Build this as my initial prototype

Copy-paste this component to /components/ui folder:
```tsx
sidebar.tsx
"use client"

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 "@/components/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 } 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 SidebarContext = {
  state: "expanded" | "collapsed"
  open: boolean
  setOpen: (open: boolean) => void
  openMobile: boolean
  setOpenMobile: (open: boolean) => void
  isMobile: boolean
  toggleSidebar: () => void
}

const SidebarContext = React.createContext<SidebarContext | 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<SidebarContext>(
    () => ({
      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}
        >
          <div className="flex h-full w-full flex-col">{children}</div>
        </SheetContent>
      </Sheet>
    )
  }
```

```
  return (
    <div
      ref={ref}
      className="group peer hidden md:block text-sidebar-foreground"
      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(
          "duration-200 relative h-svh w-[--sidebar-width] bg-transparent transition-[width]
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(
          "duration-200 fixed inset-y-0 z-10 hidden h-svh w-[--sidebar-width]
transition-[left,right,width] ease-linear md:flex",
          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-data-[side=right]:border-l",
          className
        )}
        {...props}
      >
        <div
          data-sidebar="sidebar"
```

```
          className="flex h-full w-full flex-col bg-sidebar group-data-[variant=floating]:rounded-lg
group-data-[variant=floating]:border group-data-[variant=floating]:border-sidebar-border
group-data-[variant=floating]:shadow"
        >
          {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:inset-y-0 after:left-1/2 after:w-[2px] hover:after:bg-sidebar-border
group-data-[side=left]:-right-4 group-data-[side=right]:left-0 sm:flex",
        "[[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]_&]:cursor-w-resize",
        "group-data-[collapsible=offcanvas]:translate-x-0
group-data-[collapsible=offcanvas]:after:left-full
group-data-[collapsible=offcanvas]:hover:bg-sidebar",
        "[[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 min-h-svh flex-1 flex-col bg-background",
        "peer-data-[variant=inset]:min-h-[calc(100svh-theme(spacing.4))]
md:peer-data-[variant=inset]:m-2 md:peer-data-[state=collapsed]:peer-data-[variant=inset]:ml-2
md:peer-data-[variant=inset]:ml-0 md:peer-data-[variant=inset]:rounded-xl
md:peer-data-[variant=inset]:shadow",
        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(
        "duration-200 flex h-8 shrink-0 items-center rounded-md px-2 text-xs font-medium
text-sidebar-foreground/70 outline-none ring-sidebar-ring transition-[margin,opacity] ease-linear
focus-visible:ring-2 [&>svg]:size-4 [&>svg]:shrink-0",
        "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-sidebar-foreground outline-none ring-sidebar-ring transition-transform
hover:bg-sidebar-accent hover:text-sidebar-accent-foreground focus-visible:ring-2
[&>svg]:size-4 [&>svg]:shrink-0",
        // 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-none ring-sidebar-ring transition-[width,height,padding] hover:bg-sidebar-accent
hover:text-sidebar-accent-foreground focus-visible:ring-2 active:bg-sidebar-accent
active:text-sidebar-accent-foreground disabled:pointer-events-none disabled:opacity-50
group-has-[[data-sidebar=menu-action]]/menu-item:pr-8 aria-disabled:pointer-events-none
aria-disabled:opacity-50 data-[active=true]:bg-sidebar-accent data-[active=true]:font-medium
data-[active=true]:text-sidebar-accent-foreground data-[state=open]:hover:bg-sidebar-accent
data-[state=open]:hover:text-sidebar-accent-foreground group-data-[collapsible=icon]:!size-8
group-data-[collapsible=icon]:!p-2 [&>span:last-child]:truncate [&>svg]:size-4 [&>svg]:shrink-0",
  {
    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-sidebar-accent-foreground hover:shadow-[0_0_0_1px_hsl(var(--sidebar-accent))]",
      },
      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-sidebar-foreground outline-none ring-sidebar-ring transition-transform
hover:bg-sidebar-accent hover:text-sidebar-accent-foreground focus-visible:ring-2
peer-hover/menu-button:text-sidebar-accent-foreground [&>svg]:size-4 [&>svg]:shrink-0",
        // 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]:opacity-100
peer-data-[active=true]/menu-button:text-sidebar-accent-foreground md:opacity-0",
        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(
      "absolute right-1 flex h-5 min-w-5 items-center justify-center rounded-md px-1 text-xs
font-medium tabular-nums text-sidebar-foreground select-none pointer-events-none",
      "peer-hover/menu-button:text-sidebar-accent-foreground
peer-data-[active=true]/menu-button:text-sidebar-accent-foreground",
      "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("rounded-md h-8 flex gap-2 px-2 items-center", className)}
      {...props}
    >
      {showIcon && (
        <Skeleton
          className="size-4 rounded-md"
          data-sidebar="menu-skeleton-icon"
        />
      )}
      <Skeleton
        className="h-4 flex-1 max-w-[--skeleton-width]"
        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-foreground outline-none ring-sidebar-ring hover:bg-sidebar-accent
hover:text-sidebar-accent-foreground focus-visible:ring-2 active:bg-sidebar-accent
active:text-sidebar-accent-foreground disabled:pointer-events-none disabled:opacity-50
aria-disabled:pointer-events-none aria-disabled:opacity-50 [&>span:last-child]:truncate
[&>svg]:size-4 [&>svg]:shrink-0 [&>svg]:text-sidebar-accent-foreground",
        "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,
}
```

demo.tsx

```
import {
  Sidebar,
  SidebarProvider,
  SidebarContent,
  SidebarGroup,
  SidebarGroupLabel,
  SidebarGroupContent,
  SidebarMenu,
  SidebarMenuItem,
  SidebarMenuButton,
  SidebarFooter,
  SidebarTrigger,
} from "@/components/blocks/sidebar"

import {
  User,
```

```
  ChevronsUpDown,
  Calendar,
  Home,
  Inbox,
  Search,
  Settings
} from "lucide-react"

// Menu items
const items = [
  {
    title: "Home",
    url: "#",
    icon: Home,
  },
  {
    title: "Inbox",
    url: "#",
    icon: Inbox,
  },
  {
    title: "Calendar",
    url: "#",
    icon: Calendar,
  },
  {
    title: "Search",
    url: "#",
    icon: Search,
  },
  {
    title: "Settings",
    url: "#",
    icon: Settings,
  },
]

export function SidebarDemo() {
  return (
    <SidebarProvider>
      <Sidebar>
        <SidebarContent>
          <SidebarGroup>
            <SidebarGroupLabel>Application</SidebarGroupLabel>
```

```
        <SidebarGroupContent>
          <SidebarMenu>
            {items.map((item) => (
              <SidebarMenuItem key={item.title}>
                <SidebarMenuButton asChild tooltip={item.title}>
                  <a href={item.url}>
                    <item.icon />
                    <span>{item.title}</span>
                  </a>
                </SidebarMenuButton>
              </SidebarMenuItem>
            ))}
          </SidebarMenu>
        </SidebarGroupContent>
      </SidebarGroup>
    </SidebarContent>

    <SidebarFooter>
      <SidebarGroup>
        <SidebarMenuButton className="w-full justify-between gap-3 h-12">
          <div className="flex items-center gap-2">
            <User className="h-5 w-5 rounded-md" />
            <div className="flex flex-col items-start">
              <span className="text-sm font-medium">John Doe</span>
              <span className="text-xs text-muted-foreground">john@example.com</span>
            </div>
          </div>
          <ChevronsUpDown className="h-5 w-5 rounded-md" />
        </SidebarMenuButton>
      </SidebarGroup>
    </SidebarFooter>
  </Sidebar>

  <main className="flex-1 min-w-100vh">
    <div className="px-4 py-2">
      <SidebarTrigger className="h-4 w-4 mt-2" />
    </div>
    <div className="p-6">
      <h1 className="text-2xl font-bold">Main Content</h1>
      <p className="mt-2">This is the main content area of the page.</p>
    </div>
  </main>
</SidebarProvider>
)
```

}
```

Copy-paste these files for dependencies:
```tsx
shadcn/use-mobile
import * as React from "react"

const MOBILE_BREAKPOINT = 768

export function useIsMobile() {
  const [isMobile, setIsMobile] = React.useState<boolean | undefined>(undefined)

  React.useEffect(() => {
    const mql = window.matchMedia(`(max-width: ${MOBILE_BREAKPOINT - 1}px)`)
    const onChange = () => {
      setIsMobile(window.innerWidth < MOBILE_BREAKPOINT)
    }
    mql.addEventListener("change", onChange)
    setIsMobile(window.innerWidth < MOBILE_BREAKPOINT)
    return () => mql.removeEventListener("change", onChange)
  }, [])

  return !!isMobile
}

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

import { cn } from "@/lib/utils";

const buttonVariants = cva(
  "inline-flex items-center justify-center whitespace-nowrap rounded-lg text-sm font-medium
transition-colors outline-offset-2 focus-visible:outline focus-visible:outline-2
focus-visible:outline-ring/70 disabled:pointer-events-none disabled:opacity-50
[&_svg]:pointer-events-none [&_svg]:shrink-0",
  {
    variants: {
      variant: {

```
        default: "bg-primary text-primary-foreground shadow-sm shadow-black/5
hover:bg-primary/90",
        destructive:
          "bg-destructive text-destructive-foreground shadow-sm shadow-black/5
hover:bg-destructive/90",
        outline:
          "border border-input bg-background shadow-sm shadow-black/5 hover:bg-accent
hover:text-accent-foreground",
        secondary:
          "bg-secondary text-secondary-foreground shadow-sm shadow-black/5
hover:bg-secondary/80",
        ghost: "hover:bg-accent hover:text-accent-foreground",
        link: "text-primary underline-offset-4 hover:underline",
      },
      size: {
        default: "h-9 px-4 py-2",
        sm: "h-8 rounded-lg px-3 text-xs",
        lg: "h-10 rounded-lg px-8",
        icon: "h-9 w-9",
      },
    },
    defaultVariants: {
      variant: "default",
      size: "default",
    },
  },
);

export interface ButtonProps
  extends React.ButtonHTMLAttributes<HTMLButtonElement>,
    VariantProps<typeof buttonVariants> {
  asChild?: boolean;
}

const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(
  ({ className, variant, size, asChild = false, ...props }, ref) => {
    const Comp = asChild ? Slot : "button";
    return (
      <Comp className={cn(buttonVariants({ variant, size, className }))} ref={ref} {...props} />
    );
  },
);
Button.displayName = "Button";
```

```
export { Button, buttonVariants };

```
```tsx
originui/input
import { cn } from "@/lib/utils";
import * as React from "react";

const Input = React.forwardRef<HTMLInputElement, React.ComponentProps<"input">>(
  ({ className, type, ...props }, ref) => {
    return (
      <input
        type={type}
        className={cn(
          "flex h-9 w-full rounded-lg border border-input bg-background px-3 py-2 text-sm
text-foreground shadow-sm shadow-black/5 transition-shadow
placeholder:text-muted-foreground/70 focus-visible:border-ring focus-visible:outline-none
focus-visible:ring-[3px] focus-visible:ring-ring/20 disabled:cursor-not-allowed
disabled:opacity-50",
          type === "search" &&
            "[&::-webkit-search-cancel-button]:appearance-none
[&::-webkit-search-decoration]:appearance-none
[&::-webkit-search-results-button]:appearance-none
[&::-webkit-search-results-decoration]:appearance-none",
          type === "file" &&
            "p-0 pr-3 italic text-muted-foreground/70 file:me-3 file:h-full file:border-0 file:border-r
file:border-solid file:border-input file:bg-transparent file:px-3 file:text-sm file:font-medium
file:not-italic file:text-foreground",
          className,
        )}
        ref={ref}
        {...props}
      />
    );
  },
);
Input.displayName = "Input";

export { Input };

```
```tsx
shadcn/separator
"use client"
```

```
import * as React from "react"
import * as SeparatorPrimitive from "@radix-ui/react-separator"

import { cn } from "@/lib/utils"

const Separator = React.forwardRef<
  React.ElementRef<typeof SeparatorPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof SeparatorPrimitive.Root>
>(
  (
    { className, orientation = "horizontal", decorative = true, ...props },
    ref
  ) => (
    <SeparatorPrimitive.Root
      ref={ref}
      decorative={decorative}
      orientation={orientation}
      className={cn(
        "shrink-0 bg-border",
        orientation === "horizontal" ? "h-[1px] w-full" : "h-full w-[1px]",
        className
      )}
      {...props}
    />
  )
)
Separator.displayName = SeparatorPrimitive.Root.displayName

export { Separator }

```
```tsx
shadcn/sheet
"use client"

import * as React from "react"
import * as SheetPrimitive from "@radix-ui/react-dialog"
import { cva, type VariantProps } from "class-variance-authority"
import { X } from "lucide-react"

import { cn } from "@/lib/utils"

const Sheet = SheetPrimitive.Root
```

```
const SheetTrigger = SheetPrimitive.Trigger

const SheetClose = SheetPrimitive.Close

const SheetPortal = SheetPrimitive.Portal

const SheetOverlay = React.forwardRef<
  React.ElementRef<typeof SheetPrimitive.Overlay>,
  React.ComponentPropsWithoutRef<typeof SheetPrimitive.Overlay>
>(({ className, ...props }, ref) => (
  <SheetPrimitive.Overlay
    className={cn(
      "fixed inset-0 z-50 bg-black/80  data-[state=open]:animate-in
data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0",
      className,
    )}
    {...props}
    ref={ref}
  />
))
SheetOverlay.displayName = SheetPrimitive.Overlay.displayName

const sheetVariants = cva(
  "fixed z-50 gap-4 bg-background p-6 shadow-lg transition ease-in-out
data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:duration-300
data-[state=open]:duration-500",
  {
    variants: {
      side: {
        top: "inset-x-0 top-0 border-b data-[state=closed]:slide-out-to-top
data-[state=open]:slide-in-from-top",
        bottom:
          "inset-x-0 bottom-0 border-t data-[state=closed]:slide-out-to-bottom
data-[state=open]:slide-in-from-bottom",
        left: "inset-y-0 left-0 h-full w-3/4 border-r data-[state=closed]:slide-out-to-left
data-[state=open]:slide-in-from-left sm:max-w-sm",
        right:
          "inset-y-0 right-0 h-full w-3/4  border-l data-[state=closed]:slide-out-to-right
data-[state=open]:slide-in-from-right sm:max-w-sm",
      },
    },
    defaultVariants: {
      side: "right",
```

```
    },
  },
)

interface SheetContentProps
  extends React.ComponentPropsWithoutRef<typeof SheetPrimitive.Content>,
    VariantProps<typeof sheetVariants> {}

const SheetContent = React.forwardRef<
  React.ElementRef<typeof SheetPrimitive.Content>,
  SheetContentProps
>(({ side = "right", className, children, ...props }, ref) => (
  <SheetPortal>
    <SheetOverlay />
    <SheetPrimitive.Content
      ref={ref}
      className={cn(sheetVariants({ side }), className)}
      {...props}
    >
      {children}
      <SheetPrimitive.Close className="absolute right-4 top-4 rounded-sm opacity-70
ring-offset-background transition-opacity hover:opacity-100 focus:outline-none focus:ring-2
focus:ring-ring focus:ring-offset-2 disabled:pointer-events-none
data-[state=open]:bg-secondary">
        <X className="h-4 w-4" />
        <span className="sr-only">Close</span>
      </SheetPrimitive.Close>
    </SheetPrimitive.Content>
  </SheetPortal>
))
SheetContent.displayName = SheetPrimitive.Content.displayName

const SheetHeader = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLDivElement>) => (
  <div
    className={cn(
      "flex flex-col space-y-2 text-center sm:text-left",
      className,
    )}
    {...props}
  />
)
```

```
SheetHeader.displayName = "SheetHeader"

const SheetFooter = ({
  className,
  ...props
}: React.HTMLAttributes<HTMLDivElement>) => (
  <div
    className={cn(
      "flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",
      className,
    )}
    {...props}
  />
)
SheetFooter.displayName = "SheetFooter"

const SheetTitle = React.forwardRef<
  React.ElementRef<typeof SheetPrimitive.Title>,
  React.ComponentPropsWithoutRef<typeof SheetPrimitive.Title>
>(({ className, ...props }, ref) => (
  <SheetPrimitive.Title
    ref={ref}
    className={cn("text-lg font-semibold text-foreground", className)}
    {...props}
  />
))
SheetTitle.displayName = SheetPrimitive.Title.displayName

const SheetDescription = React.forwardRef<
  React.ElementRef<typeof SheetPrimitive.Description>,
  React.ComponentPropsWithoutRef<typeof SheetPrimitive.Description>
>(({ className, ...props }, ref) => (
  <SheetPrimitive.Description
    ref={ref}
    className={cn("text-sm text-muted-foreground", className)}
    {...props}
  />
))
SheetDescription.displayName = SheetPrimitive.Description.displayName

export {
  Sheet,
  SheetPortal,
  SheetOverlay,
```

```
  SheetTrigger,
  SheetClose,
  SheetContent,
  SheetHeader,
  SheetFooter,
  SheetTitle,
  SheetDescription,
}

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

import { cn } from "@/lib/utils"

const buttonVariants = cva(
  "inline-flex items-center justify-center whitespace-nowrap rounded-md text-sm font-medium
ring-offset-background transition-colors focus-visible:outline-none focus-visible:ring-2
focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none
disabled:opacity-50",
  {
    variants: {
      variant: {
        default: "bg-primary text-primary-foreground hover:bg-primary/90",
        destructive:
          "bg-destructive text-destructive-foreground hover:bg-destructive/90",
        outline:
          "border border-input bg-background hover:bg-accent hover:text-accent-foreground",
        secondary:
          "bg-secondary text-secondary-foreground hover:bg-secondary/80",
        ghost: "hover:bg-accent hover:text-accent-foreground",
        link: "text-primary underline-offset-4 hover:underline",
      },
      size: {
        default: "h-10 px-4 py-2",
        sm: "h-9 rounded-md px-3",
        lg: "h-11 rounded-md px-8",
        icon: "h-10 w-10",
      },
    },
    defaultVariants: {
```

```
      variant: "default",
      size: "default",
    },
  },
)

export interface ButtonProps
  extends React.ButtonHTMLAttributes<HTMLButtonElement>,
    VariantProps<typeof buttonVariants> {
  asChild?: boolean
}

const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(
  ({ className, variant, size, asChild = false, ...props }, ref) => {
    const Comp = asChild ? Slot : "button"
    return (
      <Comp
        className={cn(buttonVariants({ variant, size, className }))}
        ref={ref}
        {...props}
      />
    )
  },
)
Button.displayName = "Button"

export { Button, buttonVariants }

```
```tsx
shadcn/input
import * as React from "react"

import { cn } from "@/lib/utils"

export interface InputProps
  extends React.InputHTMLAttributes<HTMLInputElement> {}

const Input = React.forwardRef<HTMLInputElement, InputProps>(
  ({ className, type, ...props }, ref) => {
    return (
      <input
        type={type}
        className={cn(
```

```
        "flex h-10 w-full rounded-md border border-input bg-background px-3 py-2 text-sm
ring-offset-background file:border-0 file:bg-transparent file:text-sm file:font-medium
file:text-foreground placeholder:text-muted-foreground focus-visible:outline-none
focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed
disabled:opacity-50",
          className
        )}
        ref={ref}
        {...props}
      />
    )
  }
)
Input.displayName = "Input"

export { Input }

```
```tsx
shadcn/label
"use client"

import * as React from "react"
import * as LabelPrimitive from "@radix-ui/react-label"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const labelVariants = cva(
  "text-sm font-medium leading-none peer-disabled:cursor-not-allowed
peer-disabled:opacity-70",
)

const Label = React.forwardRef<
  React.ElementRef<typeof LabelPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root> &
    VariantProps<typeof labelVariants>
>(({ className, ...props }, ref) => (
  <LabelPrimitive.Root
    ref={ref}
    className={cn(labelVariants(), className)}
    {...props}
  />
))
```

```
Label.displayName = LabelPrimitive.Root.displayName

export { Label }

```
```tsx
shadcn/skeleton
import { cn } from "@/lib/utils"

function Skeleton({
  className,
  ...props
}: React.HTMLAttributes<HTMLDivElement>) {
  return (
    <div
      className={cn("animate-pulse rounded-md bg-muted", className)}
      {...props}
    />
  )
}

export { Skeleton }

```
```tsx
originui/tooltip
"use client";

import * as TooltipPrimitive from "@radix-ui/react-tooltip";
import * as React from "react";

import { cn } from "@/lib/utils";

const TooltipProvider = TooltipPrimitive.Provider;

const Tooltip = TooltipPrimitive.Root;

const TooltipTrigger = TooltipPrimitive.Trigger;

const TooltipContent = React.forwardRef<
  React.ElementRef<typeof TooltipPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof TooltipPrimitive.Content> & {
    showArrow?: boolean;
  }
```

```
>(({ className, sideOffset = 4, showArrow = false, ...props }, ref) => (
  <TooltipPrimitive.Portal>
    <TooltipPrimitive.Content
      ref={ref}
      sideOffset={sideOffset}
      className={cn(
        "relative z-50 max-w-[280px] rounded-lg border border-border bg-popover px-3 py-1.5
text-sm text-popover-foreground animate-in fade-in-0 zoom-in-95
data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=closed]:zoom-out-95
data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2
data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",
        className,
      )}
      {...props}
    >
      {props.children}
      {showArrow && (
        <TooltipPrimitive.Arrow className="-my-px fill-popover
drop-shadow-[0_1px_0_hsl(var(--border))]" />
      )}
    </TooltipPrimitive.Content>
  </TooltipPrimitive.Portal>
));
TooltipContent.displayName = TooltipPrimitive.Content.displayName;

export { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger };

```
```tsx
shadcn/tooltip
"use client"

import * as React from "react"
import * as TooltipPrimitive from "@radix-ui/react-tooltip"

import { cn } from "@/lib/utils"

const TooltipProvider = TooltipPrimitive.Provider

const Tooltip = TooltipPrimitive.Root

const TooltipTrigger = TooltipPrimitive.Trigger

const TooltipContent = React.forwardRef<
```

```
  React.ElementRef<typeof TooltipPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof TooltipPrimitive.Content>
>(({ className, sideOffset = 4, ...props }, ref) => (
  <TooltipPrimitive.Content
    ref={ref}
    sideOffset={sideOffset}
    className={cn(
      "z-50 overflow-hidden rounded-md border bg-popover px-3 py-1.5 text-sm
text-popover-foreground shadow-md animate-in fade-in-0 zoom-in-95
data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=closed]:zoom-out-95
data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2
data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",
      className,
    )}
    {...props}
  />
))
TooltipContent.displayName = TooltipPrimitive.Content.displayName

export { Tooltip, TooltipTrigger, TooltipContent, TooltipProvider }

```

Install these NPM dependencies:
```bash
lucide-react, @radix-ui/react-slot, class-variance-authority, @radix-ui/react-separator,
@radix-ui/react-dialog, @radix-ui/react-label, @radix-ui/react-tooltip
```

Extend existing tailwind.config.js with this code:
```js
/** @type {import('tailwindcss').Config} */
module.exports = {
  theme: {
    extend: {
      sidebar: {
        DEFAULT: 'hsl(var(--sidebar-background))',
        foreground: 'hsl(var(--sidebar-foreground))',
        primary: 'hsl(var(--sidebar-primary))',
        'primary-foreground': 'hsl(var(--sidebar-primary-foreground))',
        accent: 'hsl(var(--sidebar-accent))',
        'accent-foreground': 'hsl(var(--sidebar-accent-foreground))',
        border: 'hsl(var(--sidebar-border))',
        ring: 'hsl(var(--sidebar-ring))',
```

```
      },
    },
  },
}
```

Extend existing globals.css with this code:
```css
@layer base {
  :root {
    --sidebar-background: 0 0% 98%;
    --sidebar-foreground: 240 5.3% 26.1%;
    --sidebar-primary: 240 5.9% 10%;
    --sidebar-primary-foreground: 0 0% 98%;
    --sidebar-accent: 240 4.8% 95.9%;
    --sidebar-accent-foreground: 240 5.9% 10%;
    --sidebar-border: 220 13% 91%;
    --sidebar-ring: 217.2 91.2% 59.8%;
  }

  .dark {
    --sidebar-background: 240 5.9% 10%;
    --sidebar-foreground: 240 4.8% 95.9%;
    --sidebar-primary: 224.3 76.3% 48%;
    --sidebar-primary-foreground: 0 0% 100%;
    --sidebar-accent: 240 3.7% 15.9%;
    --sidebar-accent-foreground: 240 4.8% 95.9%;
    --sidebar-border: 240 3.7% 15.9%;
    --sidebar-ring: 217.2 91.2% 59.8%;
  }
}

```