
      ref={ref}
      data-sidebar="menu-skeleton"
      className={cn("flex h-8 items-center gap-2 rounded-md px-2", className)}
      {...props}
    >
      {showIcon && (
        <Skeleton
          className="size-4 rounded-md"
          data-sidebar="menu-skeleton-icon"
        />
      )}
      <Skeleton
        className="h-4 max-w-[--skeleton-width] flex-1"
        data-sidebar="menu-skeleton-text"
        style={
          {
            "--skeleton-width": width,
          } as React.CSSProperties
        }
      />
    </div>
  )
})
SidebarMenuSkeleton.displayName = "SidebarMenuSkeleton"

const SidebarMenuSub = React.forwardRef<
  HTMLUListElement,
  React.ComponentProps<"ul">
>(({ className, ...props }, ref) => (
  <ul
    ref={ref}
    data-sidebar="menu-sub"
    className={cn(
      "mx-3.5 flex min-w-0 translate-x-px flex-col gap-1 border-l border-sidebar-border px-2.5 py-0.5",
      "group-data-[collapsible=icon]:hidden",
      className
    )}
    {...props}
  />
))
SidebarMenuSub.displayName = "SidebarMenuSub"

const SidebarMenuSubItem = React.forwardRef<
  HTMLLIElement,
  React.ComponentProps<"li">
>(({ ...props }, ref) => <li ref={ref} {...props} />)
SidebarMenuSubItem.displayName = "SidebarMenuSubItem"

const SidebarMenuSubButton = React.forwardRef<
  HTMLAnchorElement,
  React.ComponentProps<"a"> & {
    asChild?: boolean
    size?: "sm" | "md"
    isActive?: boolean
  }
>(({ asChild = false, size = "md", isActive, className, ...props }, ref) => {
  const Comp = asChild ? Slot : "a"

  return (
    <Comp
```

```
      ref={ref}
      data-sidebar="menu-sub-button"
      data-size={size}
      data-active={isActive}
      className={cn(
        "flex h-7 min-w-0 -translate-x-px items-center gap-2 overflow-hidden rounded-md px-2 text-sidebar-fore...
        "data-[active=true]:bg-sidebar-accent data-[active=true]:text-sidebar-accent-foreground",
        size === "sm" && "text-xs",
        size === "md" && "text-sm",
        "group-data-[collapsible=icon]:hidden",
        className
      )}
      {...props}
    />
  )
})
SidebarMenuSubButton.displayName = "SidebarMenuSubButton"

export {
  Sidebar,
  SidebarContent,
  SidebarFooter,
  SidebarGroup,
  SidebarGroupAction,
  SidebarGroupContent,
  SidebarGroupLabel,
  SidebarHeader,
  SidebarInput,
  SidebarInset,
  SidebarMenu,
  SidebarMenuAction,
  SidebarMenuBadge,
  SidebarMenuButton,
  SidebarMenuItem,
  SidebarMenuSkeleton,
  SidebarMenuSub,
  SidebarMenuSubButton,
  SidebarMenuSubItem,
  SidebarProvider,
  SidebarRail,
  SidebarSeparator,
  SidebarTrigger,
  useSidebar,
}
```