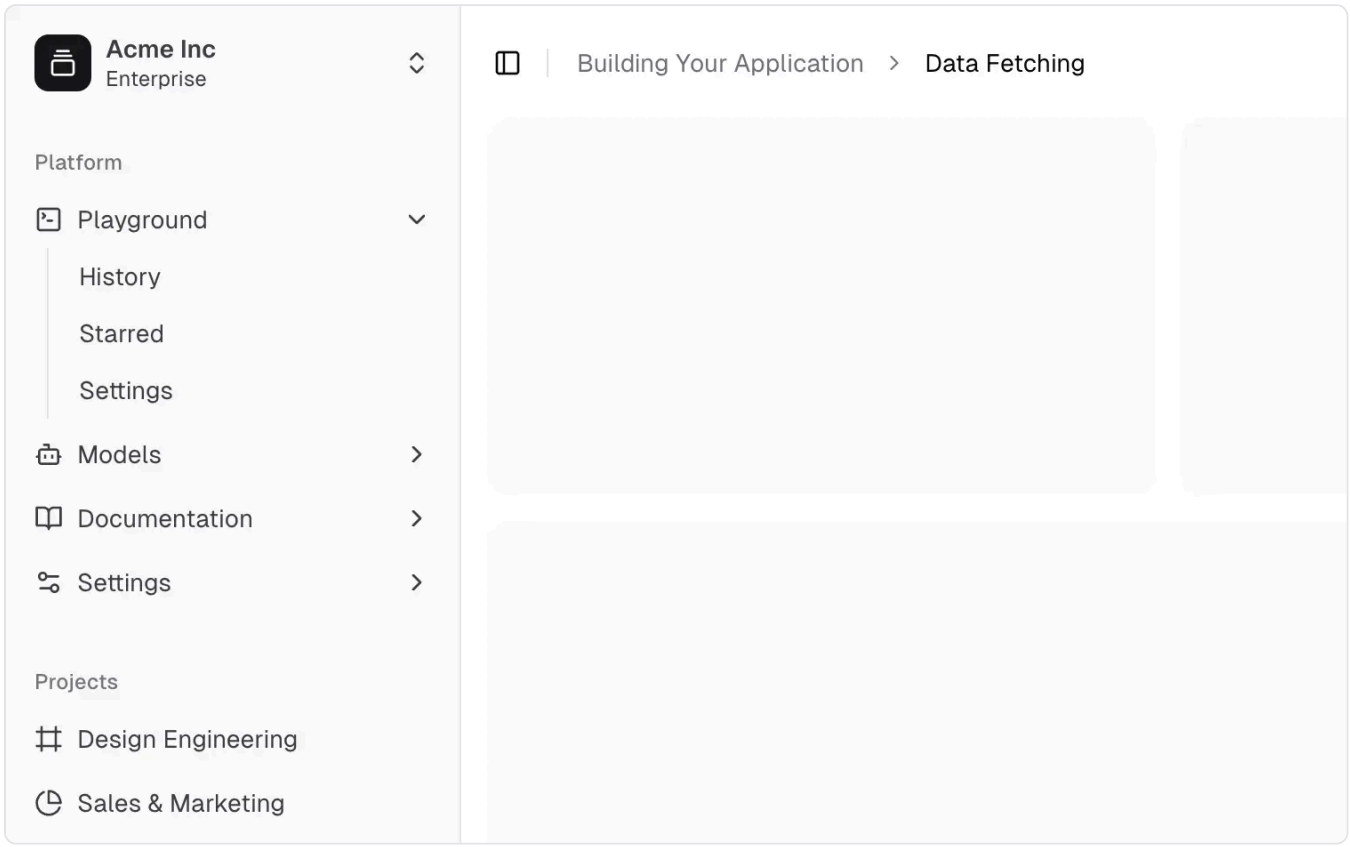
Docs > Sidebar

Sidebar

A composable, themeable and customizable sidebar component.



A sidebar that collapses to icons.

Sidebars are one of the most complex components to build. They are central to any application and often contain a lot of moving parts.

I don't like building sidebars. So I built 30+ of them. All kinds of configurations. Then I extracted the core components into `sidebar.tsx`.

We now have a solid foundation to build on top of. Composable. Themeable. Customizable.

Browse the Blocks Library.

Installation

CLI Manual

1 **Run the following command to install `sidebar.tsx`**

pnpm npm yarn bun

```
npx shadcn@latest add sidebar
```

2 **Add the following colors to your CSS file**

The command above should install the colors for you. If not, copy and paste the following in your CSS file.

We'll go over the colors later in the theming section.

app/globals.css

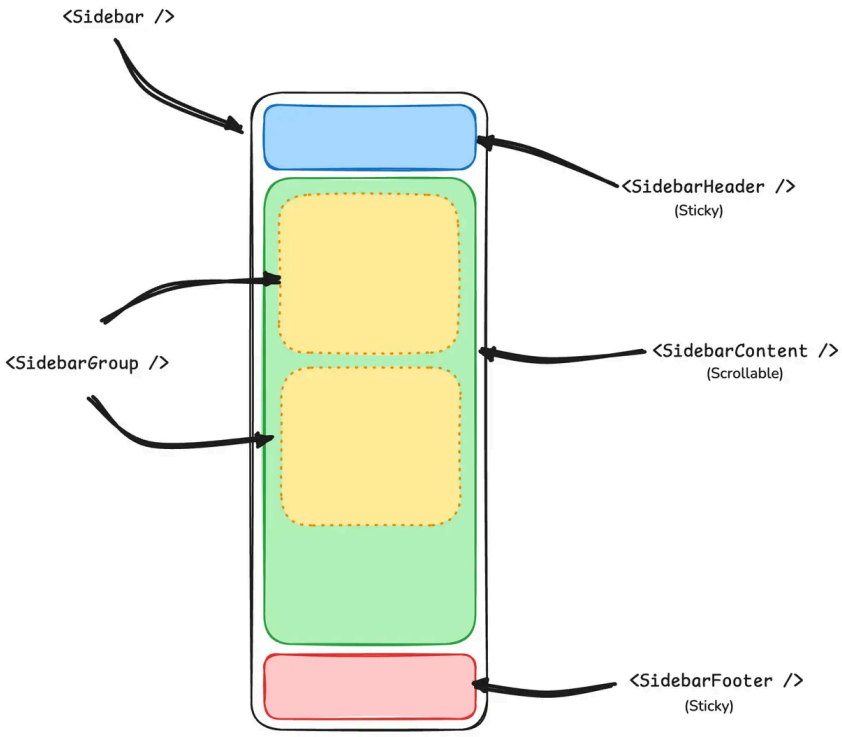
```
@layer base {
  :root {
    --sidebar-background: 0 0% 98%;
    --sidebar-foreground: 240 5.3% 26.1%;
    --sidebar-primary: 240 5.9% 10%;
    --sidebar-primary-foreground: 0 0% 98%;
    --sidebar-accent: 240 4.8% 95.9%;
    --sidebar-accent-foreground: 240 5.9% 10%;
    --sidebar-border: 220 13%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

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

Structure

A `Sidebar` component is composed of the following parts:

- `SidebarProvider` - Handles collapsible state.
- `Sidebar` - The sidebar container.
- `SidebarHeader` and `SidebarFooter` - Sticky at the top and bottom of the sidebar.
- `SidebarContent` - Scrollable content.
- `SidebarGroup` - Section within the `SidebarContent`.
- `SidebarTrigger` - Trigger for the `Sidebar`.



Usage

app/layout.tsx

```tsx
import { SidebarProvider, SidebarTrigger } from "@/components/ui/sidebar"
import { AppSidebar } from "@/components/app-sidebar"

export default function Layout({ children }: { children: React.ReactNode
  return (
```

```
      <SidebarProvider>
        <AppSidebar />
        <main>
          <SidebarTrigger />
          {children}
        </main>
      </SidebarProvider>
    )
  }
```

components/app-sidebar.tsx

```
import {
  Sidebar,
  SidebarContent,
  SidebarFooter,
  SidebarGroup,
  SidebarHeader,
} from "@/components/ui/sidebar"

export function AppSidebar() {
  return (
    <Sidebar>
      <SidebarHeader />
      <SidebarContent>
        <SidebarGroup />
        <SidebarGroup />
      </SidebarContent>
      <SidebarFooter />
    </Sidebar>
  )
}
```

Your First Sidebar

Let's start with the most basic sidebar. A collapsible sidebar with a menu.

1 **Add a `SidebarProvider` and `SidebarTrigger` at the root of your application.**

app/layout.tsx

```tsx
import { SidebarProvider, SidebarTrigger } from "@/components/ui/si
import { AppSidebar } from "@/components/app-sidebar"

export default function Layout({ children }: { children: React.Reac
  return (
    <SidebarProvider>
      <AppSidebar />
      <main>
        <SidebarTrigger />
        {children}
      </main>
    </SidebarProvider>
  )
}
```

2 Create a new sidebar component at `components/app-sidebar.tsx`.

components/app-sidebar.tsx

```tsx
import { Sidebar, SidebarContent } from "@/components/ui/sidebar"

export function AppSidebar() {
  return (
    <Sidebar>
      <SidebarContent />
    </Sidebar>
  )
}
```

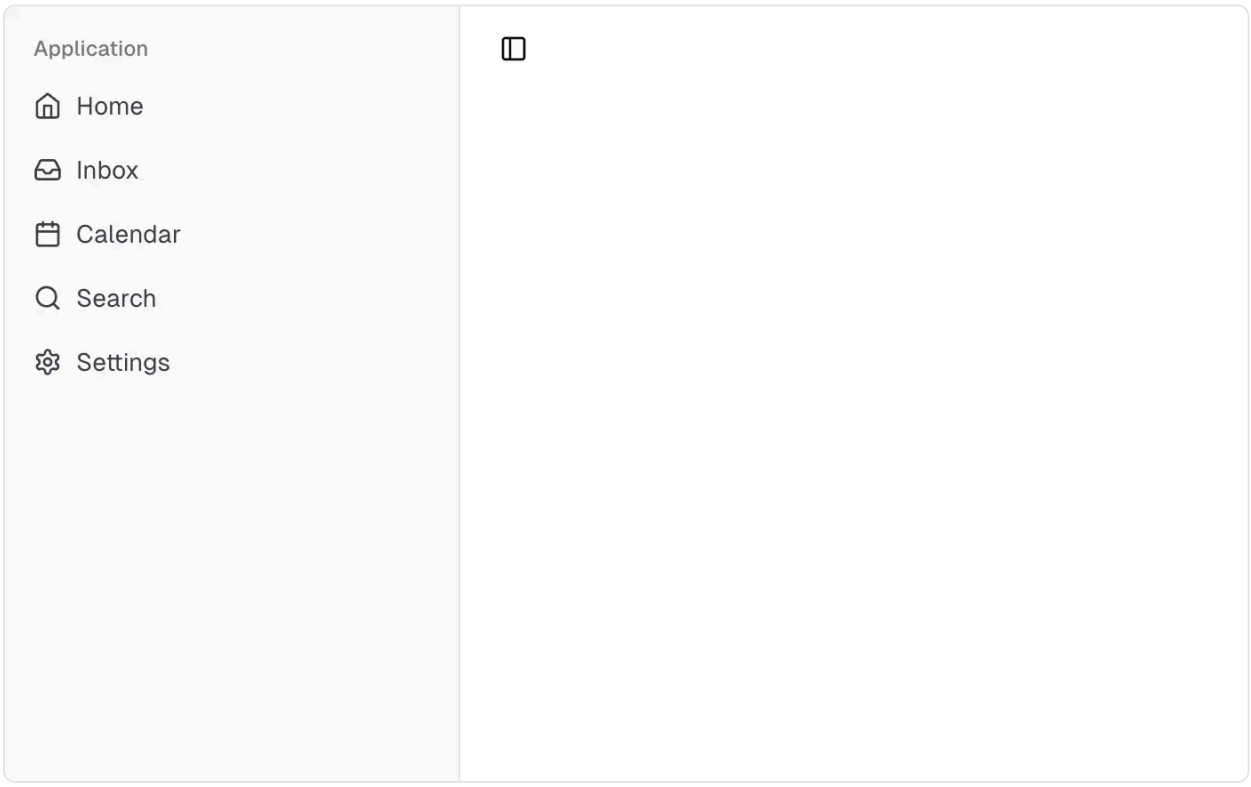
3 Now, let's add a `SidebarMenu` to the sidebar.

We'll use the `SidebarMenu` component in a `SidebarGroup`.

components/app-sidebar.tsx

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

4 You've created your first sidebar.



Your first sidebar.

Components

The components in `sidebar.tsx` are built to be composable i.e you build your sidebar by putting the provided components together. They also compose well with other shadcn/ui components such as `DropdownMenu`, `Collapsible` or `Dialog` etc.

If you need to change the code in `sidebar.tsx`, you are encouraged to do so. The code is yours. Use `sidebar.tsx` as a starting point and build your own.

In the next sections, we'll go over each component and how to use them.

SidebarProvider

The `SidebarProvider` component is used to provide the sidebar context to the `Sidebar` component. You should always wrap your application in a `SidebarProvider` component.

Props

Name	Type	Description
`defaultOpen`	`boolean`	Default open state of the sidebar.
`open`	`boolean`	Open state of the sidebar (controlled).
`onOpenChange`	`(open: boolean) => void`	Sets open state of the sidebar (controlled).

Width

If you have a single sidebar in your application, you can use the `SIDEBAR_WIDTH` and `SIDEBAR_WIDTH_MOBILE` variables in `sidebar.tsx` to set the width of the sidebar.

components/ui/sidebar.tsx

```
const SIDEBAR_WIDTH = "16rem"
```

```
const SIDEBAR_WIDTH_MOBILE = "18rem"
```

For multiple sidebars in your application, you can use the `style` prop to set the width of the sidebar.

To set the width of the sidebar, you can use the `--sidebar-width` and `--sidebar-width-mobile` CSS variables in the `style` prop.

components/ui/sidebar.tsx

```
<SidebarProvider
  style={{
    "--sidebar-width": "20rem",
    "--sidebar-width-mobile": "20rem",
  }}
>
  <Sidebar />
</SidebarProvider>
```

This will handle the width of the sidebar but also the layout spacing.

Keyboard Shortcut

The `SIDEBAR_KEYBOARD_SHORTCUT` variable is used to set the keyboard shortcut used to open and close the sidebar.

To trigger the sidebar, you use the `cmd+b` keyboard shortcut on Mac and `ctrl+b` on Windows.

You can change the keyboard shortcut by updating the `SIDEBAR_KEYBOARD_SHORTCUT` variable.

components/ui/sidebar.tsx

```
const SIDEBAR_KEYBOARD_SHORTCUT = "b"
```

Persisted State

The `SidebarProvider` supports persisting the sidebar state across page reloads and server-side rendering. It uses cookies to store the current state of the sidebar. When the sidebar state changes, a default cookie named `sidebar_state` is set with the current open/closed state. This cookie is then read on subsequent page loads to restore the sidebar state.

To persist sidebar state in Next.js, set up your `SidebarProvider` in `app/layout.tsx` like this:

app/layout.tsx

```tsx
import { cookies } from "next/headers"

import { SidebarProvider, SidebarTrigger } from "@/components/ui/sidebar"
import { AppSidebar } from "@/components/app-sidebar"

export async function Layout({ children }: { children: React.ReactNode })
  const cookieStore = await cookies()
  const defaultOpen = cookieStore.get("sidebar_state")?.value === "true"

  return (
    <SidebarProvider defaultOpen={defaultOpen}>
      <AppSidebar />
      <main>
        <SidebarTrigger />
        {children}
      </main>
    </SidebarProvider>
  )
}
```

You can change the name of the cookie by updating the `SIDEBAR_COOKIE_NAME` variable in `sidebar.tsx`.

components/ui/sidebar.tsx

```tsx
const SIDEBAR_COOKIE_NAME = "sidebar_state"
```

Sidebar

The main `Sidebar` component used to render a collapsible sidebar.

```tsx
import { Sidebar } from "@/components/ui/sidebar"

export function AppSidebar() {
  return <Sidebar />
}
```

Props

Property	Type	Description
`side`	`left` or `right`	The side of the sidebar.
`variant`	`sidebar`, `floating`, or `inset`	The variant of the sidebar.
`collapsible`	`offcanvas`, `icon`, or `none`	Collapsible state of the sidebar.

side

Use the `side` prop to change the side of the sidebar.

Available options are `left` and `right`.

```tsx
import { Sidebar } from "@/components/ui/sidebar"

export function AppSidebar() {
  return <Sidebar side="left | right" />
}
```

variant

Use the `variant` prop to change the variant of the sidebar.

Available options are `sidebar`, `floating` and `inset`.

```tsx
import { Sidebar } from "@/components/ui/sidebar"

export function AppSidebar() {
  return <Sidebar variant="sidebar | floating | inset" />
}
```

Note: If you use the `inset` variant, remember to wrap your main content in a `SidebarInset` component.

```tsx
<SidebarProvider>
  <Sidebar variant="inset" />
  <SidebarInset>
    <main>{children}</main>
  </SidebarInset>
</SidebarProvider>
```

collapsible

Use the `collapsible` prop to make the sidebar collapsible.

Available options are `offcanvas`, `icon` and `none`.

```tsx
import { Sidebar } from "@/components/ui/sidebar"

export function AppSidebar() {
  return <Sidebar collapsible="offcanvas | icon | none" />
}
```

Prop	Description
`offcanvas`	A collapsible sidebar that slides in from the left or right.
`icon`	A sidebar that collapses to icons.
`none`	A non-collapsible sidebar.

useSidebar

The `useSidebar` hook is used to control the sidebar.

```tsx
import { useSidebar } from "@/components/ui/sidebar"

export function AppSidebar() {
  const {
    state,
    open,
    setOpen,
    openMobile,
    setOpenMobile,
    isMobile,
    toggleSidebar,
  } = useSidebar()
}
```

Property	Type	Description
`state`	`expanded or collapsed`	The current state of the sidebar.
`open`	`boolean`	Whether the sidebar is open.
`setOpen`	`(open: boolean) => void`	Sets the open state of the sidebar.
`openMobile`	`boolean`	Whether the sidebar is open on mobile.
`setOpenMobile`	`(open: boolean) => void`	Sets the open state of the sidebar on mobile.
`isMobile`	`boolean`	Whether the sidebar is on mobile.
`toggleSidebar`	`() => void`	Toggles the sidebar. Desktop and mobile.

SidebarHeader

Use the `SidebarHeader` component to add a sticky header to the sidebar.

The following example adds a `<DropdownMenu>` to the `SidebarHeader`.

A sidebar header with a dropdown menu.

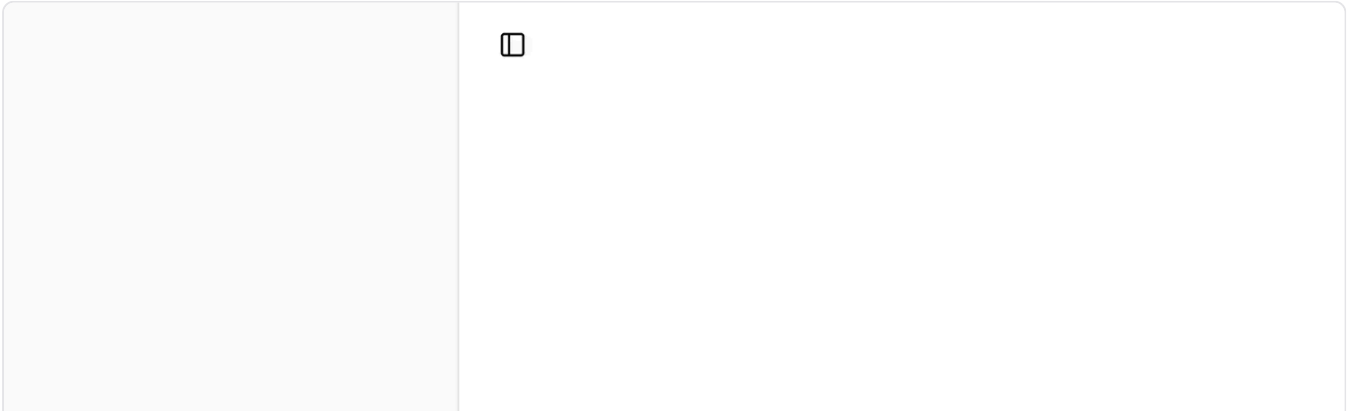
components/app-sidebar.tsx

```tsx
<Sidebar>
  <SidebarHeader>
    <SidebarMenu>
      <SidebarMenuItem>
        <DropdownMenu>
          <DropdownMenuTrigger asChild>
            <SidebarMenuButton>
              Select Workspace
              <ChevronDown className="ml-auto" />
            </SidebarMenuButton>
          </DropdownMenuTrigger>
          <DropdownMenuContent className="w-[--radix-popper-anchor-width]
            <DropdownMenuItem>
              <span>Acme Inc</span>
            </DropdownMenuItem>
            <DropdownMenuItem>
              <span>Acme Corp.</span>
            </DropdownMenuItem>
          </DropdownMenuContent>
        </DropdownMenu>
      </SidebarMenuItem>
    </SidebarMenu>
  </SidebarHeader>
</Sidebar>
```

SidebarFooter

Use the `SidebarFooter` component to add a sticky footer to the sidebar.

The following example adds a `<DropdownMenu>` to the `SidebarFooter`.



A sidebar footer with a dropdown menu.

components/app-sidebar.tsx

```tsx
export function AppSidebar() {
  return (
    <SidebarProvider>
      <Sidebar>
        <SidebarHeader />
        <SidebarContent />
        <SidebarFooter>
          <SidebarMenu>
            <SidebarMenuItem>
              <DropdownMenu>
                <DropdownMenuTrigger asChild>
                  <SidebarMenuButton>
                    <User2 /> Username
                    <ChevronUp className="ml-auto" />
                  </SidebarMenuButton>
                </DropdownMenuTrigger>
                <DropdownMenuContent
                  side="top"
                  className="w-[--radix-popper-anchor-width]"
                >
                  <DropdownMenuItem>
                    <span>Account</span>
                  </DropdownMenuItem>
                  <DropdownMenuItem>
                    <span>Billing</span>
                  </DropdownMenuItem>
```

SidebarContent

The `SidebarContent` component is used to wrap the content of the sidebar. This is where you add your `SidebarGroup` components. It is scrollable.

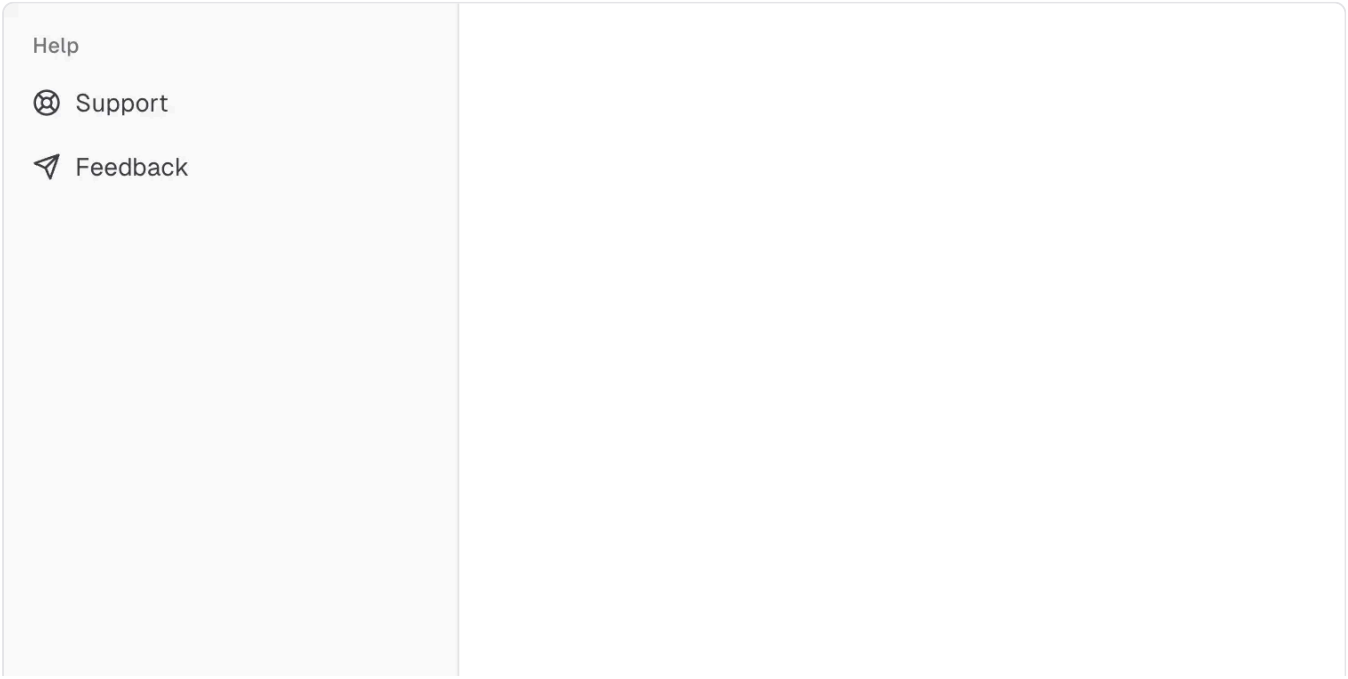
```tsx
import { Sidebar, SidebarContent } from "@/components/ui/sidebar"

export function AppSidebar() {
  return (
    <Sidebar>
      <SidebarContent>
        <SidebarGroup />
        <SidebarGroup />
      </SidebarContent>
    </Sidebar>
  )
}
```

SidebarGroup

Use the `SidebarGroup` component to create a section within the sidebar.

A `SidebarGroup` has a `SidebarGroupLabel`, a `SidebarGroupContent` and an optional `SidebarGroupAction`.



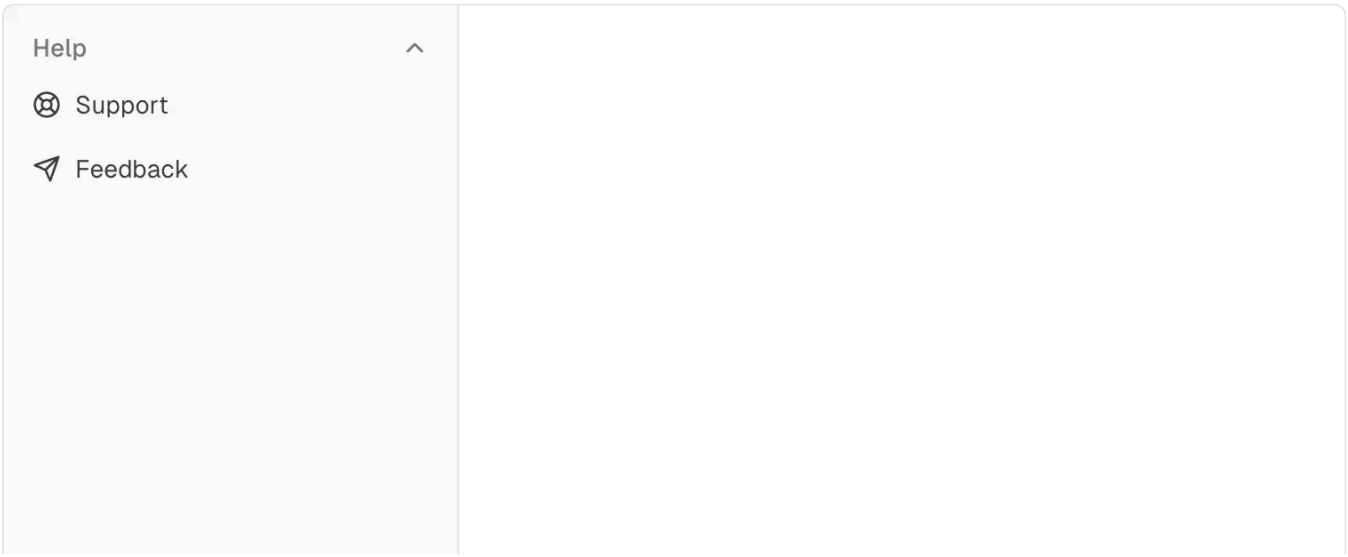
A sidebar group.

```tsx
import { Sidebar, SidebarContent, SidebarGroup } from "@/components/ui/si

export function AppSidebar() {
  return (
    <Sidebar>
      <SidebarContent>
        <SidebarGroup>
          <SidebarGroupLabel>Application</SidebarGroupLabel>
          <SidebarGroupAction>
            <Plus /> <span className="sr-only">Add Project</span>
          </SidebarGroupAction>
          <SidebarGroupContent></SidebarGroupContent>
        </SidebarGroup>
      </SidebarContent>
    </Sidebar>
  )
}
```

Collapsible SidebarGroup

To make a `SidebarGroup` collapsible, wrap it in a `Collapsible`.



A collapsible sidebar group.

```tsx
export function AppSidebar() {
  return (
    <Collapsible defaultOpen className="group/collapsible">
      <SidebarGroup>
        <SidebarGroupLabel asChild>
          <CollapsibleTrigger>
            Help
            <ChevronDown className="ml-auto transition-transform group-da
          </CollapsibleTrigger>
        </SidebarGroupLabel>
        <CollapsibleContent>
          <SidebarGroupContent />
        </CollapsibleContent>
      </SidebarGroup>
    </Collapsible>
  )
}
```

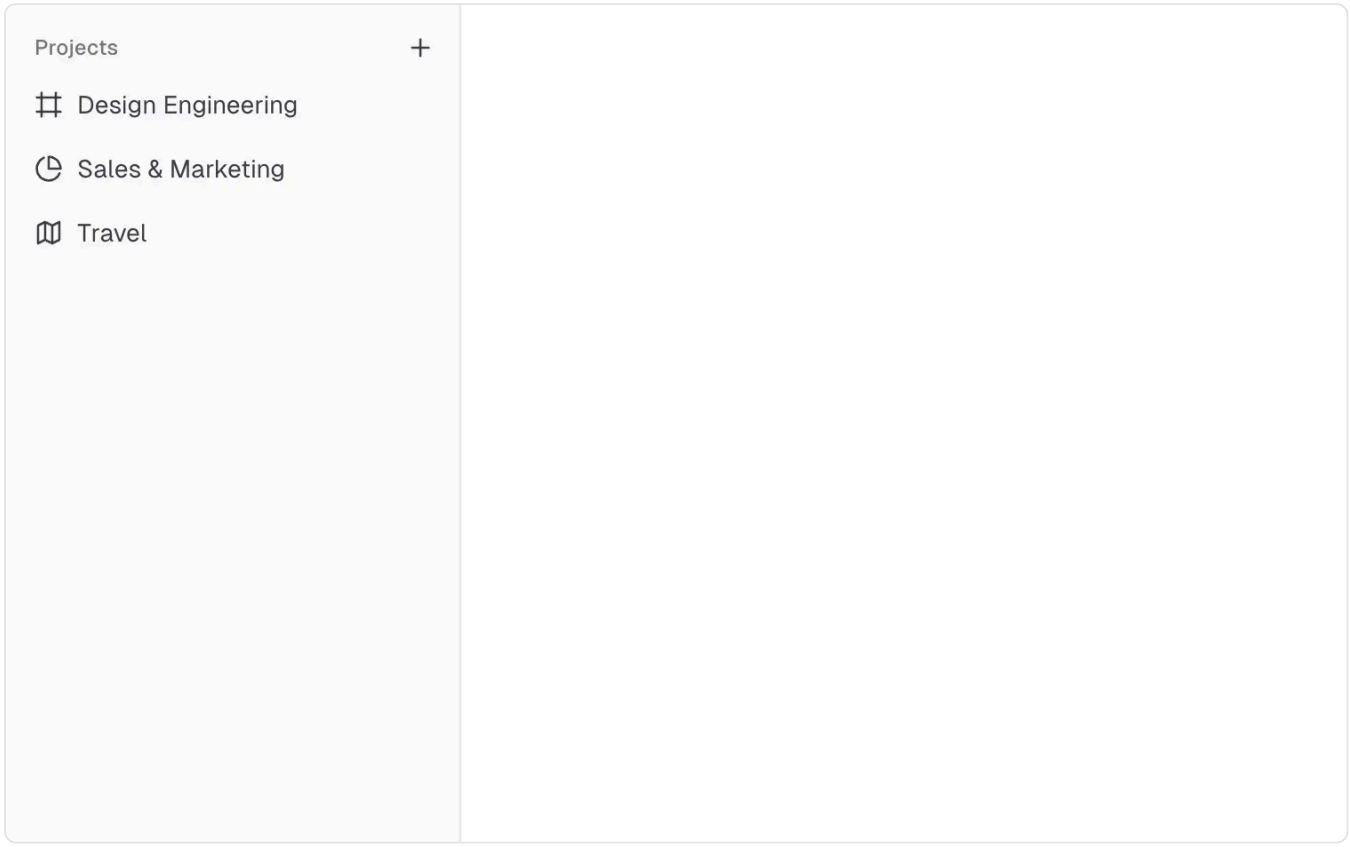
Note: We wrap the `CollapsibleTrigger` in a `SidebarGroupLabel` to render a button.

SidebarGroupAction

Use the `SidebarGroupAction` component to add an action button to the `SidebarGroup`.

```tsx
export function AppSidebar() {
  return (
    <SidebarGroup>
      <SidebarGroupLabel asChild>Projects</SidebarGroupLabel>
      <SidebarGroupAction title="Add Project">
```

```
          <Plus /> <span className="sr-only">Add Project</span>
        </SidebarGroupAction>
        <SidebarGroupContent />
      </SidebarGroup>
    )
  }
```

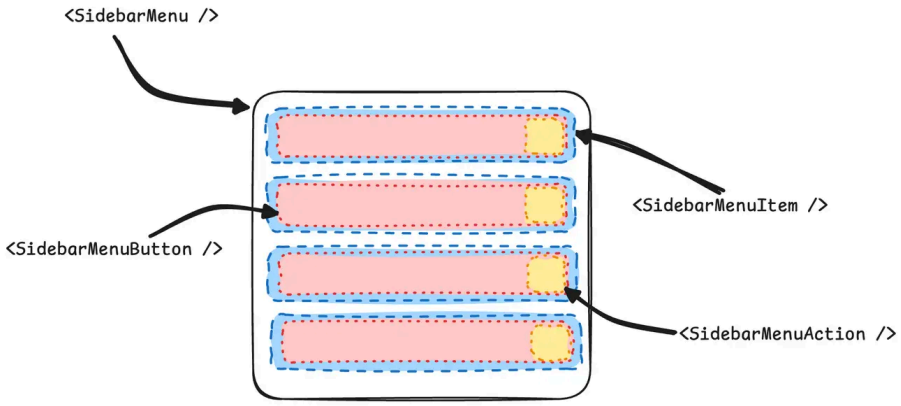


A sidebar group with an action button.

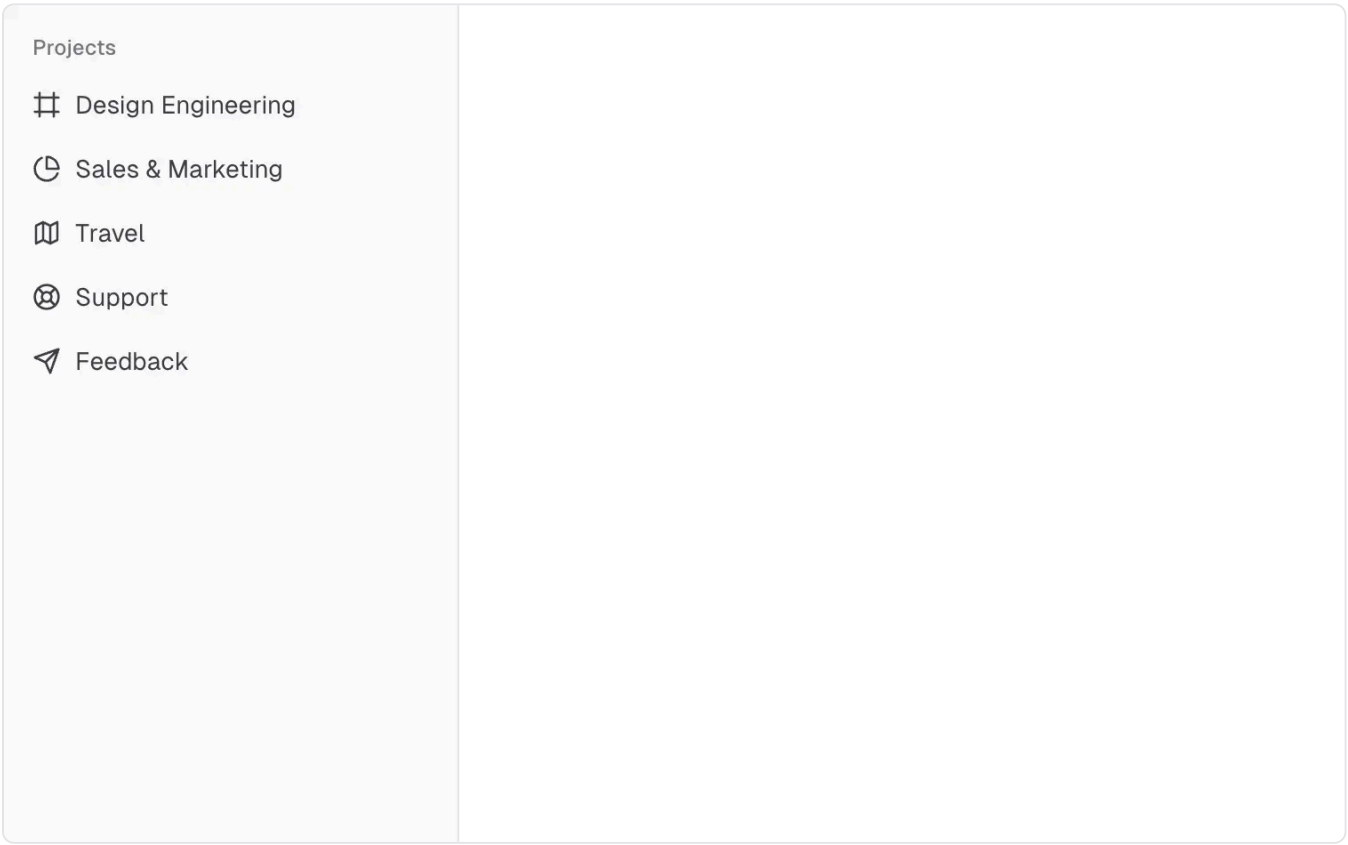
SidebarMenu

The `SidebarMenu` component is used for building a menu within a `SidebarGroup`.

A `SidebarMenu` component is composed of `SidebarMenuItem`, `SidebarMenuButton`, `<SidebarMenuAction />` and `<SidebarMenuSub />` components.



Here's an example of a `SidebarMenu` component rendering a list of projects.



A sidebar menu with a list of projects.

```
<Sidebar>
```

```
    <SidebarContent>
      <SidebarGroup>
        <SidebarGroupLabel>Projects</SidebarGroupLabel>
        <SidebarGroupContent>
          <SidebarMenu>
            {projects.map((project) => (
              <SidebarMenuItem key={project.name}>
                <SidebarMenuButton asChild>
                  <a href={project.url}>
                    <project.icon />
                    <span>{project.name}</span>
                  </a>
                </SidebarMenuButton>
              </SidebarMenuItem>
            ))}
          </SidebarMenu>
        </SidebarGroupContent>
      </SidebarGroup>
    </SidebarContent>
  </Sidebar>
```

SidebarMenuButton

The `SidebarMenuButton` component is used to render a menu button within a `SidebarMenuItem`.

Link or Anchor

By default, the `SidebarMenuButton` renders a button but you can use the `asChild` prop to render a different component such as a `Link` or an `a` tag.

```
<SidebarMenuButton asChild>
  <a href="#">Home</a>
</SidebarMenuButton>
```

Icon and Label

You can render an icon and a truncated label inside the button. Remember to wrap the label in a `<span>`.

```
<SidebarMenuButton asChild>
  <a href="#">
    <Home />
    <span>Home</span>
  </a>
</SidebarMenuButton>
```

isActive

Use the `isActive` prop to mark a menu item as active.

```
<SidebarMenuButton asChild isActive>
  <a href="#">Home</a>
</SidebarMenuButton>
```

SidebarMenuAction

The `SidebarMenuAction` component is used to render a menu action within a `SidebarMenuItem`.

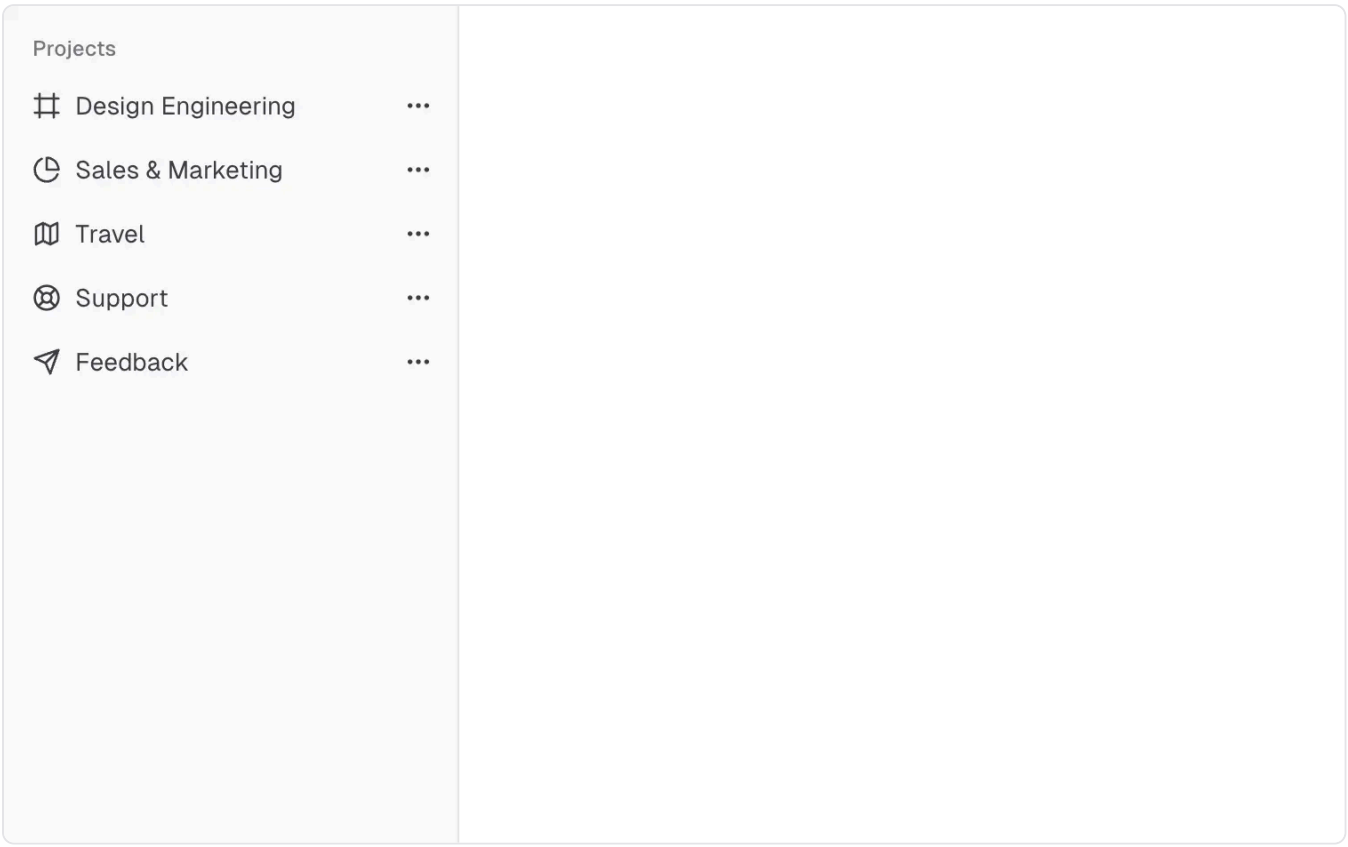
This button works independently of the `SidebarMenuButton` i.e you can have the `<SidebarMenuButton />` as a clickable link and the `<SidebarMenuAction />` as a button.

```
<SidebarMenuItem>
  <SidebarMenuButton asChild>
    <a href="#">
      <Home />
      <span>Home</span>
    </a>
  </SidebarMenuButton>
  <SidebarMenuAction>
    <Plus /> <span className="sr-only">Add Project</span>
  </SidebarMenuAction>
```

```
</SidebarMenuItem>
```

DropdownMenu

Here's an example of a `SidebarMenuAction` component rendering a `DropdownMenu`.



A sidebar menu action with a dropdown menu.

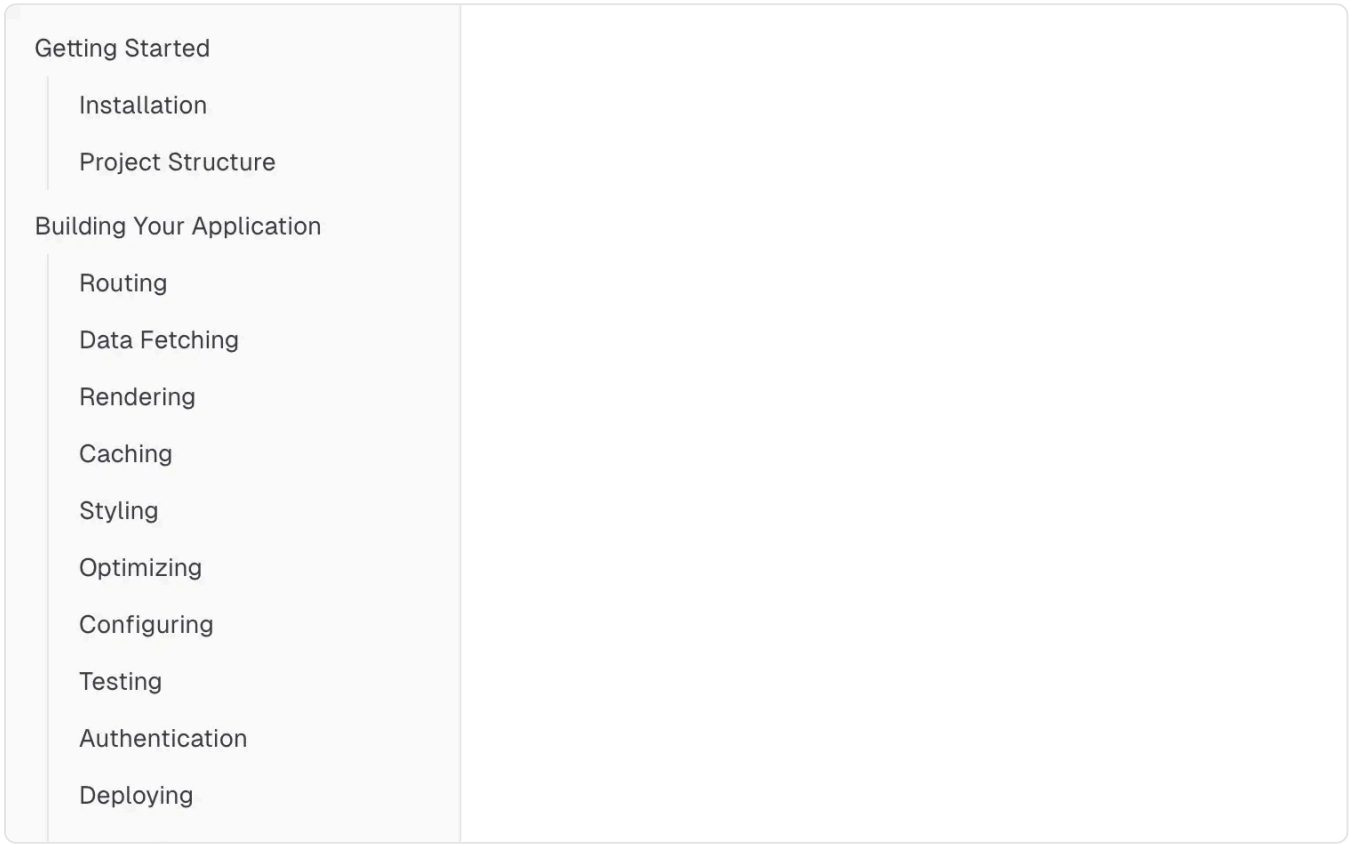
```
<SidebarMenuItem>
  <SidebarMenuButton asChild>
    <a href="#">
      <Home />
      <span>Home</span>
    </a>
  </SidebarMenuButton>
  <DropdownMenu>
    <DropdownMenuTrigger asChild>
      <SidebarMenuAction>
        <MoreHorizontal />
      </SidebarMenuAction>
    </DropdownMenuTrigger>
    <DropdownMenuContent side="right" align="start">
      <DropdownMenuItem>
```

```
          <span>Edit Project</span>
        </DropdownMenuItem>
        <DropdownMenuItem>
          <span>Delete Project</span>
        </DropdownMenuItem>
      </DropdownMenuContent>
    </DropdownMenu>
  </SidebarMenuItem>
```

SidebarMenuSub

The `SidebarMenuSub` component is used to render a submenu within a `SidebarMenu`.

Use `<SidebarMenuSubItem />` and `<SidebarMenuSubButton />` to render a submenu item.



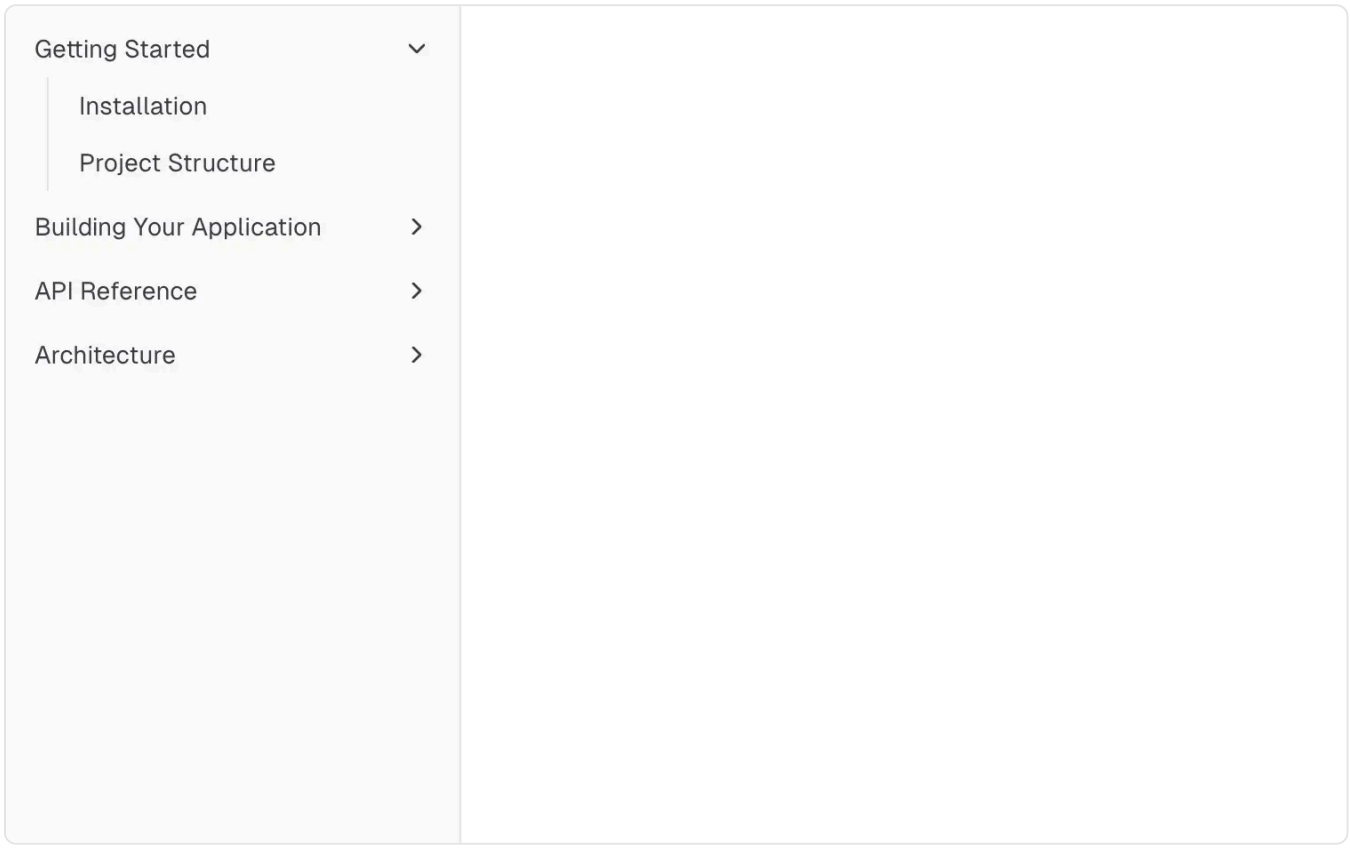
A sidebar menu with a submenu.

```
<SidebarMenuItem>
  <SidebarMenuButton />
  <SidebarMenuSub>
    <SidebarMenuSubItem>
```

```
        <SidebarMenuSubButton />
      </SidebarMenuSubItem>
      <SidebarMenuSubItem>
        <SidebarMenuSubButton />
      </SidebarMenuSubItem>
    </SidebarMenuSub>
  </SidebarMenuItem>
```

Collapsible SidebarMenu

To make a `SidebarMenu` component collapsible, wrap it and the `SidebarMenuSub` components in a `Collapsible`.



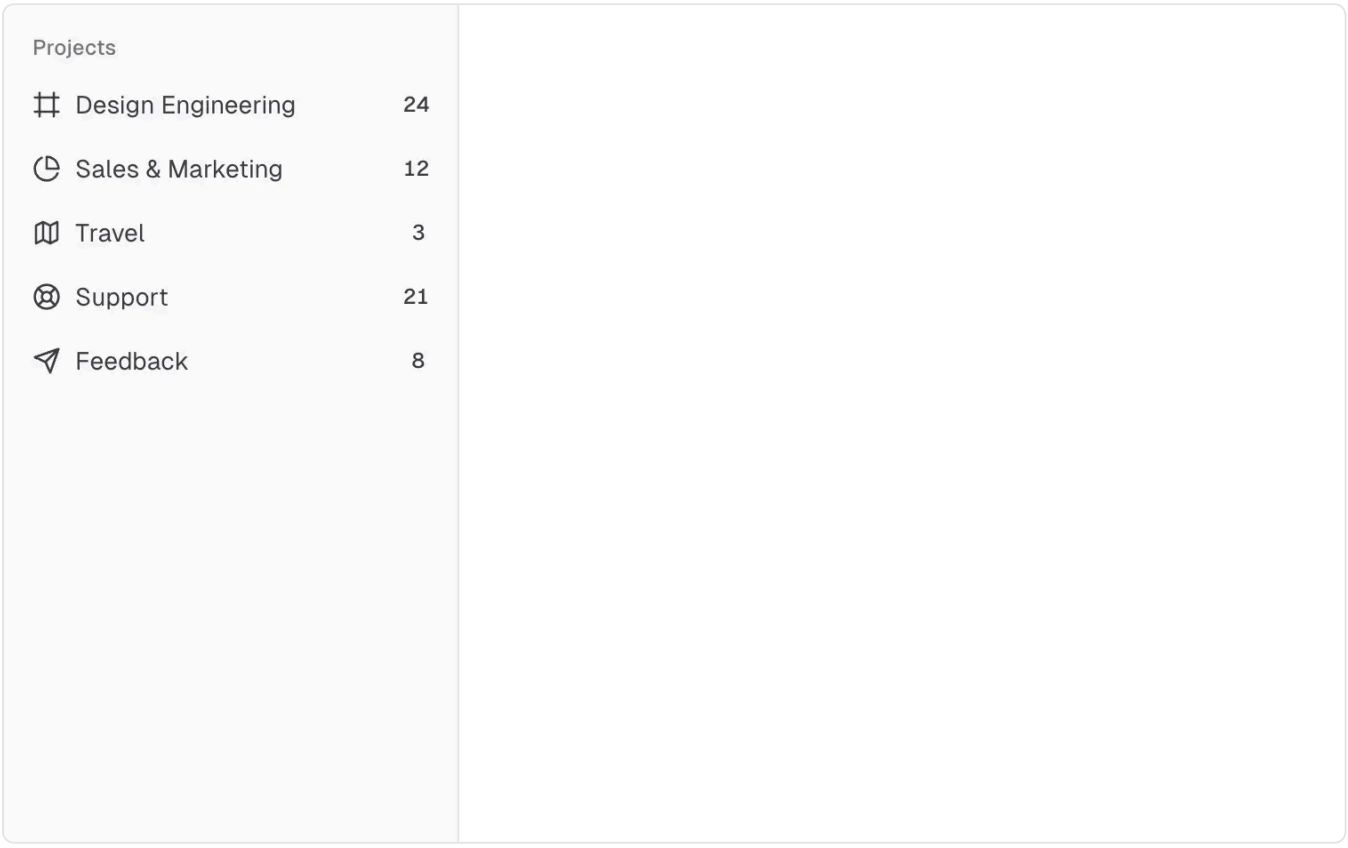
A collapsible sidebar menu.

```
<SidebarMenu>
  <Collapsible defaultOpen className="group/collapsible">
    <SidebarMenuItem>
      <CollapsibleTrigger asChild>
        <SidebarMenuButton />
      </CollapsibleTrigger>
```

```
        <CollapsibleContent>
          <SidebarMenuSub>
            <SidebarMenuSubItem />
          </SidebarMenuSub>
        </CollapsibleContent>
      </SidebarMenuItem>
    </Collapsible>
  </SidebarMenu>
```

SidebarMenuBadge

The `SidebarMenuBadge` component is used to render a badge within a `SidebarMenuItem`.



A sidebar menu with a badge.

```
<SidebarMenuItem>
  <SidebarMenuButton />
  <SidebarMenuBadge>24</SidebarMenuBadge>
</SidebarMenuItem>
```

SidebarMenuSkeleton

The `SidebarMenuSkeleton` component is used to render a skeleton for a `SidebarMenu`. You can use this to show a loading state when using React Server Components, SWR or react-query.

```tsx
function NavProjectsSkeleton() {
  return (
    <SidebarMenu>
      {Array.from({ length: 5 }).map((_, index) => (
        <SidebarMenuItem key={index}>
          <SidebarMenuSkeleton />
        </SidebarMenuItem>
      ))}
    </SidebarMenu>
  )
}
```

SidebarSeparator

The `SidebarSeparator` component is used to render a separator within a `Sidebar`.

```tsx
<Sidebar>
  <SidebarHeader />
  <SidebarSeparator />
  <SidebarContent>
    <SidebarGroup />
    <SidebarSeparator />
    <SidebarGroup />
  </SidebarContent>
</Sidebar>
```

SidebarTrigger

Use the `SidebarTrigger` component to render a button that toggles the sidebar.

The `SidebarTrigger` component must be used within a `SidebarProvider`.

```
<SidebarProvider>
  <Sidebar />
  <main>
    <SidebarTrigger />
  </main>
</SidebarProvider>
```

Custom Trigger

To create a custom trigger, you can use the `useSidebar` hook.

```
import { useSidebar } from "@/components/ui/sidebar"

export function CustomTrigger() {
  const { toggleSidebar } = useSidebar()

  return <button onClick={toggleSidebar}>Toggle Sidebar</button>
}
```

SidebarRail

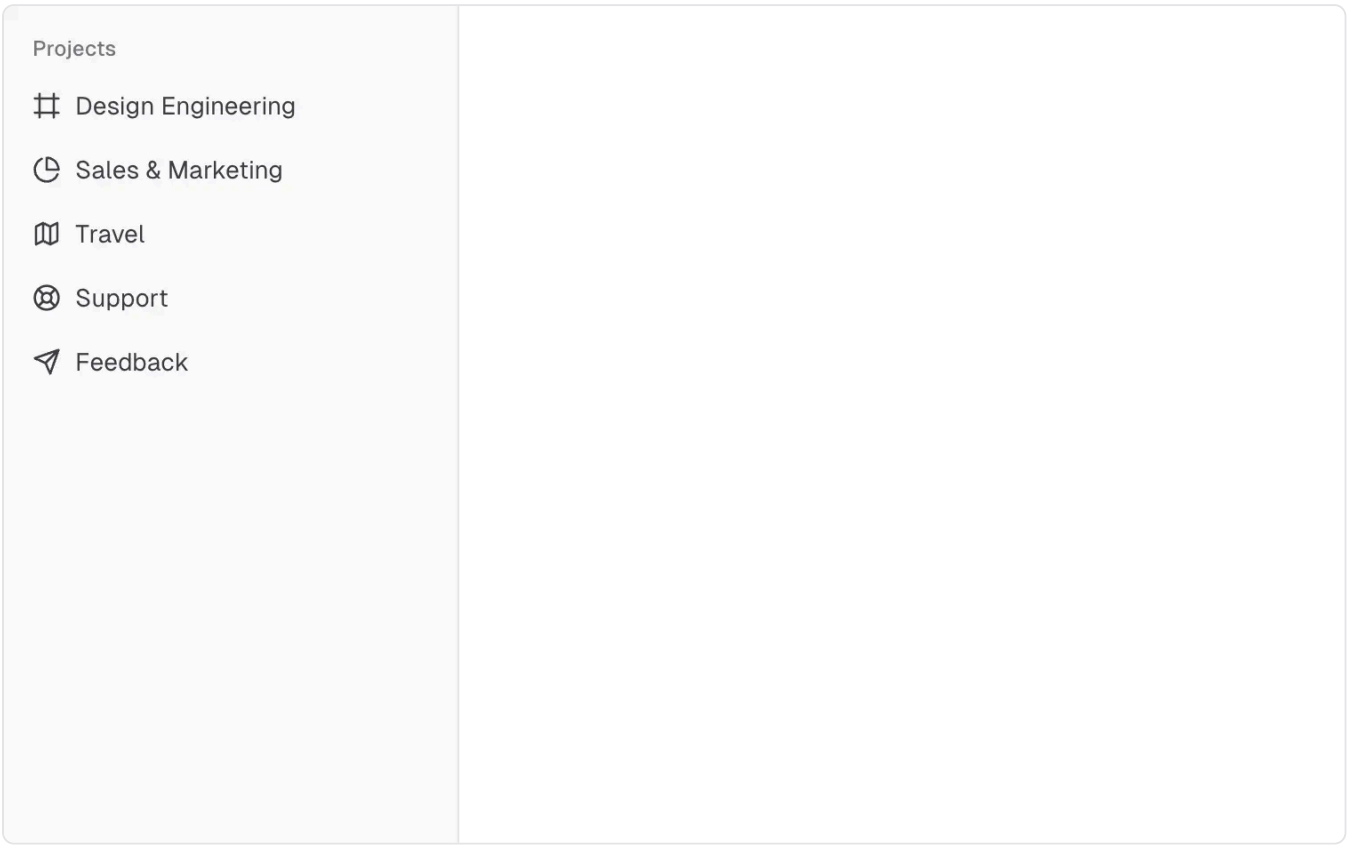
The `SidebarRail` component is used to render a rail within a `Sidebar`. This rail can be used to toggle the sidebar.

```
<Sidebar>
  <SidebarHeader />
  <SidebarContent>
    <SidebarGroup />
  </SidebarContent>
  <SidebarFooter />
  <SidebarRail />
</Sidebar>
```

Data Fetching

React Server Components

Here's an example of a `SidebarMenu` component rendering a list of projects using React Server Components.



A sidebar menu using React Server Components.

Skeleton to show loading state.

```tsx
function NavProjectsSkeleton() {
  return (
    <SidebarMenu>
      {Array.from({ length: 5 }).map((_, index) => (
        <SidebarMenuItem key={index}>
          <SidebarMenuSkeleton showIcon />
        </SidebarMenuItem>
      ))}
    </SidebarMenu>
  )
}
```

Server component fetching data.

```tsx
async function NavProjects() {
  const projects = await fetchProjects()

  return (
    <SidebarMenu>
      {projects.map((project) => (
        <SidebarMenuItem key={project.name}>
          <SidebarMenuButton asChild>
            <a href={project.url}>
              <project.icon />
              <span>{project.name}</span>
            </a>
          </SidebarMenuButton>
        </SidebarMenuItem>
      ))}
    </SidebarMenu>
  )
}
```

Usage with React Suspense.

```tsx
function AppSidebar() {
  return (
    <Sidebar>
      <SidebarContent>
        <SidebarGroup>
          <SidebarGroupLabel>Projects</SidebarGroupLabel>
          <SidebarGroupContent>
            <React.Suspense fallback={<NavProjectsSkeleton />}>
              <NavProjects />
            </React.Suspense>
          </SidebarGroupContent>
        </SidebarGroup>
      </SidebarContent>
    </Sidebar>
  )
}
```

SWR and React Query

Search... ⌘ K

SWR

```tsx
function NavProjects() {
  const { data, isLoading } = useSWR("/api/projects", fetcher)

  if (isLoading) {
    return (
      <SidebarMenu>
        {Array.from({ length: 5 }).map((_, index) => (
          <SidebarMenuItem key={index}>
            <SidebarMenuSkeleton showIcon />
          </SidebarMenuItem>
        ))}
      </SidebarMenu>
    )
  }

  if (!data) {
    return ...
  }

  return (
    <SidebarMenu>
      {data.map((project) => (
        <SidebarMenuItem key={project.name}>
          <SidebarMenuButton asChild>
            <a href={project.url}>
              <project.icon />
```

React Query

```tsx
function NavProjects() {
  const { data, isLoading } = useQuery()

  if (isLoading) {
    return (
      <SidebarMenu>
        {Array.from({ length: 5 }).map((_, index) => (
```

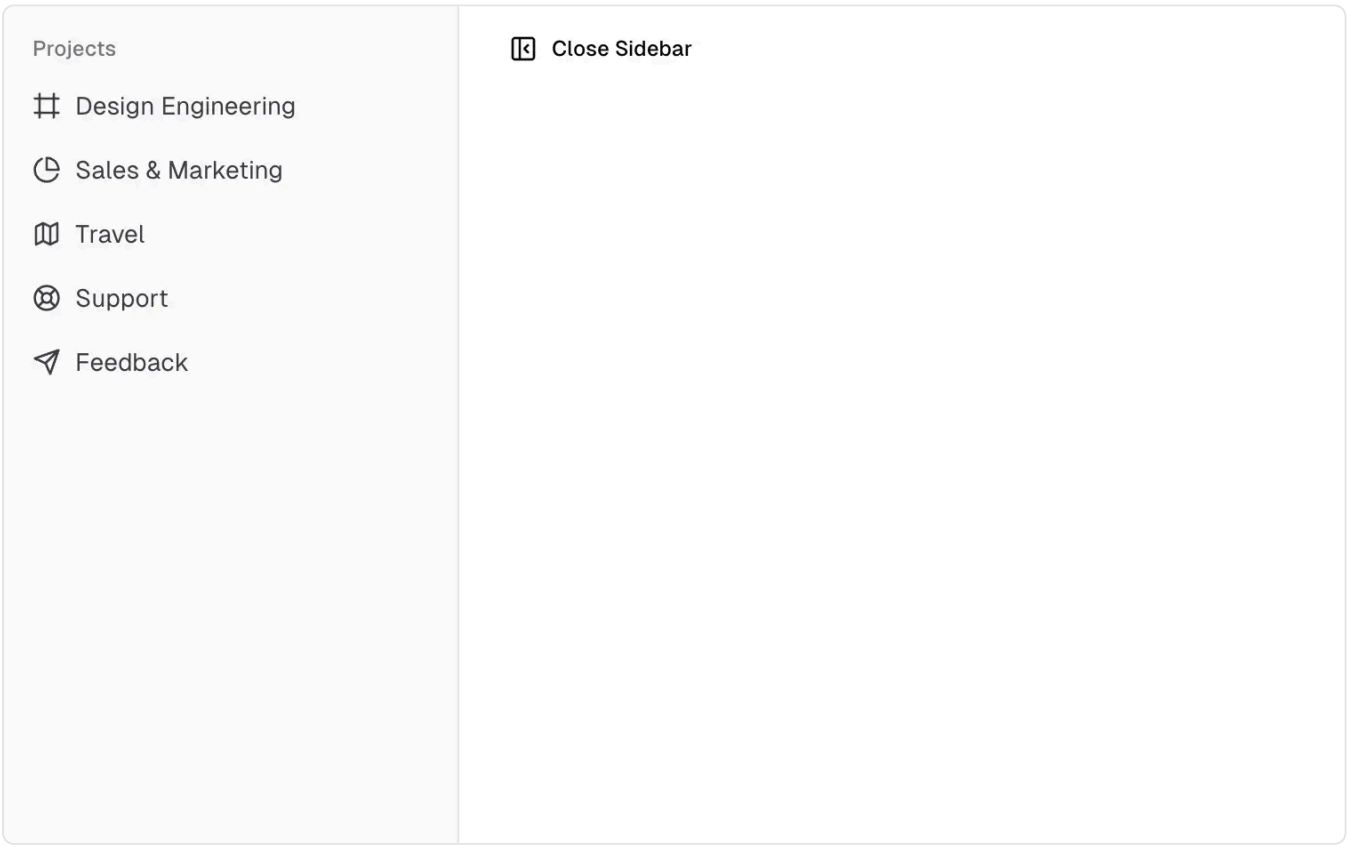
```
            <SidebarMenuItem key={index}>
              <SidebarMenuSkeleton showIcon />
            </SidebarMenuItem>
          ))}
        </SidebarMenu>
      )
    }

    if (!data) {
      return ...
    }

    return (
      <SidebarMenu>
        {data.map((project) => (
          <SidebarMenuItem key={project.name}>
            <SidebarMenuButton asChild>
              <a href={project.url}>
                <project.icon />
```

Controlled Sidebar

Use the `open` and `onOpenChange` props to control the sidebar.



A controlled sidebar.

```tsx
export function AppSidebar() {
  const [open, setOpen] = React.useState(false)

  return (
    <SidebarProvider open={open} onOpenChange={setOpen}>
      <Sidebar />
    </SidebarProvider>
  )
}
```

Theming

We use the following CSS variables to theme the sidebar.

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

  .dark {
    --sidebar-background: 240 5.9% 10%;
    --sidebar-foreground: 240 4.8% 95.9%;
    --sidebar-primary: 0 0% 98%;
    --sidebar-primary-foreground: 240 5.9% 10%;
    --sidebar-accent: 240 3.7% 15.9%;
    --sidebar-accent-foreground: 240 4.8% 95.9%;
    --sidebar-border: 240 3.7% 15.9%;
    --sidebar-ring: 217.2 91.2% 59.8%;
  }
}
```

We intentionally use different variables for the sidebar and the rest of the application to make it easy to have a sidebar that is styled differently from the rest of the application. Think a sidebar with a darker shade from the main application.

Styling

Here are some tips for styling the sidebar based on different states.

- **Styling an element based on the sidebar collapsible state.** The following will hide the `SidebarGroup` when the sidebar is in `icon` mode.

```
<Sidebar collapsible="icon">
  <SidebarContent>
    <SidebarGroup className="group-data-[collapsible=icon]:hidden" />
  </SidebarContent>
```

```
</Sidebar>
```

- **Styling a menu action based on the menu button active state.** The following will force the menu action to be visible when the menu button is active.

```
<SidebarMenuItem>
  <SidebarMenuButton />
  <SidebarMenuAction className="peer-data-[active=true]/menu-button:opacity-10
</SidebarMenuItem>
```

You can find more tips on using states for styling in this Twitter thread.

Changelog

2024-10-30 Cookie handling in setOpen

- #5593 - Improved setOpen callback logic in `<SidebarProvider>`.

Update the `setOpen` callback in `<SidebarProvider>` as follows:

```
const setOpen = React.useCallback(
  (value: boolean | ((value: boolean) => boolean)) => {
    const openState = typeof value === "function" ? value(open) : value
    if (setOpenProp) {
      setOpenProp(openState)
    } else {
      _setOpen(openState)
    }

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

2024-10-21 Fixed `text-sidebar-foreground`

- #5491 - Moved `text-sidebar-foreground` from `<SidebarProvider>` to `<Sidebar>` component.

2024-10-20 Typo in `useSidebar` hook.

Fixed typo in `useSidebar` hook.

sidebar.tsx

```
- throw new Error("useSidebar must be used within a Sidebar.")
+ throw new Error("useSidebar must be used within a SidebarProvider.")
```

‹ Sheet

Skeleton ›