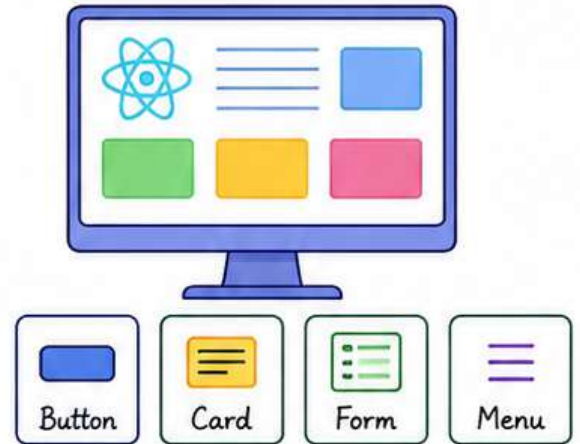


Chapter 8: React UI Libraries

8.1 Introduction to UI Libraries

8.1.1 What are UI Libraries?

- * UI Libraries are **pre-built** React component libraries that help in building **beautiful, responsive** and **consistent** user interfaces quickly.
- * They provide ready-to-use components like buttons, forms, cards, modals, navigation menus, etc.
- * These libraries follow **best practices** and design principles which **saves time** and improves productivity.



Key Points :

- ✓ UI Libraries provide reusable and customizable components.
- ✓ They ensure consistency in design and user experience.
- ✓ They are responsive and work across different devices.
- ✓ They allow developers to focus more on application logic.
- ✓ They reduce development time and effort.

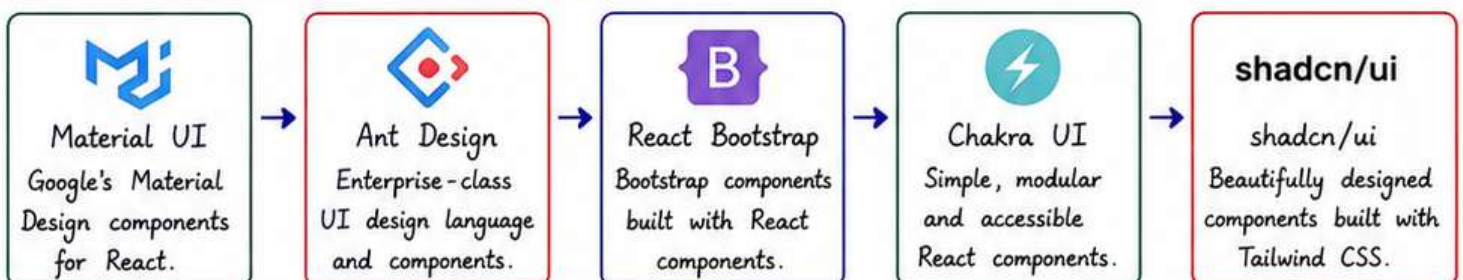


8.1.2 Why Use UI Libraries?

- * **Saves Time** - No need to build components from scratch.
- * **Consistent Design** - Pre-designed components follow the same style.
- * **Responsive** - Components are mobile-friendly.
- * **Customizable** - Easy to customize as per project needs.
- * **Accessibility** - Many libraries follow accessibility best practices.
- * **Community Support** - Active communities and regular updates.



8.1.3 Popular React UI Libraries



Key Takeaway :



UI Libraries in React help us build **modern, responsive** and **consistent** user interfaces **faster** and with **less effort**.



8.2 Material UI

8.2.1 What is Material UI?

- * Material UI (MUI) is a popular React UI library that implements **Google's Material Design**.
- * It provides a large collection of **pre-built, customizable** components.
- * It helps developers build **responsive** and **beautiful** applications faster.
- * MUI is **open-source** and widely used in the industry.



MATERIAL UI



Key Points :

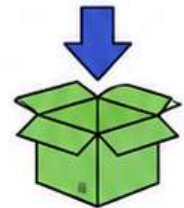
- ✓ Follows Google's Material Design guidelines.
- ✓ Offers a rich set of components.
- ✓ Highly customizable and themeable.
- ✓ Responsive design out of the box.
- ✓ Well documented and has a large community.
- ✓ Supports both JavaScript and TypeScript.



8.2.2 How to Install Material UI?

- * We can install Material UI using npm or yarn.

Using npm	Using yarn
<code>npm install @mui/material @emotion/react @emotion/styled</code>	<code>yarn add @mui/material @emotion/react @emotion/styled</code>



8.2.3 Basic Example

```
import Button from '@mui/material/Button';
function App() {
  return (
    <div style={{ padding: '20px' }}>
      <Button variant="contained" color="primary">
        Click Me
      </Button>
    </div>
  );
}
export default App;
```

Output :



Key Takeaway :



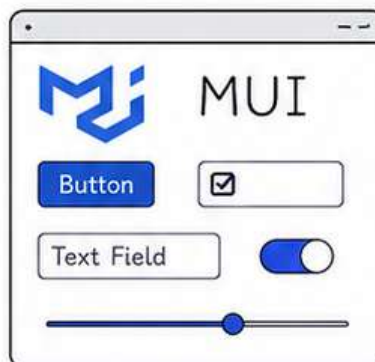
Material UI helps us build **modern, responsive** and **consistent** user interfaces with **less code** and **more efficiency**.



8.3 MUI Components

8.3.1 What are MUI Components?

- * MUI provides a wide range of **pre-built** components to build **modern** and **responsive** user interfaces.
- * These components are easy to use, **customizable** and well documented.
- * They follow **Material Design** principles and ensure consistency across applications.



8.3.2 Commonly Used MUI Components

Category	Component	Description	Example									
Layout	Container	Provides a fixed width container.										
	Grid	Creates responsive grid layouts.										
Inputs	TextField	Used to get text input from the user.	Enter text									
	Checkbox	Allows the user to select one or more options.	☒ Option 1 ☐ Option 2									
	Radio	Allows the user to select only one option.	☒ Option 1 ☐ Option 2									
Feedback	Alert	Displays important messages to the user.										
Navigation	AppBar	Used to create application top navigation bar.										
	Drawer	Used to create side navigation menus.										
Data	Table	Used to display data in tabular format.	<table><thead><tr><th>ID</th><th>Name</th><th>Email</th></tr></thead><tbody><tr><td>1</td><td>Alice</td><td>alice@mail.com</td></tr><tr><td>2</td><td>Bob</td><td>bob@mail.com</td></tr></tbody></table>	ID	Name	Email	1	Alice	alice@mail.com	2	Bob	bob@mail.com
ID	Name	Email										
1	Alice	alice@mail.com										
2	Bob	bob@mail.com										
Display	Card	Used to group related content in a container.										



Key Points :

- ✓ MUI components are pre-built and ready to use.
- ✓ They follow Material Design guidelines.
- ✓ They are responsive, accessible and customizable.
- ✓ They help in building consistent and professional UIs quickly.
- ✓ MUI has a large collection of components for any type of application.



Key Takeaway :



MUI components make UI development **faster**, **easier** and more **consistent** by providing a rich set of **ready-to-use** elements.



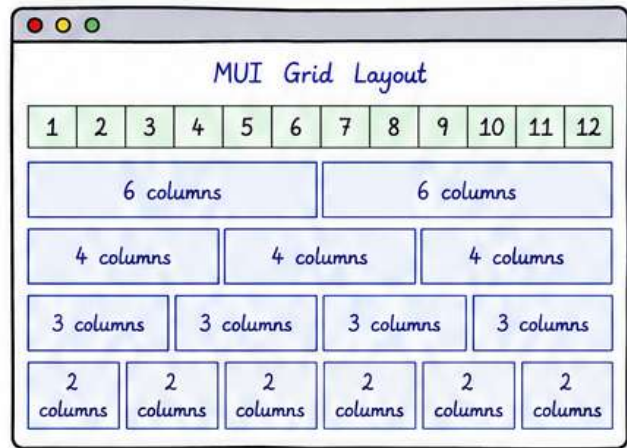
8.4 MUI Grid

8.4.1 What is MUI Grid?

- * MUI Grid is a layout system that uses **flexbox** to create **responsive layouts**.
- * It divides the page into **12 columns**.
- * It helps in aligning items in rows and columns easily.
- * Grid is fully **responsive** and adjusts based on screen size.

Key Points :

- ✓ Based on 12-column layout system.
- ✓ Create responsive and flexible layouts.
- ✓ Works on all screen sizes.
- ✓ Easy to align and space components.
- ✓ Improves UI design and structure.



8.4.2 Grid Breakpoints

- * MUI Grid supports breakpoints for responsiveness.

Breakpoint	Screen Size
xs (extra small)	0px and up
sm (small)	600px and up
md (medium)	900px and up
lg (large)	1200px and up
xl (extra large)	1536px and up

Example (Responsive Behavior)

xs = 12 columns (full width)
sm = 6 columns (half width)
md = 4 columns
lg = 3 columns
xl = 2 columns

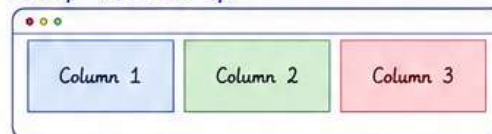


8.4.3 Basic Example

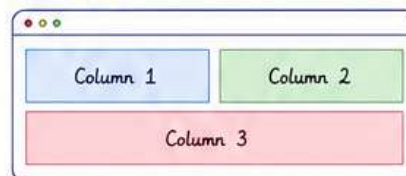
```
import Grid from '@mui/material/Grid';
function App() {
  return (
    <Grid container spacing={2}>
      <Grid item xs={12} sm={6} md={4}>
        <div style={{background:'#bbdefb', padding:'20px'}}>
          Column 1
        </div>
      </Grid>
      <Grid item xs={12} sm={6} md={4}>
        <div style={{background:'#c8e6c9', padding:'20px'}}>
          Column 2
        </div>
      </Grid>
      <Grid item xs={12} sm={6} md={4}>
        <div style={{background:'#ffcdd2', padding:'20px'}}>
          Column 3
        </div>
      </Grid>
    </Grid>
  );
}
export default App;
```

Output :

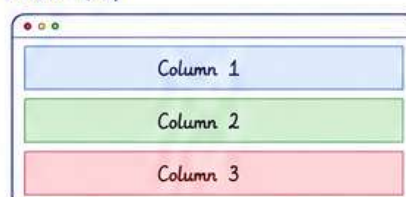
Desktop (md and up)



Tablet (sm)



Mobile (xs)



Key Takeaway :



MUI Grid makes it easy to create **responsive** and **flexible layouts** using the **12-column system** and breakpoints.



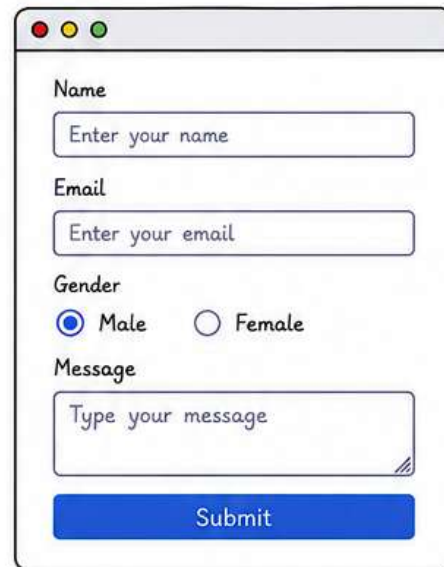
8.5 MUI Forms

8.5.1 What is MUI Forms?

- * MUI provides ready-to-use form components to collect user input.
- * These components are easy to use, customizable and follow **Material Design** guidelines.
- * MUI forms help in building **beautiful, consistent** and **accessible** user interfaces.

Key Points :

- ✓ Provides pre-built input components.
- ✓ Easy to handle form validation.
- ✓ Supports both controlled and uncontrolled forms.
- ✓ Customizable and follows Material Design.
- ✓ Improves user experience and consistency.



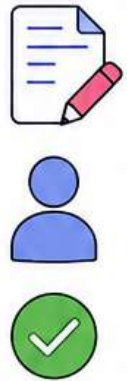
Name
Enter your name

Email
Enter your email

Gender
☒ Male ☐ Female

Message
Type your message

Submit



8.5.2 Commonly Used MUI Form Components

Component	Description	Example
TextField	Allows the user to enter text input.	Enter text
Checkbox	Allows the user to select one or more options.	☒ Accept Terms & Conditions
Radio	Allows the user to select only one option.	☒ Option 1 ☐ Option 2
Select	Provides a dropdown list of options.	Select option
FormControl	Helps in managing form state and validation.	
FormHelperText	Displays helper or error text for inputs.	This field is required.
Button	Used to submit or reset the form.	Submit Reset

Why MUI Forms?

- * Simple to implement
- * Built-in validation support
- * Consistent and responsive
- * Accessible form components



8.5.3 Basic Example

```
import TextField from '@mui/material/TextField';
import Button from '@mui/material/Button';
import { Box } from '@mui/material';
function App() {
  return (
    <Box component="form" sx={{ display: 'flex', flexDirection: 'column',
      gap: 2, maxWidth: 400, mx: 'auto', mt: 2 }}>
      <TextField label="Name" variant="outlined" />
      <TextField label="Email" type="email" variant="outlined" />
      <TextField label="Message" multiline rows={4} variant="outlined" />
      <Button variant="contained" color="primary" type="submit">
        Submit
      </Button>
    </Box>
  );
}
export default App;
```

Output :



Name

Email

Message

Submit

Key Takeaway :



MUI Forms provide **powerful, easy-to-use** components that help in collecting and validating user input with **less code** and **more efficiency**.



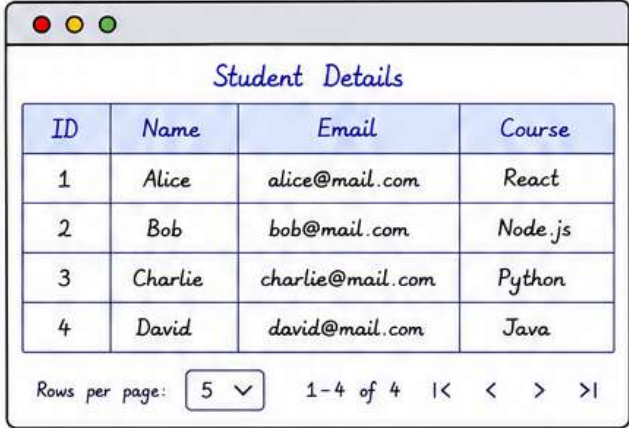
8.6 MUI Tables

8.6.1 What is MUI Tables?

- * MUI Tables are used to display data in **rows and columns**.
- * They provide a **clean and structured** way to present information.
- * MUI Tables follow **Material Design** guidelines and are **responsive**.
- * They support **sorting, pagination** and **customization**.

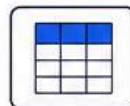
Key Points :

- ✓ Display data in a structured format.
- ✓ Easy to implement and customize.
- ✓ Supports sorting and pagination.
- ✓ Responsive and mobile-friendly.
- ✓ Follows Material Design principles.

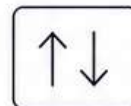


ID	Name	Email	Course
1	Alice	alice@mail.com	React
2	Bob	bob@mail.com	Node.js
3	Charlie	charlie@mail.com	Python
4	David	david@mail.com	Java

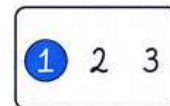
Rows per page: 5 1-4 of 4 |< < > >|



Structured Data



Sorting



Pagination



Responsive

8.6.2 Commonly Used MUI Table Components

Component	Description	Example
Table	The root component used to create a table.	<code><Table> ... </Table></code>
TableContainer	Provides a container with scrolling support.	<code><TableContainer> ... </TableContainer></code>
TableHead	Used to define the header section of the table.	<code><TableHead> ... </TableHead></code>
TableBody	Used to define the body section of the table.	<code><TableBody> ... </TableBody></code>
TableRow	Represents a row in the table.	<code><TableRow> ... </TableRow></code>
TableCell	Represents a cell in a row.	<code><TableCell> ... </TableCell></code>
TablePagination	Used to add pagination controls to the table.	<code><TablePagination /></code>

Why MUI Tables?

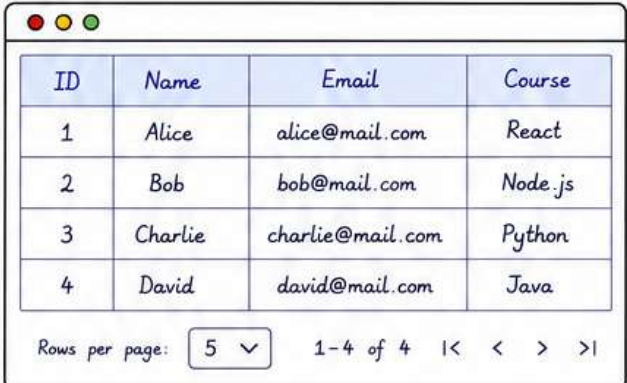
- * Clear and structured data presentation
- * Easy sorting and pagination
- * Improved readability and user experience
- * Responsive across all devices

8.6.3 Basic Example

```
import { Table, TableContainer, TableHead, TableRow,
  TableCell, TableBody, Paper } from '@mui/material';

function App() {
  const rows = [
    { id: 1, name: 'Alice', email: 'alice@mail.com', course: 'React' },
    { id: 2, name: 'Bob', email: 'bob@mail.com', course: 'Node.js' },
    { id: 3, name: 'Charlie', email: 'charlie@mail.com', course: 'Python' },
    { id: 4, name: 'David', email: 'david@mail.com', course: 'Java' }
  ];
  return (
    <TableContainer component={Paper} style={{ maxWidth: 600 }}>
      <Table>
        <TableHead>
          <TableRow>
            <TableCell>ID</TableCell>
            <TableCell>Name</TableCell>
            <TableCell>Email</TableCell>
            <TableCell>Course</TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {rows.map(row => (
            <TableRow key={row.id}>
              <TableCell>{row.id}</TableCell>
              <TableCell>{row.name}</TableCell>
              <TableCell>{row.email}</TableCell>
              <TableCell>{row.course}</TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </TableContainer>
  );
}
export default App;
```

Output :



ID	Name	Email	Course
1	Alice	alice@mail.com	React
2	Bob	bob@mail.com	Node.js
3	Charlie	charlie@mail.com	Python
4	David	david@mail.com	Java

Rows per page: 5 1-4 of 4 |< < > >|

Key Takeaway :



MUI Tables provide a **simple, efficient and responsive** way to display structured data with **sorting and pagination** support.



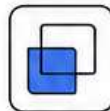
8.7 MUI Dialogs

8.7.1 What is MUI Dialogs?

- * MUI Dialogs are **popup windows** that appear on top of the main content.
- * They are used to display **important messages, forms or confirmation boxes**.
- * Dialogs help users **focus** on a specific task without leaving the page.
- * They are **customizable** and follow **Material Design** guidelines.

Key Points :

- ✓ Used to display information, alerts or forms.
- ✓ Appears above the current page content.
- ✓ Can be customized in size, style and actions.
- ✓ Supports animations and transitions.
- ✓ Improves user interaction and experience.



Modal
Popup



Alerts &
Notifications



Forms in
Dialogs



Customizable
& Responsive



8.7.2 Commonly Used MUI Dialog Components

Component	Description	Example Use	Purpose
Dialog	The root component for a dialog.	<code><Dialog> ... </Dialog></code>	Wraps the entire dialog content.
DialogTitle	Defines the title of the dialog.	<code><DialogTitle>Title</DialogTitle></code>	Displays the dialog header.
DialogContent	Holds the main content of the dialog.	<code><DialogContent>...</DialogContent></code>	Displays text, forms or other components.
DialogActions	Holds the action buttons at the bottom.	<code><DialogActions>...</DialogActions></code>	Used for buttons like OK, Cancel, etc.
Button	Used inside actions for user response.	<code><Button>OK</Button></code>	Performs actions like Submit or Cancel.
IconButton	Used to add close or icon actions.	<code><IconButton> ... </IconButton></code>	Used for close or custom icon actions.
DialogContentText	Used to display text inside content section.	<code><DialogContentText>Text</...></code> <code>DialogContentText</code>	Displays messages or additional text.

8.7.3 Basic Example

Output :

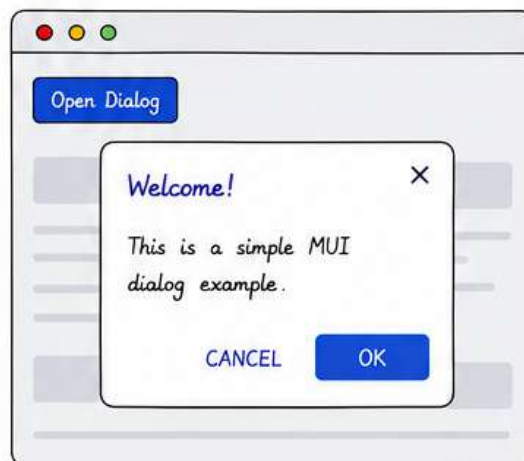
```
import { useState } from 'react';
import { Button, Dialog, DialogTitle,
  DialogContent, DialogActions } from '@mui/material';

function App() {
  const [open, setOpen] = useState(false);

  return (
    <div>
      <Button variant="contained" onClick={() => setOpen(true)}>
        Open Dialog
      </Button>

      <Dialog open={open} onClose={() => setOpen(false)}>
        <DialogTitle>Welcome!</DialogTitle>
        <DialogContent>
          <DialogContentText>
            This is a simple MUI dialog example.
          </DialogContentText>
        </DialogContent>
        <DialogActions>
          <Button onClick={() => setOpen(false)}>Cancel</Button>
          <Button variant="contained" onClick={() => setOpen(false)}>
            OK
          </Button>
        </DialogActions>
      </Dialog>
    </div>
  );
}

export default App;
```



Key Takeaway :



MUI Dialogs help in displaying **important messages, forms and alerts** in a clean, focused and interactive way with **customizable actions**.



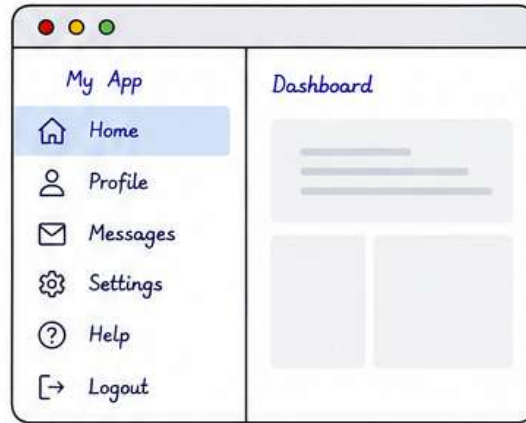
8.8 MUI Navigation

8.8.1 What is MUI Navigation?

- * MUI Navigation components help users **move** between different pages or sections of an application.
- * They provide **links and menus** to access various parts of the app easily.
- * MUI Navigation follows **Material Design** guidelines and is responsive.

Key Points :

- ✓ Helps users navigate easily across the app.
- ✓ Supports different types of navigation menus.
- ✓ Responsive and works on all devices.
- ✓ Customizable and follows Material Design.
- ✓ Improves user experience and accessibility.



Drawer Navigation



List Navigation



Bottom Navigation



Breadcrumbs Navigation



8.8.2 Commonly Used MUI Navigation Components

Component	Description	Example Use	Purpose
AppBar	Top app bar used for navigation and branding.	<code><AppBar> ... </AppBar></code>	Displays app title, menu, and actions.
Toolbar	Container inside AppBar to place content.	<code><Toolbar> ... </Toolbar></code>	Holds title, menus and buttons.
Drawer	Side navigation panel that slides in.	<code><Drawer> ... </Drawer></code>	Used for sidebar menus.
List	Container for list items in navigation.	<code><List> ... </List></code>	Groups navigation items.
ListItem	Single item in a list.	<code><ListItem.button> ... </ListItem></code>	Represents a navigation link.
BottomNavigation	Bottom navigation bar for mobile apps.	<code><BottomNavigation> ... </BottomNavigation></code>	Provides quick access to main sections.
Breadcrumbs	Displays the current page path.	<code><Breadcrumbs> ... </Breadcrumbs></code>	Shows hierarchy of pages.
Tabs	Horizontal navigation between sections.	<code><Tabs> ... </Tabs></code>	Used to switch between related content.

Why MUI Navigation?

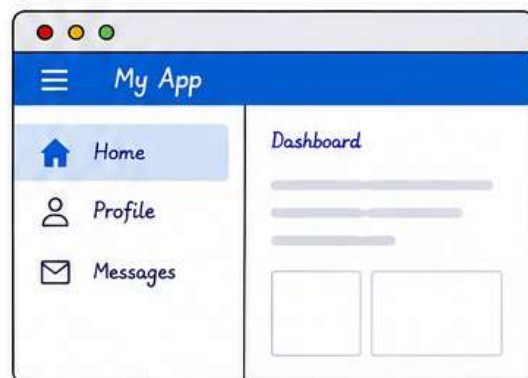
- * Makes navigation simple and user-friendly.
- * Provides multiple navigation styles.
- * Responsive across all devices.
- * Follows Material Design standards.
- * Easily customizable to fit the app.

8.8.3 Basic Example

```
import { useState } from 'react';
import { AppBar, Toolbar, Typography, IconButton, Drawer,
  List, ListItem, ListItemIcon, ListItemText } from '@mui/material';
import MenuIcon from '@mui/icons-material/Menu';
import HomeIcon from '@mui/icons-material/Home';
import PersonIcon from '@mui/icons-material/Person';
import MailIcon from '@mui/icons-material/Mail';

function App() {
  const [open, setOpen] = useState(false);
  return (
    <div>
      <AppBar position="static" sx={{ bgcolor: '#1976d2' }}>
        <Toolbar>
          <IconButton edge="start" color="inherit"
            onClick={() => setOpen(true)}>
            <MenuIcon />
          </IconButton>
          <Typography variant="h6" sx={{ flexGrow: 1 }}>My App</Typography>
        </Toolbar>
      </AppBar>
      <Drawer open={open} onClose={() => setOpen(false)}>
        <List>
          <ListItem button><ListItemIcon><HomeIcon /></ListItemIcon>
            <ListItemText primary="Home" /></ListItem>
          <ListItem button><ListItemIcon><PersonIcon /></ListItemIcon>
            <ListItemText primary="Profile" /></ListItem>
          <ListItem button><ListItemIcon><MailIcon /></ListItemIcon>
            <ListItemText primary="Messages" /></ListItem>
        </List>
      </Drawer>
    </div>
  );
}
export default App;
```

Output :



Navigation Types in MUI

- **Drawer Navigation** - Side menu for larger screens.
- **Bottom Navigation** - Bottom bar for mobile apps.
- **Breadcrumbs** - Shows the path of current page.
- **Tabs Navigation** - Switch between related sections.



Key Takeaway :



MUI Navigation components provide a **flexible, responsive and user-friendly** way to navigate through your application with **multiple navigation styles**.



8.9 React Icons

Popular Icon Packs in React Icons

8.9.1 What is React Icons?

- * React Icons is a popular library that provides a **huge collection of icons** from different icon packs.
- * It makes it easy to use **scalable vector icons** in React projects.
- * Icons are customizable in **size, color, and style** using props.

Key Points :

- ✓ Large collection of icons from multiple packs.
- ✓ Easy to install and use.
- ✓ Lightweight and tree-shakeable.
- ✓ Highly customizable (size, color, className, etc.).
- ✓ Improves UI with beautiful icons.

Fa (Font Awesome)	Md (Material Design)	Bs (Bootstrap)	Gi (Game Icons)	Ai (Ant Design)
				
Fi (Feather Icons)	Io (Ionicons)	Si (Simple Icons)	Ri (Remix Icons)	Cg (Clarity Icons)
				

Example of Icon Usage

					
Home	Mail	User	Settings	Like	Cart



8.9.2 Commonly Used React Icons

Icon Pack	Import Statement	Example Icon	Usage	Purpose
Font Awesome (Fa)	<code>import { FaHome } from 'react-icons/fa';</code>		<code><FaHome size={24} color="blue" /></code>	General purpose icons (home, user, search, etc.)
Material Design (Md)	<code>import { MdEmail } from 'react-icons/md';</code>		<code><MdEmail size={24} color="green" /></code>	Material Design style icons
Bootstrap (Bs)	<code>import { BsGear } from 'react-icons/bs';</code>		<code><BsGear size={24} color="purple" /></code>	Bootstrap styled icons
Feather (Fi)	<code>import { FiSearch } from 'react-icons/fi';</code>		<code><FiSearch size={24} color="gray" /></code>	Minimal and clean icons
Ionicons (Io)	<code>import { IoMdNotifications } from 'react-icons/io';</code>		<code><IoMdNotifications size={24} color="red" /></code>	Mobile / App oriented icons
Simple Icons (Si)	<code>import { SiGithub } from 'react-icons/si';</code>		<code><SiGithub size={24} color="black" /></code>	Brand / Logo icons
Remix Icons (Ri)	<code>import { RiBookmarkLine } from 'react-icons/ri';</code>		<code><RiBookmarkLine size={24} color="teal" /></code>	Versatile line icons

Why Use React Icons?

- * Easy to use and customize.
- * Consistent and scalable icons.
- * Supports many icon libraries.
- * Improves UI and UX.

8.9.3 Basic Example

```
import { FaHome, FaUser, FaCog, FaHeart } from 'react-icons/fa';
import { MdEmail } from 'react-icons/md';

function App() {
  return (
    <div style={{ display: 'flex', gap: '30px', alignItems: 'center' }}>
      <FaHome size={28} color="blue" title="Home" />
      <FaUser size={28} color="green" title="User" />
      <MdEmail size={28} color="orange" title="Email" />
      <FaCog size={28} color="purple" title="Settings" />
      <FaHeart size={28} color="red" title="Like" />
    </div>
  );
}
export default App;
```

Output :



Customization Examples

- Change size : `<FaHome size={40} />`
- Change color : `<FaHome color="teal" />`
- Add class name : `<FaHome className="my-icon" />`
- Inline style : `<FaHome style={{ color: 'red', fontSize: '30px' }} />`

Key Takeaway :



React Icons is a powerful library that provides a **wide range of scalable icons**. It is easy to use, customizable and helps to build **modern and attractive** user interfaces.



8.10 Component Customization

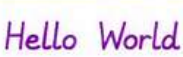
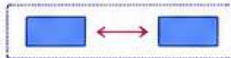
8.10.1 What is Component Customization?

- * MUI allows us to customize components to match our application's design and branding.
- * We can customize **styles, colors, typography, spacing and behavior**.
- * Customization helps to create **consistent and unique UI**.

Key Points :

- ✓ Change appearance to match our design.
- ✓ Customize colors, fonts, spacing, etc.
- ✓ Use `sx` prop or `styled` API.
- ✓ Create reusable and consistent UI.
- ✓ Enhances user experience and branding.

8.10.2 Commonly Customized Aspects

Aspect	What Can Be Customized?	Example
 Colors	Primary color, secondary color, background, text color, etc.	Change primary button color to match brand. 
 Typography	Font family, font size, font weight, text transform, etc.	Set custom font and size for headings. 
 Spacing	Margin, padding, gap, line-height, etc.	Add spacing between components. 
 Size	Width, height, min/max size, component size (small, medium, large).	Make buttons or inputs larger or smaller. 
 Borders & Shape	Border color, border radius, box shadow, outline, etc.	Round corners or add shadows. 
 States & Effects	Hover, focus, active, disabled styles, transitions, etc.	Change styles on hover or focus. 
 Variants	Change variants like contained, outlined, text, etc.	Use different variants for buttons.   

8.10.3 Basic Example (sx Prop)

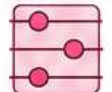
```
import { Button, Typography, Box } from '@mui/material';
function App() {
  return (
    <Box sx={{ p: 3, bgcolor: '#f5f5f5', borderRadius: 2 }}>
      <Typography sx={{ color: 'primary.main', fontSize: '24px',
        fontWeight: 'bold', mb: 2 }}>
        Welcome to MUI Customization
      </Typography>
      <Button
        variant="contained"
        sx={{ bgcolor: 'secondary.main', color: 'white',
          px: 4, py: 1.5, borderRadius: 2, fontSize: '16px' }}>
        Custom Button
      </Button>
    </Box>
  );
}
export default App;
```

Key Takeaway :



Component customization in MUI helps to build **beautiful, consistent and brand-aligned** user interfaces using `sx` prop, `styled` API, themes and component props.

Ways to Customize MUI Components

	1 sx Prop Use the <code>sx</code> prop to apply CSS styles directly to components.
	2 styled() API Use the <code>styled()</code> utility to create custom styled components.
	3 Theme Customization Customize the theme to apply global styles across the app.
	4 Global Styles Use <code>CssBaseline</code> and <code>GlobalStyles</code> for global style changes.
	5 Props Customization Use component props to modify variants, sizes, colors, etc.



Output :



Customization Example Using styled()

```
import { styled } from '@mui/material/styles';
import Button from '@mui/material/Button';

const MyButton = styled(Button)(({ theme }) => ({
  backgroundColor: theme.palette.success.main,
  color: 'white',
  fontWeight: 'bold',
  borderRadius: 8,
  padding: '10px 20px',
  '&:hover': { backgroundColor: theme.palette.success.dark }
}));
```

8.11 Theme Customization

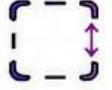
8.11.1 What is Theme Customization?

- * MUI comes with a **default theme**, but we can **customize** it to match our brand colors, typography, spacing and more.
- * Theme customization helps us maintain **consistent design** across the application.
- * It can be done using **createTheme()** and **ThemeProvider**.

Key Points :

- ✓ Create custom theme using **createTheme()**.
- ✓ Define **palette**, **typography**, **spacing**, **shape**, etc.
- ✓ Wrap the app with **ThemeProvider**.
- ✓ All MUI components automatically follow the custom theme.
- ✓ Supports light and dark mode.

Parts of a Theme

Palette  Define colors for primary, secondary, background, text, error, warning, etc.	Typography  Set font family, font size, font weight, line height, and more.	Spacing  Define spacing unit used in margins, paddings, and gaps.	Shape  Customize border radius for components.
Components  Override default styles of specific MUI components.	Shadows  Customize box-shadow values.	Breakpoints  Set custom breakpoints for responsive design.	Z-Index  Control stacking order of components.



8.11.2 Common Theme Customization Options

Option	Description	Example	Purpose
 Palette	Customize colors used in the application.	<code>palette: { primary: { main: '#1976d2' } }</code>	Change brand colors across the app.
Aa Typography	Define font styles for headings and text.	<code>typography: { h1: { fontSize: '2rem' } }</code>	Maintain consistent text styles.
 Spacing	Set the base spacing unit.	<code>spacing: 8</code>	Control spacing consistently.
 Shape	Set border radius for components.	<code>shape: { borderRadius: 8 }</code>	Customize corners and rounded edges.
 Components	Override default styles of specific components.	<code>components: { MuiButton: { styleOverrides: { ... } } }</code>	Customize component appearance.
 Shadows	Customize shadow elevations.	<code>shadows: ['none', '0px 1px 3px rgba(0,0,0,0.2),...']</code>	Change shadow effects.
 Breakpoints	Define custom responsive breakpoints.	<code>breakpoints: { values: { xs:0, sm:600, md:900, lg:1200, xl:1536 } }</code>	Control responsive behavior.
 Z-Index	Set custom z-index values.	<code>zIndex: { appBar: 1100, drawer: 1200 }</code>	Manage stacking order.

Why Customize the Theme?

- * Matches your brand identity.
- * Creates consistent UI across the app.
- * Saves time and effort.
- * Easy to maintain and update.
- * Supports both light and dark modes.

8.11.3 Basic Example

```
import { createTheme, ThemeProvider, CssBaseline, Button }  
from '@mui/material';  
  
const theme = createTheme({  
  palette: {  
    primary: { main: '#1976d2' },  
    secondary: { main: '#d00040' },  
    background: { default: '#f5f5f5' }  
  },  
  typography: {  
    fontFamily: 'Poppins, Roboto, Arial',  
    h1: { fontSize: '2rem', fontWeight: 600 }  
  },  
  shape: { borderRadius: 8 },  
  spacing: 8  
});  
  
function App() {  
  return (  
    <ThemeProvider theme={theme}>  
      <CssBaseline />  
      <div style={{ padding: theme.spacing(4) }}>  
        <Button variant="contained" color="primary">  
          Primary Button  
        </Button>  
      </div>  
    </ThemeProvider>  
  );  
}  
  
export default App;
```

Output :



Light & Dark Mode Example



Key Takeaway:



Theme customization in MUI helps us build **consistent**, **beautiful** and **brand-aligned** applications by customizing colors, typography, spacing, shapes and more.

8.12 Building Professional Interfaces

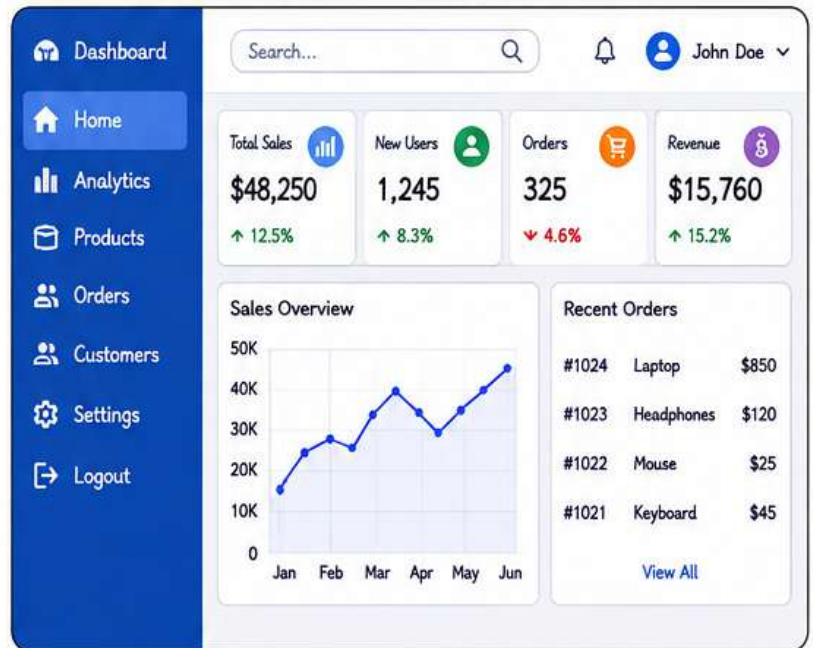
8.12.1 What is Building Professional Interfaces?

- * Building professional interfaces means creating **clean, responsive, user-friendly and visually appealing** UIs.
- * MUI provides many components and tools to help us build **modern and professional** interfaces fast.
- * A good interface improves **user experience** and leaves a strong impression.

Key Points :

- ✓ Use **consistent** colors, typography and spacing.
- ✓ Follow a **clear layout** and visual hierarchy.
- ✓ Use MUI components for **modern design**.
- ✓ Ensure **responsiveness** across devices.
- ✓ Focus on **usability, accessibility** and performance.

Example of Professional Interface (Dashboard)



8.12.2 Best Practices for Building Professional Interfaces

Practice	Description	MUI Components / Techniques	Benefit
Consistent Design	Use theme colors, typography and spacing consistently.	ThemeProvider, CssBaseline, Palette, Typography, Spacing	Creates a uniform and professional look.
Clear Layout	Organize content using proper layout and grid system.	Grid, Box, Container, Stack	Improves readability and structure.
Visual Hierarchy	Use headings, sizes, colors and spacing to guide users.	Typography, Divider, Paper, Card, Chip	Helps users focus on important information.
Responsive Design	Ensure UI works well on all screen sizes and devices.	Grid (breakpoints), Box, useMediaQuery, Stack	Better user experience on any device.
Use MUI Components	Use ready-to-use components for fast and modern UI.	AppBar, Drawer, Card, Table, Dialog, Button, Icon	Saves time and ensures modern design.
Accessibility & Usability	Use proper contrast, readable fonts and accessible components.	Button (accessible), Form controls, aria-labels	Improves usability for everyone.
Performance Optimization	Optimize images, code and component usage.	Lazy loading, Code splitting, Minimal re-renders	Faster loading and better performance.

Tips for Professional UI

- * Keep it **simple and clean**.
- * Use **white space** wisely.
- * Use **icons** to improve understanding.
- * Maintain **consistency** in colors and fonts.
- * Test on **different** devices and browsers.
- * Get user **feedback** and improve.

8.12.3 Basic Example (Dashboard Layout)

```
import { AppBar, Toolbar, Typography, Box, Grid, Card, CardContent, IconButton, Button } from '@mui/material';
import { Dashboard, People, ShoppingCart, AttachMoney } from '@mui/icons-material';

function DashboardUI() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <AppBar position="static" color="primary">
        <Toolbar>
          <Dashboard sx={{ mr: 1 }} />
          <Typography variant="h6" sx={{ flexGrow: 1 }}>Dashboard</Typography>
          <IconButton color="inherit"><People /></IconButton>
        </Toolbar>
      </AppBar>
      <Box sx={{ p: 3 }}>
        <Grid container spacing={3}>
          {
            { title: 'Users', value: '1,245', icon: <People color='primary' /> },
            { title: 'Orders', value: '325', icon: <ShoppingCart color='success' /> },
            { title: 'Revenue', value: '$15,760', icon: <AttachMoney color='secondary' /> }
          }.map((item, i) => (
            <Grid item xs={12} sm={4} key={i}>
              <Card>
                <CardContent>
                  <Box sx={{ display:'flex', alignItems:'center', gap:2 }}>
                    {item.icon}
                    <Typography variant="subtitle2" color="text.secondary">
                      {item.title}
                    </Typography>
                    <Typography variant="h5" color="text.primary">
                      {item.value}
                    </Typography>
                  </Box>
                </CardContent>
              </Card>
            </Grid>
          ))
        </Grid>
      </Box>
    </Box>
  );
}

export default DashboardUI;
```

Output :



Key Takeaway :

Building professional interfaces is about using the right components, following best practices and focusing on **usability, consistency and performance**.

