# 📚 Complete Code Documentation

## Directory: NewLinkInsightTracker-main (C:-main-main)

client/src/components/ui/checkbox.tsx

import \* as React from "react"  
import \* as CheckboxPrimitive from "@radix-ui/react-checkbox"  
import { Check } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
const Checkbox = React.forwardRef<  
 React.ElementRef<typeof CheckboxPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof CheckboxPrimitive.Root>  
>(({ className, ...props }, ref) => (  
 <CheckboxPrimitive.Root  
 ref={ref}  
 className={cn(  
 "peer h-4 w-4 shrink-0 rounded-sm border border-primary ring-offset-background focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50 data-[state=checked]:bg-primary data-[state=checked]:text-primary-foreground",  
 className  
 )}  
 {...props}  
 >  
 <CheckboxPrimitive.Indicator  
 className={cn("flex items-center justify-center text-current")}  
 >  
 <Check className="h-4 w-4" />  
 </CheckboxPrimitive.Indicator>  
 </CheckboxPrimitive.Root>  
))  
Checkbox.displayName = CheckboxPrimitive.Root.displayName  
  
export { Checkbox }

client/src/components/ui/collapsible.tsx

import \* as CollapsiblePrimitive from "@radix-ui/react-collapsible"  
  
const Collapsible = CollapsiblePrimitive.Root  
  
const CollapsibleTrigger = CollapsiblePrimitive.CollapsibleTrigger  
  
const CollapsibleContent = CollapsiblePrimitive.CollapsibleContent  
  
export { Collapsible, CollapsibleTrigger, CollapsibleContent }

client/src/components/ui/command.tsx

import \* as React from "react"  
import { type DialogProps } from "@radix-ui/react-dialog"  
import { Command as CommandPrimitive } from "cmdk"  
import { Search } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
import { Dialog, DialogContent } from "@/components/ui/dialog"  
  
const Command = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive>  
>(({ className, ...props }, ref) => (  
 <CommandPrimitive  
 ref={ref}  
 className={cn(  
 "flex h-full w-full flex-col overflow-hidden rounded-md bg-popover text-popover-foreground",  
 className  
 )}  
 {...props}  
 />  
))  
Command.displayName = CommandPrimitive.displayName  
  
interface CommandDialogProps extends DialogProps {}  
  
const CommandDialog = ({ children, ...props }: CommandDialogProps) => {  
 return (  
 <Dialog {...props}>  
 <DialogContent className="overflow-hidden p-0 shadow-lg">  
 <Command className="[&\_[cmdk-group-heading]]:px-2 [&\_[cmdk-group-heading]]:font-medium [&\_[cmdk-group-heading]]:text-muted-foreground [&\_[cmdk-group]:not([hidden])\_~[cmdk-group]]:pt-0 [&\_[cmdk-group]]:px-2 [&\_[cmdk-input-wrapper]\_svg]:h-5 [&\_[cmdk-input-wrapper]\_svg]:w-5 [&\_[cmdk-input]]:h-12 [&\_[cmdk-item]]:px-2 [&\_[cmdk-item]]:py-3 [&\_[cmdk-item]\_svg]:h-5 [&\_[cmdk-item]\_svg]:w-5">  
 {children}  
 </Command>  
 </DialogContent>  
 </Dialog>  
 )  
}  
  
const CommandInput = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive.Input>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive.Input>  
>(({ className, ...props }, ref) => (  
 <div className="flex items-center border-b px-3" cmdk-input-wrapper="">  
 <Search className="mr-2 h-4 w-4 shrink-0 opacity-50" />  
 <CommandPrimitive.Input  
 ref={ref}  
 className={cn(  
 "flex h-11 w-full rounded-md bg-transparent py-3 text-sm outline-none placeholder:text-muted-foreground disabled:cursor-not-allowed disabled:opacity-50",  
 className  
 )}  
 {...props}  
 />  
 </div>  
))  
  
CommandInput.displayName = CommandPrimitive.Input.displayName  
  
const CommandList = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive.List>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive.List>  
>(({ className, ...props }, ref) => (  
 <CommandPrimitive.List  
 ref={ref}  
 className={cn("max-h-[300px] overflow-y-auto overflow-x-hidden", className)}  
 {...props}  
 />  
))  
  
CommandList.displayName = CommandPrimitive.List.displayName  
  
const CommandEmpty = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive.Empty>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive.Empty>  
>((props, ref) => (  
 <CommandPrimitive.Empty  
 ref={ref}  
 className="py-6 text-center text-sm"  
 {...props}  
 />  
))  
  
CommandEmpty.displayName = CommandPrimitive.Empty.displayName  
  
const CommandGroup = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive.Group>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive.Group>  
>(({ className, ...props }, ref) => (  
 <CommandPrimitive.Group  
 ref={ref}  
 className={cn(  
 "overflow-hidden p-1 text-foreground [&\_[cmdk-group-heading]]:px-2 [&\_[cmdk-group-heading]]:py-1.5 [&\_[cmdk-group-heading]]:text-xs [&\_[cmdk-group-heading]]:font-medium [&\_[cmdk-group-heading]]:text-muted-foreground",  
 className  
 )}  
 {...props}  
 />  
))  
  
CommandGroup.displayName = CommandPrimitive.Group.displayName  
  
const CommandSeparator = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive.Separator>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive.Separator>  
>(({ className, ...props }, ref) => (  
 <CommandPrimitive.Separator  
 ref={ref}  
 className={cn("-mx-1 h-px bg-border", className)}  
 {...props}  
 />  
))  
CommandSeparator.displayName = CommandPrimitive.Separator.displayName  
  
const CommandItem = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive.Item>  
>(({ className, ...props }, ref) => (  
 <CommandPrimitive.Item  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none data-[disabled=true]:pointer-events-none data-[selected='true']:bg-accent data-[selected=true]:text-accent-foreground data-[disabled=true]:opacity-50",  
 className  
 )}  
 {...props}  
 />  
))  
  
CommandItem.displayName = CommandPrimitive.Item.displayName  
  
const CommandShortcut = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLSpanElement>) => {  
 return (  
 <span  
 className={cn(  
 "ml-auto text-xs tracking-widest text-muted-foreground",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
CommandShortcut.displayName = "CommandShortcut"  
  
export {  
 Command,  
 CommandDialog,  
 CommandInput,  
 CommandList,  
 CommandEmpty,  
 CommandGroup,  
 CommandItem,  
 CommandShortcut,  
 CommandSeparator,  
}

client/src/components/ui/context-menu.tsx

import \* as React from "react"  
import \* as ContextMenuPrimitive from "@radix-ui/react-context-menu"  
import { Check, ChevronRight, Circle } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
const ContextMenu = ContextMenuPrimitive.Root  
  
const ContextMenuTrigger = ContextMenuPrimitive.Trigger  
  
const ContextMenuGroup = ContextMenuPrimitive.Group  
  
const ContextMenuPortal = ContextMenuPrimitive.Portal  
  
const ContextMenuSub = ContextMenuPrimitive.Sub  
  
const ContextMenuRadioGroup = ContextMenuPrimitive.RadioGroup  
  
const ContextMenuSubTrigger = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.SubTrigger>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.SubTrigger> & {  
 inset?: boolean  
 }  
>(({ className, inset, children, ...props }, ref) => (  
 <ContextMenuPrimitive.SubTrigger  
 ref={ref}  
 className={cn(  
 "flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[state=open]:bg-accent data-[state=open]:text-accent-foreground",  
 inset && "pl-8",  
 className  
 )}  
 {...props}  
 >  
 {children}  
 <ChevronRight className="ml-auto h-4 w-4" />  
 </ContextMenuPrimitive.SubTrigger>  
))  
ContextMenuSubTrigger.displayName = ContextMenuPrimitive.SubTrigger.displayName  
  
const ContextMenuSubContent = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.SubContent>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.SubContent>  
>(({ className, ...props }, ref) => (  
 <ContextMenuPrimitive.SubContent  
 ref={ref}  
 className={cn(  
 "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-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}  
 />  
))  
ContextMenuSubContent.displayName = ContextMenuPrimitive.SubContent.displayName  
  
const ContextMenuContent = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Content>  
>(({ className, ...props }, ref) => (  
 <ContextMenuPrimitive.Portal>  
 <ContextMenuPrimitive.Content  
 ref={ref}  
 className={cn(  
 "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md animate-in fade-in-80 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-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}  
 />  
 </ContextMenuPrimitive.Portal>  
))  
ContextMenuContent.displayName = ContextMenuPrimitive.Content.displayName  
  
const ContextMenuItem = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Item> & {  
 inset?: boolean  
 }  
>(({ className, inset, ...props }, ref) => (  
 <ContextMenuPrimitive.Item  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 inset && "pl-8",  
 className  
 )}  
 {...props}  
 />  
))  
ContextMenuItem.displayName = ContextMenuPrimitive.Item.displayName  
  
const ContextMenuCheckboxItem = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.CheckboxItem>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.CheckboxItem>  
>(({ className, children, checked, ...props }, ref) => (  
 <ContextMenuPrimitive.CheckboxItem  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className  
 )}  
 checked={checked}  
 {...props}  
 >  
 <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">  
 <ContextMenuPrimitive.ItemIndicator>  
 <Check className="h-4 w-4" />  
 </ContextMenuPrimitive.ItemIndicator>  
 </span>  
 {children}  
 </ContextMenuPrimitive.CheckboxItem>  
))  
ContextMenuCheckboxItem.displayName =  
 ContextMenuPrimitive.CheckboxItem.displayName  
  
const ContextMenuRadioItem = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.RadioItem>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.RadioItem>  
>(({ className, children, ...props }, ref) => (  
 <ContextMenuPrimitive.RadioItem  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className  
 )}  
 {...props}  
 >  
 <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">  
 <ContextMenuPrimitive.ItemIndicator>  
 <Circle className="h-2 w-2 fill-current" />  
 </ContextMenuPrimitive.ItemIndicator>  
 </span>  
 {children}  
 </ContextMenuPrimitive.RadioItem>  
))  
ContextMenuRadioItem.displayName = ContextMenuPrimitive.RadioItem.displayName  
  
const ContextMenuLabel = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.Label>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Label> & {  
 inset?: boolean  
 }  
>(({ className, inset, ...props }, ref) => (  
 <ContextMenuPrimitive.Label  
 ref={ref}  
 className={cn(  
 "px-2 py-1.5 text-sm font-semibold text-foreground",  
 inset && "pl-8",  
 className  
 )}  
 {...props}  
 />  
))  
ContextMenuLabel.displayName = ContextMenuPrimitive.Label.displayName  
  
const ContextMenuSeparator = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.Separator>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Separator>  
>(({ className, ...props }, ref) => (  
 <ContextMenuPrimitive.Separator  
 ref={ref}  
 className={cn("-mx-1 my-1 h-px bg-border", className)}  
 {...props}  
 />  
))  
ContextMenuSeparator.displayName = ContextMenuPrimitive.Separator.displayName  
  
const ContextMenuShortcut = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLSpanElement>) => {  
 return (  
 <span  
 className={cn(  
 "ml-auto text-xs tracking-widest text-muted-foreground",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
ContextMenuShortcut.displayName = "ContextMenuShortcut"  
  
export {  
 ContextMenu,  
 ContextMenuTrigger,  
 ContextMenuContent,  
 ContextMenuItem,  
 ContextMenuCheckboxItem,  
 ContextMenuRadioItem,  
 ContextMenuLabel,  
 ContextMenuSeparator,  
 ContextMenuShortcut,  
 ContextMenuGroup,  
 ContextMenuPortal,  
 ContextMenuSub,  
 ContextMenuSubContent,  
 ContextMenuSubTrigger,  
 ContextMenuRadioGroup,  
}

client/src/components/ui/dialog.tsx

import \* as React from "react"  
import \* as DialogPrimitive from "@radix-ui/react-dialog"  
import { X } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
const Dialog = DialogPrimitive.Root  
  
const DialogTrigger = DialogPrimitive.Trigger  
  
const DialogPortal = DialogPrimitive.Portal  
  
const DialogClose = DialogPrimitive.Close  
  
const DialogOverlay = React.forwardRef<  
 React.ElementRef<typeof DialogPrimitive.Overlay>,  
 React.ComponentPropsWithoutRef<typeof DialogPrimitive.Overlay>  
>(({ className, ...props }, ref) => (  
 <DialogPrimitive.Overlay  
 ref={ref}  
 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}  
 />  
))  
DialogOverlay.displayName = DialogPrimitive.Overlay.displayName  
  
const DialogContent = React.forwardRef<  
 React.ElementRef<typeof DialogPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof DialogPrimitive.Content>  
>(({ className, children, ...props }, ref) => (  
 <DialogPortal>  
 <DialogOverlay />  
 <DialogPrimitive.Content  
 ref={ref}  
 className={cn(  
 "fixed left-[50%] top-[50%] z-50 grid w-full max-w-lg translate-x-[-50%] translate-y-[-50%] gap-4 border bg-background p-6 shadow-lg duration-200 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[state=closed]:slide-out-to-left-1/2 data-[state=closed]:slide-out-to-top-[48%] data-[state=open]:slide-in-from-left-1/2 data-[state=open]:slide-in-from-top-[48%] sm:rounded-lg",  
 className  
 )}  
 {...props}  
 >  
 {children}  
 <DialogPrimitive.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-accent data-[state=open]:text-muted-foreground">  
 <X className="h-4 w-4" />  
 <span className="sr-only">Close</span>  
 </DialogPrimitive.Close>  
 </DialogPrimitive.Content>  
 </DialogPortal>  
))  
DialogContent.displayName = DialogPrimitive.Content.displayName  
  
const DialogHeader = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLDivElement>) => (  
 <div  
 className={cn(  
 "flex flex-col space-y-1.5 text-center sm:text-left",  
 className  
 )}  
 {...props}  
 />  
)  
DialogHeader.displayName = "DialogHeader"  
  
const DialogFooter = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLDivElement>) => (  
 <div  
 className={cn(  
 "flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",  
 className  
 )}  
 {...props}  
 />  
)  
DialogFooter.displayName = "DialogFooter"  
  
const DialogTitle = React.forwardRef<  
 React.ElementRef<typeof DialogPrimitive.Title>,  
 React.ComponentPropsWithoutRef<typeof DialogPrimitive.Title>  
>(({ className, ...props }, ref) => (  
 <DialogPrimitive.Title  
 ref={ref}  
 className={cn(  
 "text-lg font-semibold leading-none tracking-tight",  
 className  
 )}  
 {...props}  
 />  
))  
DialogTitle.displayName = DialogPrimitive.Title.displayName  
  
const DialogDescription = React.forwardRef<  
 React.ElementRef<typeof DialogPrimitive.Description>,  
 React.ComponentPropsWithoutRef<typeof DialogPrimitive.Description>  
>(({ className, ...props }, ref) => (  
 <DialogPrimitive.Description  
 ref={ref}  
 className={cn("text-sm text-muted-foreground", className)}  
 {...props}  
 />  
))  
DialogDescription.displayName = DialogPrimitive.Description.displayName  
  
export {  
 Dialog,  
 DialogPortal,  
 DialogOverlay,  
 DialogClose,  
 DialogTrigger,  
 DialogContent,  
 DialogHeader,  
 DialogFooter,  
 DialogTitle,  
 DialogDescription,  
}

client/src/components/ui/drawer.tsx

import \* as React from "react"  
import { Drawer as DrawerPrimitive } from "vaul"  
  
import { cn } from "@/lib/utils"  
  
const Drawer = ({  
 shouldScaleBackground = true,  
 ...props  
}: React.ComponentProps<typeof DrawerPrimitive.Root>) => (  
 <DrawerPrimitive.Root  
 shouldScaleBackground={shouldScaleBackground}  
 {...props}  
 />  
)  
Drawer.displayName = "Drawer"  
  
const DrawerTrigger = DrawerPrimitive.Trigger  
  
const DrawerPortal = DrawerPrimitive.Portal  
  
const DrawerClose = DrawerPrimitive.Close  
  
const DrawerOverlay = React.forwardRef<  
 React.ElementRef<typeof DrawerPrimitive.Overlay>,  
 React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Overlay>  
>(({ className, ...props }, ref) => (  
 <DrawerPrimitive.Overlay  
 ref={ref}  
 className={cn("fixed inset-0 z-50 bg-black/80", className)}  
 {...props}  
 />  
))  
DrawerOverlay.displayName = DrawerPrimitive.Overlay.displayName  
  
const DrawerContent = React.forwardRef<  
 React.ElementRef<typeof DrawerPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Content>  
>(({ className, children, ...props }, ref) => (  
 <DrawerPortal>  
 <DrawerOverlay />  
 <DrawerPrimitive.Content  
 ref={ref}  
 className={cn(  
 "fixed inset-x-0 bottom-0 z-50 mt-24 flex h-auto flex-col rounded-t-[10px] border bg-background",  
 className  
 )}  
 {...props}  
 >  
 <div className="mx-auto mt-4 h-2 w-[100px] rounded-full bg-muted" />  
 {children}  
 </DrawerPrimitive.Content>  
 </DrawerPortal>  
))  
DrawerContent.displayName = "DrawerContent"  
  
const DrawerHeader = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLDivElement>) => (  
 <div  
 className={cn("grid gap-1.5 p-4 text-center sm:text-left", className)}  
 {...props}  
 />  
)  
DrawerHeader.displayName = "DrawerHeader"  
  
const DrawerFooter = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLDivElement>) => (  
 <div  
 className={cn("mt-auto flex flex-col gap-2 p-4", className)}  
 {...props}  
 />  
)  
DrawerFooter.displayName = "DrawerFooter"  
  
const DrawerTitle = React.forwardRef<  
 React.ElementRef<typeof DrawerPrimitive.Title>,  
 React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Title>  
>(({ className, ...props }, ref) => (  
 <DrawerPrimitive.Title  
 ref={ref}  
 className={cn(  
 "text-lg font-semibold leading-none tracking-tight",  
 className  
 )}  
 {...props}  
 />  
))  
DrawerTitle.displayName = DrawerPrimitive.Title.displayName  
  
const DrawerDescription = React.forwardRef<  
 React.ElementRef<typeof DrawerPrimitive.Description>,  
 React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Description>  
>(({ className, ...props }, ref) => (  
 <DrawerPrimitive.Description  
 ref={ref}  
 className={cn("text-sm text-muted-foreground", className)}  
 {...props}  
 />  
))  
DrawerDescription.displayName = DrawerPrimitive.Description.displayName  
  
export {  
 Drawer,  
 DrawerPortal,  
 DrawerOverlay,  
 DrawerTrigger,  
 DrawerClose,  
 DrawerContent,  
 DrawerHeader,  
 DrawerFooter,  
 DrawerTitle,  
 DrawerDescription,  
}

client/src/components/ui/dropdown-menu.tsx

import \* as React from "react"  
import \* as DropdownMenuPrimitive from "@radix-ui/react-dropdown-menu"  
import { Check, ChevronRight, Circle } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
const DropdownMenu = DropdownMenuPrimitive.Root  
  
const DropdownMenuTrigger = DropdownMenuPrimitive.Trigger  
  
const DropdownMenuGroup = DropdownMenuPrimitive.Group  
  
const DropdownMenuPortal = DropdownMenuPrimitive.Portal  
  
const DropdownMenuSub = DropdownMenuPrimitive.Sub  
  
const DropdownMenuRadioGroup = DropdownMenuPrimitive.RadioGroup  
  
const DropdownMenuSubTrigger = React.forwardRef<  
 React.ElementRef<typeof DropdownMenuPrimitive.SubTrigger>,  
 React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubTrigger> & {  
 inset?: boolean  
 }  
>(({ className, inset, children, ...props }, ref) => (  
 <DropdownMenuPrimitive.SubTrigger  
 ref={ref}  
 className={cn(  
 "flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent data-[state=open]:bg-accent",  
 inset && "pl-8",  
 className  
 )}  
 {...props}  
 >  
 {children}  
 <ChevronRight className="ml-auto h-4 w-4" />  
 </DropdownMenuPrimitive.SubTrigger>  
))  
DropdownMenuSubTrigger.displayName =  
 DropdownMenuPrimitive.SubTrigger.displayName  
  
const DropdownMenuSubContent = React.forwardRef<  
 React.ElementRef<typeof DropdownMenuPrimitive.SubContent>,  
 React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubContent>  
>(({ className, ...props }, ref) => (  
 <DropdownMenuPrimitive.SubContent  
 ref={ref}  
 className={cn(  
 "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-lg data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-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}  
 />  
))  
DropdownMenuSubContent.displayName =  
 DropdownMenuPrimitive.SubContent.displayName  
  
const DropdownMenuContent = React.forwardRef<  
 React.ElementRef<typeof DropdownMenuPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Content>  
>(({ className, sideOffset = 4, ...props }, ref) => (  
 <DropdownMenuPrimitive.Portal>  
 <DropdownMenuPrimitive.Content  
 ref={ref}  
 sideOffset={sideOffset}  
 className={cn(  
 "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-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}  
 />  
 </DropdownMenuPrimitive.Portal>  
))  
DropdownMenuContent.displayName = DropdownMenuPrimitive.Content.displayName  
  
const DropdownMenuItem = React.forwardRef<  
 React.ElementRef<typeof DropdownMenuPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Item> & {  
 inset?: boolean  
 }  
>(({ className, inset, ...props }, ref) => (  
 <DropdownMenuPrimitive.Item  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center gap-2 rounded-sm px-2 py-1.5 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50 [&\_svg]:pointer-events-none [&\_svg]:size-4 [&\_svg]:shrink-0",  
 inset && "pl-8",  
 className  
 )}  
 {...props}  
 />  
))  
DropdownMenuItem.displayName = DropdownMenuPrimitive.Item.displayName  
  
const DropdownMenuCheckboxItem = React.forwardRef<  
 React.ElementRef<typeof DropdownMenuPrimitive.CheckboxItem>,  
 React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.CheckboxItem>  
>(({ className, children, checked, ...props }, ref) => (  
 <DropdownMenuPrimitive.CheckboxItem  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className  
 )}  
 checked={checked}  
 {...props}  
 >  
 <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">  
 <DropdownMenuPrimitive.ItemIndicator>  
 <Check className="h-4 w-4" />  
 </DropdownMenuPrimitive.ItemIndicator>  
 </span>  
 {children}  
 </DropdownMenuPrimitive.CheckboxItem>  
))  
DropdownMenuCheckboxItem.displayName =  
 DropdownMenuPrimitive.CheckboxItem.displayName  
  
const DropdownMenuRadioItem = React.forwardRef<  
 React.ElementRef<typeof DropdownMenuPrimitive.RadioItem>,  
 React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.RadioItem>  
>(({ className, children, ...props }, ref) => (  
 <DropdownMenuPrimitive.RadioItem  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className  
 )}  
 {...props}  
 >  
 <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">  
 <DropdownMenuPrimitive.ItemIndicator>  
 <Circle className="h-2 w-2 fill-current" />  
 </DropdownMenuPrimitive.ItemIndicator>  
 </span>  
 {children}  
 </DropdownMenuPrimitive.RadioItem>  
))  
DropdownMenuRadioItem.displayName = DropdownMenuPrimitive.RadioItem.displayName  
  
const DropdownMenuLabel = React.forwardRef<  
 React.ElementRef<typeof DropdownMenuPrimitive.Label>,  
 React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Label> & {  
 inset?: boolean  
 }  
>(({ className, inset, ...props }, ref) => (  
 <DropdownMenuPrimitive.Label  
 ref={ref}  
 className={cn(  
 "px-2 py-1.5 text-sm font-semibold",  
 inset && "pl-8",  
 className  
 )}  
 {...props}  
 />  
))  
DropdownMenuLabel.displayName = DropdownMenuPrimitive.Label.displayName  
  
const DropdownMenuSeparator = React.forwardRef<  
 React.ElementRef<typeof DropdownMenuPrimitive.Separator>,  
 React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Separator>  
>(({ className, ...props }, ref) => (  
 <DropdownMenuPrimitive.Separator  
 ref={ref}  
 className={cn("-mx-1 my-1 h-px bg-muted", className)}  
 {...props}  
 />  
))  
DropdownMenuSeparator.displayName = DropdownMenuPrimitive.Separator.displayName  
  
const DropdownMenuShortcut = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLSpanElement>) => {  
 return (  
 <span  
 className={cn("ml-auto text-xs tracking-widest opacity-60", className)}  
 {...props}  
 />  
 )  
}  
DropdownMenuShortcut.displayName = "DropdownMenuShortcut"  
  
export {  
 DropdownMenu,  
 DropdownMenuTrigger,  
 DropdownMenuContent,  
 DropdownMenuItem,  
 DropdownMenuCheckboxItem,  
 DropdownMenuRadioItem,  
 DropdownMenuLabel,  
 DropdownMenuSeparator,  
 DropdownMenuShortcut,  
 DropdownMenuGroup,  
 DropdownMenuPortal,  
 DropdownMenuSub,  
 DropdownMenuSubContent,  
 DropdownMenuSubTrigger,  
 DropdownMenuRadioGroup,  
}

client/src/components/ui/form.tsx

import \* as React from "react"  
import \* as LabelPrimitive from "@radix-ui/react-label"  
import { Slot } from "@radix-ui/react-slot"  
import {  
 Controller,  
 ControllerProps,  
 FieldPath,  
 FieldValues,  
 FormProvider,  
 useFormContext,  
} from "react-hook-form"  
  
import { cn } from "@/lib/utils"  
import { Label } from "@/components/ui/label"  
  
const Form = FormProvider  
  
type FormFieldContextValue<  
 TFieldValues extends FieldValues = FieldValues,  
 TName extends FieldPath<TFieldValues> = FieldPath<TFieldValues>  
> = {  
 name: TName  
}  
  
const FormFieldContext = React.createContext<FormFieldContextValue>(  
 {} as FormFieldContextValue  
)  
  
const FormField = <  
 TFieldValues extends FieldValues = FieldValues,  
 TName extends FieldPath<TFieldValues> = FieldPath<TFieldValues>  
>({  
 ...props  
}: ControllerProps<TFieldValues, TName>) => {  
 return (  
 <FormFieldContext.Provider value={{ name: props.name }}>  
 <Controller {...props} />  
 </FormFieldContext.Provider>  
 )  
}  
  
const useFormField = () => {  
 const fieldContext = React.useContext(FormFieldContext)  
 const itemContext = React.useContext(FormItemContext)  
 const { getFieldState, formState } = useFormContext()  
  
 const fieldState = getFieldState(fieldContext.name, formState)  
  
 if (!fieldContext) {  
 throw new Error("useFormField should be used within <FormField>")  
 }  
  
 const { id } = itemContext  
  
 return {  
 id,  
 name: fieldContext.name,  
 formItemId: `${id}-form-item`,  
 formDescriptionId: `${id}-form-item-description`,  
 formMessageId: `${id}-form-item-message`,  
 ...fieldState,  
 }  
}  
  
type FormItemContextValue = {  
 id: string  
}  
  
const FormItemContext = React.createContext<FormItemContextValue>(  
 {} as FormItemContextValue  
)  
  
const FormItem = React.forwardRef<  
 HTMLDivElement,  
 React.HTMLAttributes<HTMLDivElement>  
>(({ className, ...props }, ref) => {  
 const id = React.useId()  
  
 return (  
 <FormItemContext.Provider value={{ id }}>  
 <div ref={ref} className={cn("space-y-2", className)} {...props} />  
 </FormItemContext.Provider>  
 )  
})  
FormItem.displayName = "FormItem"  
  
const FormLabel = React.forwardRef<  
 React.ElementRef<typeof LabelPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root>  
>(({ className, ...props }, ref) => {  
 const { error, formItemId } = useFormField()  
  
 return (  
 <Label  
 ref={ref}  
 className={cn(error && "text-destructive", className)}  
 htmlFor={formItemId}  
 {...props}  
 />  
 )  
})  
FormLabel.displayName = "FormLabel"  
  
const FormControl = React.forwardRef<  
 React.ElementRef<typeof Slot>,  
 React.ComponentPropsWithoutRef<typeof Slot>  
>(({ ...props }, ref) => {  
 const { error, formItemId, formDescriptionId, formMessageId } = useFormField()  
  
 return (  
 <Slot  
 ref={ref}  
 id={formItemId}  
 aria-describedby={  
 !error  
 ? `${formDescriptionId}`  
 : `${formDescriptionId} ${formMessageId}`  
 }  
 aria-invalid={!!error}  
 {...props}  
 />  
 )  
})  
FormControl.displayName = "FormControl"  
  
const FormDescription = React.forwardRef<  
 HTMLParagraphElement,  
 React.HTMLAttributes<HTMLParagraphElement>  
>(({ className, ...props }, ref) => {  
 const { formDescriptionId } = useFormField()  
  
 return (  
 <p  
 ref={ref}  
 id={formDescriptionId}  
 className={cn("text-sm text-muted-foreground", className)}  
 {...props}  
 />  
 )  
})  
FormDescription.displayName = "FormDescription"  
  
const FormMessage = React.forwardRef<  
 HTMLParagraphElement,  
 React.HTMLAttributes<HTMLParagraphElement>  
>(({ className, children, ...props }, ref) => {  
 const { error, formMessageId } = useFormField()  
 const body = error ? String(error?.message) : children  
  
 if (!body) {  
 return null  
 }  
  
 return (  
 <p  
 ref={ref}  
 id={formMessageId}  
 className={cn("text-sm font-medium text-destructive", className)}  
 {...props}  
 >  
 {body}  
 </p>  
 )  
})  
FormMessage.displayName = "FormMessage"  
  
export {  
 useFormField,  
 Form,  
 FormItem,  
 FormLabel,  
 FormControl,  
 FormDescription,  
 FormMessage,  
 FormField,  
}

client/src/components/ui/hover-card.tsx

import \* as React from "react"  
import \* as HoverCardPrimitive from "@radix-ui/react-hover-card"  
  
import { cn } from "@/lib/utils"  
  
const HoverCard = HoverCardPrimitive.Root  
  
const HoverCardTrigger = HoverCardPrimitive.Trigger  
  
const HoverCardContent = React.forwardRef<  
 React.ElementRef<typeof HoverCardPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof HoverCardPrimitive.Content>  
>(({ className, align = "center", sideOffset = 4, ...props }, ref) => (  
 <HoverCardPrimitive.Content  
 ref={ref}  
 align={align}  
 sideOffset={sideOffset}  
 className={cn(  
 "z-50 w-64 rounded-md border bg-popover p-4 text-popover-foreground shadow-md outline-none data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-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}  
 />  
))  
HoverCardContent.displayName = HoverCardPrimitive.Content.displayName  
  
export { HoverCard, HoverCardTrigger, HoverCardContent }

client/src/components/ui/input-otp.tsx

import \* as React from "react"  
import { OTPInput, OTPInputContext } from "input-otp"  
import { Dot } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
const InputOTP = React.forwardRef<  
 React.ElementRef<typeof OTPInput>,  
 React.ComponentPropsWithoutRef<typeof OTPInput>  
>(({ className, containerClassName, ...props }, ref) => (  
 <OTPInput  
 ref={ref}  
 containerClassName={cn(  
 "flex items-center gap-2 has-[:disabled]:opacity-50",  
 containerClassName  
 )}  
 className={cn("disabled:cursor-not-allowed", className)}  
 {...props}  
 />  
))  
InputOTP.displayName = "InputOTP"  
  
const InputOTPGroup = React.forwardRef<  
 React.ElementRef<"div">,  
 React.ComponentPropsWithoutRef<"div">  
>(({ className, ...props }, ref) => (  
 <div ref={ref} className={cn("flex items-center", className)} {...props} />  
))  
InputOTPGroup.displayName = "InputOTPGroup"  
  
const InputOTPSlot = React.forwardRef<  
 React.ElementRef<"div">,  
 React.ComponentPropsWithoutRef<"div"> & { index: number }  
>(({ index, className, ...props }, ref) => {  
 const inputOTPContext = React.useContext(OTPInputContext)  
 const { char, hasFakeCaret, isActive } = inputOTPContext.slots[index]  
  
 return (  
 <div  
 ref={ref}  
 className={cn(  
 "relative flex h-10 w-10 items-center justify-center border-y border-r border-input text-sm transition-all first:rounded-l-md first:border-l last:rounded-r-md",  
 isActive && "z-10 ring-2 ring-ring ring-offset-background",  
 className  
 )}  
 {...props}  
 >  
 {char}  
 {hasFakeCaret && (  
 <div className="pointer-events-none absolute inset-0 flex items-center justify-center">  
 <div className="h-4 w-px animate-caret-blink bg-foreground duration-1000" />  
 </div>  
 )}  
 </div>  
 )  
})  
InputOTPSlot.displayName = "InputOTPSlot"  
  
const InputOTPSeparator = React.forwardRef<  
 React.ElementRef<"div">,  
 React.ComponentPropsWithoutRef<"div">  
>(({ ...props }, ref) => (  
 <div ref={ref} role="separator" {...props}>  
 <Dot />  
 </div>  
))  
InputOTPSeparator.displayName = "InputOTPSeparator"  
  
export { InputOTP, InputOTPGroup, InputOTPSlot, InputOTPSeparator }

client/src/components/ui/input.tsx

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 }

client/src/components/ui/label.tsx

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 }

client/src/components/ui/menubar.tsx

import \* as React from "react"  
import \* as MenubarPrimitive from "@radix-ui/react-menubar"  
import { Check, ChevronRight, Circle } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
const MenubarMenu = MenubarPrimitive.Menu  
  
const MenubarGroup = MenubarPrimitive.Group  
  
const MenubarPortal = MenubarPrimitive.Portal  
  
const MenubarSub = MenubarPrimitive.Sub  
  
const MenubarRadioGroup = MenubarPrimitive.RadioGroup  
  
const Menubar = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Root>  
>(({ className, ...props }, ref) => (  
 <MenubarPrimitive.Root  
 ref={ref}  
 className={cn(  
 "flex h-10 items-center space-x-1 rounded-md border bg-background p-1",  
 className  
 )}  
 {...props}  
 />  
))  
Menubar.displayName = MenubarPrimitive.Root.displayName  
  
const MenubarTrigger = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.Trigger>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Trigger>  
>(({ className, ...props }, ref) => (  
 <MenubarPrimitive.Trigger  
 ref={ref}  
 className={cn(  
 "flex cursor-default select-none items-center rounded-sm px-3 py-1.5 text-sm font-medium outline-none focus:bg-accent focus:text-accent-foreground data-[state=open]:bg-accent data-[state=open]:text-accent-foreground",  
 className  
 )}  
 {...props}  
 />  
))  
MenubarTrigger.displayName = MenubarPrimitive.Trigger.displayName  
  
const MenubarSubTrigger = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.SubTrigger>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.SubTrigger> & {  
 inset?: boolean  
 }  
>(({ className, inset, children, ...props }, ref) => (  
 <MenubarPrimitive.SubTrigger  
 ref={ref}  
 className={cn(  
 "flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[state=open]:bg-accent data-[state=open]:text-accent-foreground",  
 inset && "pl-8",  
 className  
 )}  
 {...props}  
 >  
 {children}  
 <ChevronRight className="ml-auto h-4 w-4" />  
 </MenubarPrimitive.SubTrigger>  
))  
MenubarSubTrigger.displayName = MenubarPrimitive.SubTrigger.displayName  
  
const MenubarSubContent = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.SubContent>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.SubContent>  
>(({ className, ...props }, ref) => (  
 <MenubarPrimitive.SubContent  
 ref={ref}  
 className={cn(  
 "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-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}  
 />  
))  
MenubarSubContent.displayName = MenubarPrimitive.SubContent.displayName  
  
const MenubarContent = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Content>  
>(  
 (  
 { className, align = "start", alignOffset = -4, sideOffset = 8, ...props },  
 ref  
 ) => (  
 <MenubarPrimitive.Portal>  
 <MenubarPrimitive.Content  
 ref={ref}  
 align={align}  
 alignOffset={alignOffset}  
 sideOffset={sideOffset}  
 className={cn(  
 "z-50 min-w-[12rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-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}  
 />  
 </MenubarPrimitive.Portal>  
 )  
)  
MenubarContent.displayName = MenubarPrimitive.Content.displayName  
  
const MenubarItem = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Item> & {  
 inset?: boolean  
 }  
>(({ className, inset, ...props }, ref) => (  
 <MenubarPrimitive.Item  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 inset && "pl-8",  
 className  
 )}  
 {...props}  
 />  
))  
MenubarItem.displayName = MenubarPrimitive.Item.displayName  
  
const MenubarCheckboxItem = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.CheckboxItem>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.CheckboxItem>  
>(({ className, children, checked, ...props }, ref) => (  
 <MenubarPrimitive.CheckboxItem  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className  
 )}  
 checked={checked}  
 {...props}  
 >  
 <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">  
 <MenubarPrimitive.ItemIndicator>  
 <Check className="h-4 w-4" />  
 </MenubarPrimitive.ItemIndicator>  
 </span>  
 {children}  
 </MenubarPrimitive.CheckboxItem>  
))  
MenubarCheckboxItem.displayName = MenubarPrimitive.CheckboxItem.displayName  
  
const MenubarRadioItem = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.RadioItem>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.RadioItem>  
>(({ className, children, ...props }, ref) => (  
 <MenubarPrimitive.RadioItem  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className  
 )}  
 {...props}  
 >  
 <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">  
 <MenubarPrimitive.ItemIndicator>  
 <Circle className="h-2 w-2 fill-current" />  
 </MenubarPrimitive.ItemIndicator>  
 </span>  
 {children}  
 </MenubarPrimitive.RadioItem>  
))  
MenubarRadioItem.displayName = MenubarPrimitive.RadioItem.displayName  
  
const MenubarLabel = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.Label>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Label> & {  
 inset?: boolean  
 }  
>(({ className, inset, ...props }, ref) => (  
 <MenubarPrimitive.Label  
 ref={ref}  
 className={cn(  
 "px-2 py-1.5 text-sm font-semibold",  
 inset && "pl-8",  
 className  
 )}  
 {...props}  
 />  
))  
MenubarLabel.displayName = MenubarPrimitive.Label.displayName  
  
const MenubarSeparator = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.Separator>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Separator>  
>(({ className, ...props }, ref) => (  
 <MenubarPrimitive.Separator  
 ref={ref}  
 className={cn("-mx-1 my-1 h-px bg-muted", className)}  
 {...props}  
 />  
))  
MenubarSeparator.displayName = MenubarPrimitive.Separator.displayName  
  
const MenubarShortcut = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLSpanElement>) => {  
 return (  
 <span  
 className={cn(  
 "ml-auto text-xs tracking-widest text-muted-foreground",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
MenubarShortcut.displayname = "MenubarShortcut"  
  
export {  
 Menubar,  
 MenubarMenu,  
 MenubarTrigger,  
 MenubarContent,  
 MenubarItem,  
 MenubarSeparator,  
 MenubarLabel,  
 MenubarCheckboxItem,  
 MenubarRadioGroup,  
 MenubarRadioItem,  
 MenubarPortal,  
 MenubarSubContent,  
 MenubarSubTrigger,  
 MenubarGroup,  
 MenubarSub,  
 MenubarShortcut,  
}

client/src/components/ui/navigation-menu.tsx

import \* as React from "react"  
import \* as NavigationMenuPrimitive from "@radix-ui/react-navigation-menu"  
import { cva } from "class-variance-authority"  
import { ChevronDown } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
const NavigationMenu = React.forwardRef<  
 React.ElementRef<typeof NavigationMenuPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Root>  
>(({ className, children, ...props }, ref) => (  
 <NavigationMenuPrimitive.Root  
 ref={ref}  
 className={cn(  
 "relative z-10 flex max-w-max flex-1 items-center justify-center",  
 className  
 )}  
 {...props}  
 >  
 {children}  
 <NavigationMenuViewport />  
 </NavigationMenuPrimitive.Root>  
))  
NavigationMenu.displayName = NavigationMenuPrimitive.Root.displayName  
  
const NavigationMenuList = React.forwardRef<  
 React.ElementRef<typeof NavigationMenuPrimitive.List>,  
 React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.List>  
>(({ className, ...props }, ref) => (  
 <NavigationMenuPrimitive.List  
 ref={ref}  
 className={cn(  
 "group flex flex-1 list-none items-center justify-center space-x-1",  
 className  
 )}  
 {...props}  
 />  
))  
NavigationMenuList.displayName = NavigationMenuPrimitive.List.displayName  
  
const NavigationMenuItem = NavigationMenuPrimitive.Item  
  
const navigationMenuTriggerStyle = cva(  
 "group inline-flex h-10 w-max items-center justify-center rounded-md bg-background px-4 py-2 text-sm font-medium transition-colors hover:bg-accent hover:text-accent-foreground focus:bg-accent focus:text-accent-foreground focus:outline-none disabled:pointer-events-none disabled:opacity-50 data-[active]:bg-accent/50 data-[state=open]:bg-accent/50"  
)  
  
const NavigationMenuTrigger = React.forwardRef<  
 React.ElementRef<typeof NavigationMenuPrimitive.Trigger>,  
 React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Trigger>  
>(({ className, children, ...props }, ref) => (  
 <NavigationMenuPrimitive.Trigger  
 ref={ref}  
 className={cn(navigationMenuTriggerStyle(), "group", className)}  
 {...props}  
 >  
 {children}{" "}  
 <ChevronDown  
 className="relative top-[1px] ml-1 h-3 w-3 transition duration-200 group-data-[state=open]:rotate-180"  
 aria-hidden="true"  
 />  
 </NavigationMenuPrimitive.Trigger>  
))  
NavigationMenuTrigger.displayName = NavigationMenuPrimitive.Trigger.displayName  
  
const NavigationMenuContent = React.forwardRef<  
 React.ElementRef<typeof NavigationMenuPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Content>  
>(({ className, ...props }, ref) => (  
 <NavigationMenuPrimitive.Content  
 ref={ref}  
 className={cn(  
 "left-0 top-0 w-full data-[motion^=from-]:animate-in data-[motion^=to-]:animate-out data-[motion^=from-]:fade-in data-[motion^=to-]:fade-out data-[motion=from-end]:slide-in-from-right-52 data-[motion=from-start]:slide-in-from-left-52 data-[motion=to-end]:slide-out-to-right-52 data-[motion=to-start]:slide-out-to-left-52 md:absolute md:w-auto ",  
 className  
 )}  
 {...props}  
 />  
))  
NavigationMenuContent.displayName = NavigationMenuPrimitive.Content.displayName  
  
const NavigationMenuLink = NavigationMenuPrimitive.Link  
  
const NavigationMenuViewport = React.forwardRef<  
 React.ElementRef<typeof NavigationMenuPrimitive.Viewport>,  
 React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Viewport>  
>(({ className, ...props }, ref) => (  
 <div className={cn("absolute left-0 top-full flex justify-center")}>  
 <NavigationMenuPrimitive.Viewport  
 className={cn(  
 "origin-top-center relative mt-1.5 h-[var(--radix-navigation-menu-viewport-height)] w-full overflow-hidden rounded-md border bg-popover text-popover-foreground shadow-lg data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-90 md:w-[var(--radix-navigation-menu-viewport-width)]",  
 className  
 )}  
 ref={ref}  
 {...props}  
 />  
 </div>  
))  
NavigationMenuViewport.displayName =  
 NavigationMenuPrimitive.Viewport.displayName  
  
const NavigationMenuIndicator = React.forwardRef<  
 React.ElementRef<typeof NavigationMenuPrimitive.Indicator>,  
 React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Indicator>  
>(({ className, ...props }, ref) => (  
 <NavigationMenuPrimitive.Indicator  
 ref={ref}  
 className={cn(  
 "top-full z-[1] flex h-1.5 items-end justify-center overflow-hidden data-[state=visible]:animate-in data-[state=hidden]:animate-out data-[state=hidden]:fade-out data-[state=visible]:fade-in",  
 className  
 )}  
 {...props}  
 >  
 <div className="relative top-[60%] h-2 w-2 rotate-45 rounded-tl-sm bg-border shadow-md" />  
 </NavigationMenuPrimitive.Indicator>  
))  
NavigationMenuIndicator.displayName =  
 NavigationMenuPrimitive.Indicator.displayName  
  
export {  
 navigationMenuTriggerStyle,  
 NavigationMenu,  
 NavigationMenuList,  
 NavigationMenuItem,  
 NavigationMenuContent,  
 NavigationMenuTrigger,  
 NavigationMenuLink,  
 NavigationMenuIndicator,  
 NavigationMenuViewport,  
}

client/src/components/ui/pagination.tsx

import \* as React from "react"  
import { ChevronLeft, ChevronRight, MoreHorizontal } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
import { ButtonProps, buttonVariants } from "@/components/ui/button"  
  
const Pagination = ({ className, ...props }: React.ComponentProps<"nav">) => (  
 <nav  
 role="navigation"  
 aria-label="pagination"  
 className={cn("mx-auto flex w-full justify-center", className)}  
 {...props}  
 />  
)  
Pagination.displayName = "Pagination"  
  
const PaginationContent = React.forwardRef<  
 HTMLUListElement,  
 React.ComponentProps<"ul">  
>(({ className, ...props }, ref) => (  
 <ul  
 ref={ref}  
 className={cn("flex flex-row items-center gap-1", className)}  
 {...props}  
 />  
))  
PaginationContent.displayName = "PaginationContent"  
  
const PaginationItem = React.forwardRef<  
 HTMLLIElement,  
 React.ComponentProps<"li">  
>(({ className, ...props }, ref) => (  
 <li ref={ref} className={cn("", className)} {...props} />  
))  
PaginationItem.displayName = "PaginationItem"  
  
type PaginationLinkProps = {  
 isActive?: boolean  
} & Pick<ButtonProps, "size"> &  
 React.ComponentProps<"a">  
  
const PaginationLink = ({  
 className,  
 isActive,  
 size = "icon",  
 ...props  
}: PaginationLinkProps) => (  
 <a  
 aria-current={isActive ? "page" : undefined}  
 className={cn(  
 buttonVariants({  
 variant: isActive ? "outline" : "ghost",  
 size,  
 }),  
 className  
 )}  
 {...props}  
 />  
)  
PaginationLink.displayName = "PaginationLink"  
  
const PaginationPrevious = ({  
 className,  
 ...props  
}: React.ComponentProps<typeof PaginationLink>) => (  
 <PaginationLink  
 aria-label="Go to previous page"  
 size="default"  
 className={cn("gap-1 pl-2.5", className)}  
 {...props}  
 >  
 <ChevronLeft className="h-4 w-4" />  
 <span>Previous</span>  
 </PaginationLink>  
)  
PaginationPrevious.displayName = "PaginationPrevious"  
  
const PaginationNext = ({  
 className,  
 ...props  
}: React.ComponentProps<typeof PaginationLink>) => (  
 <PaginationLink  
 aria-label="Go to next page"  
 size="default"  
 className={cn("gap-1 pr-2.5", className)}  
 {...props}  
 >  
 <span>Next</span>  
 <ChevronRight className="h-4 w-4" />  
 </PaginationLink>  
)  
PaginationNext.displayName = "PaginationNext"  
  
const PaginationEllipsis = ({  
 className,  
 ...props  
}: React.ComponentProps<"span">) => (  
 <span  
 aria-hidden  
 className={cn("flex h-9 w-9 items-center justify-center", className)}  
 {...props}  
 >  
 <MoreHorizontal className="h-4 w-4" />  
 <span className="sr-only">More pages</span>  
 </span>  
)  
PaginationEllipsis.displayName = "PaginationEllipsis"  
  
export {  
 Pagination,  
 PaginationContent,  
 PaginationEllipsis,  
 PaginationItem,  
 PaginationLink,  
 PaginationNext,  
 PaginationPrevious,  
}

client/src/components/ui/popover.tsx

import \* as React from "react"  
import \* as PopoverPrimitive from "@radix-ui/react-popover"  
  
import { cn } from "@/lib/utils"  
  
const Popover = PopoverPrimitive.Root  
  
const PopoverTrigger = PopoverPrimitive.Trigger  
  
const PopoverContent = React.forwardRef<  
 React.ElementRef<typeof PopoverPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof PopoverPrimitive.Content>  
>(({ className, align = "center", sideOffset = 4, ...props }, ref) => (  
 <PopoverPrimitive.Portal>  
 <PopoverPrimitive.Content  
 ref={ref}  
 align={align}  
 sideOffset={sideOffset}  
 className={cn(  
 "z-50 w-72 rounded-md border bg-popover p-4 text-popover-foreground shadow-md outline-none data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-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}  
 />  
 </PopoverPrimitive.Portal>  
))  
PopoverContent.displayName = PopoverPrimitive.Content.displayName  
  
export { Popover, PopoverTrigger, PopoverContent }

client/src/components/ui/progress.tsx

import \* as React from "react"  
import \* as ProgressPrimitive from "@radix-ui/react-progress"  
  
import { cn } from "@/lib/utils"  
  
const Progress = React.forwardRef<  
 React.ElementRef<typeof ProgressPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof ProgressPrimitive.Root>  
>(({ className, value, ...props }, ref) => (  
 <ProgressPrimitive.Root  
 ref={ref}  
 className={cn(  
 "relative h-4 w-full overflow-hidden rounded-full bg-secondary",  
 className  
 )}  
 {...props}  
 >  
 <ProgressPrimitive.Indicator  
 className="h-full w-full flex-1 bg-primary transition-all"  
 style={{ transform: `translateX(-${100 - (value || 0)}%)` }}  
 />  
 </ProgressPrimitive.Root>  
))  
Progress.displayName = ProgressPrimitive.Root.displayName  
  
export { Progress }

client/src/components/ui/radio-group.tsx

import \* as React from "react"  
import \* as RadioGroupPrimitive from "@radix-ui/react-radio-group"  
import { Circle } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
const RadioGroup = React.forwardRef<  
 React.ElementRef<typeof RadioGroupPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof RadioGroupPrimitive.Root>  
>(({ className, ...props }, ref) => {  
 return (  
 <RadioGroupPrimitive.Root  
 className={cn("grid gap-2", className)}  
 {...props}  
 ref={ref}  
 />  
 )  
})  
RadioGroup.displayName = RadioGroupPrimitive.Root.displayName  
  
const RadioGroupItem = React.forwardRef<  
 React.ElementRef<typeof RadioGroupPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof RadioGroupPrimitive.Item>  
>(({ className, ...props }, ref) => {  
 return (  
 <RadioGroupPrimitive.Item  
 ref={ref}  
 className={cn(  
 "aspect-square h-4 w-4 rounded-full border border-primary text-primary ring-offset-background focus:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50",  
 className  
 )}  
 {...props}  
 >  
 <RadioGroupPrimitive.Indicator className="flex items-center justify-center">  
 <Circle className="h-2.5 w-2.5 fill-current text-current" />  
 </RadioGroupPrimitive.Indicator>  
 </RadioGroupPrimitive.Item>  
 )  
})  
RadioGroupItem.displayName = RadioGroupPrimitive.Item.displayName  
  
export { RadioGroup, RadioGroupItem }

client/src/components/ui/resizable.tsx

import { GripVertical } from "lucide-react"  
import \* as ResizablePrimitive from "react-resizable-panels"  
  
import { cn } from "@/lib/utils"  
  
const ResizablePanelGroup = ({  
 className,  
 ...props  
}: React.ComponentProps<typeof ResizablePrimitive.PanelGroup>) => (  
 <ResizablePrimitive.PanelGroup  
 className={cn(  
 "flex h-full w-full data-[panel-group-direction=vertical]:flex-col",  
 className  
 )}  
 {...props}  
 />  
)  
  
const ResizablePanel = ResizablePrimitive.Panel  
  
const ResizableHandle = ({  
 withHandle,  
 className,  
 ...props  
}: React.ComponentProps<typeof ResizablePrimitive.PanelResizeHandle> & {  
 withHandle?: boolean  
}) => (  
 <ResizablePrimitive.PanelResizeHandle  
 className={cn(  
 "relative flex w-px items-center justify-center bg-border after:absolute after:inset-y-0 after:left-1/2 after:w-1 after:-translate-x-1/2 focus-visible:outline-none focus-visible:ring-1 focus-visible:ring-ring focus-visible:ring-offset-1 data-[panel-group-direction=vertical]:h-px data-[panel-group-direction=vertical]:w-full data-[panel-group-direction=vertical]:after:left-0 data-[panel-group-direction=vertical]:after:h-1 data-[panel-group-direction=vertical]:after:w-full data-[panel-group-direction=vertical]:after:-translate-y-1/2 data-[panel-group-direction=vertical]:after:translate-x-0 [&[data-panel-group-direction=vertical]>div]:rotate-90",  
 className  
 )}  
 {...props}  
 >  
 {withHandle && (  
 <div className="z-10 flex h-4 w-3 items-center justify-center rounded-sm border bg-border">  
 <GripVertical className="h-2.5 w-2.5" />  
 </div>  
 )}  
 </ResizablePrimitive.PanelResizeHandle>  
)  
  
export { ResizablePanelGroup, ResizablePanel, ResizableHandle }

client/src/components/ui/scroll-area.tsx

import \* as React from "react"  
import \* as ScrollAreaPrimitive from "@radix-ui/react-scroll-area"  
  
import { cn } from "@/lib/utils"  
  
const ScrollArea = React.forwardRef<  
 React.ElementRef<typeof ScrollAreaPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.Root>  
>(({ className, children, ...props }, ref) => (  
 <ScrollAreaPrimitive.Root  
 ref={ref}  
 className={cn("relative overflow-hidden", className)}  
 {...props}  
 >  
 <ScrollAreaPrimitive.Viewport className="h-full w-full rounded-[inherit]">  
 {children}  
 </ScrollAreaPrimitive.Viewport>  
 <ScrollBar />  
 <ScrollAreaPrimitive.Corner />  
 </ScrollAreaPrimitive.Root>  
))  
ScrollArea.displayName = ScrollAreaPrimitive.Root.displayName  
  
const ScrollBar = React.forwardRef<  
 React.ElementRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>,  
 React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>  
>(({ className, orientation = "vertical", ...props }, ref) => (  
 <ScrollAreaPrimitive.ScrollAreaScrollbar  
 ref={ref}  
 orientation={orientation}  
 className={cn(  
 "flex touch-none select-none transition-colors",  
 orientation === "vertical" &&  
 "h-full w-2.5 border-l border-l-transparent p-[1px]",  
 orientation === "horizontal" &&  
 "h-2.5 flex-col border-t border-t-transparent p-[1px]",  
 className  
 )}  
 {...props}  
 >  
 <ScrollAreaPrimitive.ScrollAreaThumb className="relative flex-1 rounded-full bg-border" />  
 </ScrollAreaPrimitive.ScrollAreaScrollbar>  
))  
ScrollBar.displayName = ScrollAreaPrimitive.ScrollAreaScrollbar.displayName  
  
export { ScrollArea, ScrollBar }

client/src/components/ui/select.tsx

import \* as React from "react"  
import \* as SelectPrimitive from "@radix-ui/react-select"  
import { Check, ChevronDown, ChevronUp } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
const Select = SelectPrimitive.Root  
  
const SelectGroup = SelectPrimitive.Group  
  
const SelectValue = SelectPrimitive.Value  
  
const SelectTrigger = React.forwardRef<  
 React.ElementRef<typeof SelectPrimitive.Trigger>,  
 React.ComponentPropsWithoutRef<typeof SelectPrimitive.Trigger>  
>(({ className, children, ...props }, ref) => (  
 <SelectPrimitive.Trigger  
 ref={ref}  
 className={cn(  
 "flex h-10 w-full items-center justify-between rounded-md border border-input bg-background px-3 py-2 text-sm ring-offset-background placeholder:text-muted-foreground focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50 [&>span]:line-clamp-1",  
 className  
 )}  
 {...props}  
 >  
 {children}  
 <SelectPrimitive.Icon asChild>  
 <ChevronDown className="h-4 w-4 opacity-50" />  
 </SelectPrimitive.Icon>  
 </SelectPrimitive.Trigger>  
))  
SelectTrigger.displayName = SelectPrimitive.Trigger.displayName  
  
const SelectScrollUpButton = React.forwardRef<  
 React.ElementRef<typeof SelectPrimitive.ScrollUpButton>,  
 React.ComponentPropsWithoutRef<typeof SelectPrimitive.ScrollUpButton>  
>(({ className, ...props }, ref) => (  
 <SelectPrimitive.ScrollUpButton  
 ref={ref}  
 className={cn(  
 "flex cursor-default items-center justify-center py-1",  
 className  
 )}  
 {...props}  
 >  
 <ChevronUp className="h-4 w-4" />  
 </SelectPrimitive.ScrollUpButton>  
))  
SelectScrollUpButton.displayName = SelectPrimitive.ScrollUpButton.displayName  
  
const SelectScrollDownButton = React.forwardRef<  
 React.ElementRef<typeof SelectPrimitive.ScrollDownButton>,  
 React.ComponentPropsWithoutRef<typeof SelectPrimitive.ScrollDownButton>  
>(({ className, ...props }, ref) => (  
 <SelectPrimitive.ScrollDownButton  
 ref={ref}  
 className={cn(  
 "flex cursor-default items-center justify-center py-1",  
 className  
 )}  
 {...props}  
 >  
 <ChevronDown className="h-4 w-4" />  
 </SelectPrimitive.ScrollDownButton>  
))  
SelectScrollDownButton.displayName =  
 SelectPrimitive.ScrollDownButton.displayName  
  
const SelectContent = React.forwardRef<  
 React.ElementRef<typeof SelectPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof SelectPrimitive.Content>  
>(({ className, children, position = "popper", ...props }, ref) => (  
 <SelectPrimitive.Portal>  
 <SelectPrimitive.Content  
 ref={ref}  
 className={cn(  
 "relative z-50 max-h-96 min-w-[8rem] overflow-hidden rounded-md border bg-popover text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-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",  
 position === "popper" &&  
 "data-[side=bottom]:translate-y-1 data-[side=left]:-translate-x-1 data-[side=right]:translate-x-1 data-[side=top]:-translate-y-1",  
 className  
 )}  
 position={position}  
 {...props}  
 >  
 <SelectScrollUpButton />  
 <SelectPrimitive.Viewport  
 className={cn(  
 "p-1",  
 position === "popper" &&  
 "h-[var(--radix-select-trigger-height)] w-full min-w-[var(--radix-select-trigger-width)]"  
 )}  
 >  
 {children}  
 </SelectPrimitive.Viewport>  
 <SelectScrollDownButton />  
 </SelectPrimitive.Content>  
 </SelectPrimitive.Portal>  
))  
SelectContent.displayName = SelectPrimitive.Content.displayName  
  
const SelectLabel = React.forwardRef<  
 React.ElementRef<typeof SelectPrimitive.Label>,  
 React.ComponentPropsWithoutRef<typeof SelectPrimitive.Label>  
>(({ className, ...props }, ref) => (  
 <SelectPrimitive.Label  
 ref={ref}  
 className={cn("py-1.5 pl-8 pr-2 text-sm font-semibold", className)}  
 {...props}  
 />  
))  
SelectLabel.displayName = SelectPrimitive.Label.displayName  
  
const SelectItem = React.forwardRef<  
 React.ElementRef<typeof SelectPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof SelectPrimitive.Item>  
>(({ className, children, ...props }, ref) => (  
 <SelectPrimitive.Item  
 ref={ref}  
 className={cn(  
 "relative flex w-full cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className  
 )}  
 {...props}  
 >  
 <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">  
 <SelectPrimitive.ItemIndicator>  
 <Check className="h-4 w-4" />  
 </SelectPrimitive.ItemIndicator>  
 </span>  
  
 <SelectPrimitive.ItemText>{children}</SelectPrimitive.ItemText>  
 </SelectPrimitive.Item>  
))  
SelectItem.displayName = SelectPrimitive.Item.displayName  
  
const SelectSeparator = React.forwardRef<  
 React.ElementRef<typeof SelectPrimitive.Separator>,  
 React.ComponentPropsWithoutRef<typeof SelectPrimitive.Separator>  
>(({ className, ...props }, ref) => (  
 <SelectPrimitive.Separator  
 ref={ref}  
 className={cn("-mx-1 my-1 h-px bg-muted", className)}  
 {...props}  
 />  
))  
SelectSeparator.displayName = SelectPrimitive.Separator.displayName  
  
export {  
 Select,  
 SelectGroup,  
 SelectValue,  
 SelectTrigger,  
 SelectContent,  
 SelectLabel,  
 SelectItem,  
 SelectSeparator,  
 SelectScrollUpButton,  
 SelectScrollDownButton,  
}

client/src/components/ui/separator.tsx

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 }

client/src/components/ui/sheet.tsx

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,  
}

client/src/components/ui/sidebar.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 } 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)) => {  
 if (setOpenProp) {  
 return setOpenProp?.(  
 typeof value === "function" ? value(open) : value  
 )  
 }  
  
 \_setOpen(value)  
  
 // This sets the cookie to keep the sidebar state.  
 document.cookie = `${SIDEBAR\_COOKIE\_NAME}=${open}; 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 text-sidebar-foreground 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"  
 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,opa] 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,  
}

client/src/components/ui/skeleton.tsx

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 }