Let's add Mantine to the Zustand project!

Why use Mantine

Mantine is a modern React UI system that gives you:

- **Prebuilt components** (AppShell, Paper, Modal, Drawer, Stepper, DataTable*) that look good out-of-the-box and are fully accessible.
- **One theme to rule them all**: colors, fonts, radii, spacing, shadows—controlled centrally (dark mode included).
- **Great dev ergonomics**: simple props for spacing/typography; flexible composition; solid TypeScript support.
- **Fewer CSS worries**: you focus on *structure and logic* while Mantine styles and layouts stay consistent across pages.
- **Scales with you**: easy to add forms, notifications, modals, drawers, tabs, grids, etc., without re-inventing UI.

* Data table is via community libs, but Mantine plays nicely with them.

Setup

`npm i @mantine/core @mantine/hooks @tabler/icons-react`
Then import Mantine styles once

`import '@mantine/core/styles.css';`
Tip: Wrap your app **once** with `<MantineProvider>`. If you also wrap in `main.tsx`, remove it from `App.tsx` (or vice-versa).

App shell & theming (your `App.tsx`)

We're using Mantine v7's `AppShell` with nested:

```
<AppShell.Header> ... </AppShell.Header>
<AppShell.Main> ... </AppShell.Main>
```

We define 3 themes (`tealTheme`, `pinkTheme`, `corporateBlueTheme`). To switch:

```
<MantineProvider theme={pinkTheme}
defaultColorScheme="auto">
```

- `defaultColorScheme="auto"` respects the OS dark/light setting.

- The `ThemeToggle` you added flips light/dark at runtime.

Using Mantine on your pages

`Home.tsx`

```
import { Title, Container } from '@mantine/core';

export default function Home() {
  return (
    <Container size="sm" mt="xl">
      <Title order={1}>Home Page</Title>
    </Container>
  );
}
```

- `Container` centers content with max width; `mt="xl"` adds vertical spacing.
- `Title order={1}` renders an accessible H1 with theme typography.

`About.tsx`

```
import { Title, Container, Paper } from '@mantine/core';

export default function About() {
  return (
    <Container size="sm" mt="xl">
      <Paper shadow="md" p="xl" radius="md" withBorder>
        <Title order={1}>About Us</Title>
      </Paper>
    </Container>
  );
}
```

- `Paper` gives a card-like surface with theme shadow, padding, rounded corners, and a border.
- Great for any content block or card layout.

`Dashboard.tsx`

```
import { useLoaderData } from 'react-router-dom';
import { Title, Text, Container, Paper } from '@mantine/
core';

export default function Dashboard() {
  const data = useLoaderData() as { message: string };

  return (
    <Container size="sm" mt="xl">
      <Paper shadow="md" p="xl" radius="md" withBorder>
        <Title order={1} color="green">Dashboard</Title>
        <Text mt="md">{data.message}</Text>
      </Paper>
    </Container>
  );
}
```

- Pulls data from your route loader and displays it in a styled card.
- You can swap `color="green"` for your theme's `primaryColor` by using `c="primary"` (Mantine v7 color prop is `c`).

Login.tsx

```
import { Form, useActionData } from 'react-router-dom';
import {
  TextInput,
  PasswordInput,
  Button,
  Paper,
  Title,
  Container,
  Text,
  Stack,
} from '@mantine/core';

export default function Login() {
  const error = useActionData() as string | undefined;

  return (
    <Container size="xs" mt="xl">
```

```
      <Paper shadow="md" p="xl" radius="md" withBorder>
        <Title order={2} mb="lg">Login</Title>
        <Form method="post">
          <Stack>
            <TextInput name="username" label="Username"
required />
            <PasswordInput name="password" label="Password"
required />
            {error && (
              <Text c="red" fw={600}>
                {error}
              </Text>
            )}
            <Button type="submit" fullWidth>
              Log In
            </Button>
          </Stack>
        </Form>
      </Paper>
    </Container>
  );
}
```

- `Stack` handles vertical spacing; `TextInput/PasswordInput` are fully themed and accessible.
- `Text c="red"` shows error state if your route `action` returns one.

`App.tsx` (header, nav, theme toggle)

- `AppShell` gives you a responsive layout frame.
- Header has your nav (`<Link>`) and auth button area.
- `ThemeToggle` uses `useMantineColorScheme()` to flip dark/light.
- Only one `<MantineProvider>` in your whole app. If you later move it to `main.tsx`, just remove it from `App.tsx`.

Common pitfalls

- **Mantine v7 Header import**: Correct approach is `<AppShell.Header>` (not `import { Header }`).
- **Duplicate providers**: Keep just one `MantineProvider`.
- **Forgetting styles**: Be sure to include `@mantine/core/styles.css`