NAME: Lakshay Raina
UID: 23BAI70251
CLASS: 23AML-2B

ASSIGNMENT

1) Summarize the benefits of using design patterns in frontend development.

Design patterns are **reusable solutions to common software design problems**. In frontend development, they improve code organization, scalability, and maintainability.

Benefits

1. Reusability

- Patterns allow developers to reuse proven solutions.
- Reduces development time.
- Example: Component-based architecture in React.

2. Maintainability

- Code becomes easier to update and debug.
- Clear separation of concerns improves readability.

3. Scalability

- Applications can grow without becoming unmanageable.
- Example: Using state management patterns like Redux.

4. Consistency

- Teams follow standardized structure.
- Makes collaboration easier.

5. Testability

- Decoupled components are easier to test individually.

6. Performance Optimization

- Patterns like lazy loading and memoization improve performance.

NAME: Lakshay Raina
UID: 23BAI70251
CLASS: 23AML-2B

2) Classify the difference between global state and local state in React.

Feature	Local State	Global State
Scope	Component-specific	Shared across application
Storage	useState, useReducer	Redux, Context API
Usage	UI behavior	App-wide data
Complexity	Simple	More complex
Performance	Faster updates	May cause re-renders
Used for:	• Form inputs • Toggle buttons • Component UI state	• Authentication • Theme • Cart data • Notifications

3) Compare different routing strategies in Single Page Applications (client-side, server-side, and hybrid) and analyze the trade-offs and suitable use cases for each.

Routing Strategies in SPA

Client-side Routing

NAME: Lakshay Raina
UID: 23BAI70251
CLASS: 23AML-2B

Handled in browser using libraries like React Router.

Pros

- Fast navigation
- No page reload
- Smooth UX

Cons

- SEO issues
- Larger initial bundle

Use case

- Dashboards
- Admin panels
- Internal apps

Server-side Routing

Server returns a new HTML page for every request.

Pros

- SEO-friendly
- Faster first load

Cons

- Full page reload
- More server load

Use case

- Content websites
- Blogs
- News platforms

Hybrid Routing (SSR + CSR)

Used in frameworks like Next.js.

Pros

- SEO + SPA performance
- Better scalability

Cons

- Complex architecture

Use case

- E-commerce
- SaaS platforms
- Collaborative tools

4) Examine common component design patterns such as Container–Presentational, Higher-Order Components, and Render Props, and identify appropriate use cases for each pattern.

Container–Presentational Pattern

Separates logic from UI.

Container

- Handles data fetching
- Manages state

Presentational

- Displays UI

Use case:

- Reusable UI components
- Clean architecture

Higher-Order Components (HOC)

A function that takes a component and returns a new component.

Example:

```
withAuth(Component)
```

Use case:

- Authentication
- Logging
- Permissions

Render Props Pattern

A component shares logic using a function prop.

Example:

```
<DataProvider render={data => <UI data={data} />} />
```

Use case:

- Logic sharing
- Dynamic UI rendering

5) Demonstrate and develop a responsive navigation bar using Material UI components while applying appropriate styling and breakpoint configurations.

import React from "react";

import {

NAME: Lakshay Raina
UID: 23BAI70251
CLASS: 23AML-2B

```jsx
  AppBar,

  Toolbar,

  Typography,

  Button,

  IconButton,

  Drawer,

  Box
} from "@mui/material";

import MenuIcon from "@mui/icons-material/Menu";

import { useTheme, useMediaQuery } from "@mui/material";


export default function Navbar() {
  const theme = useTheme();

  const isMobile = useMediaQuery(theme.breakpoints.down("md"));


  const [open, setOpen] = React.useState(false);


  return (
    <AppBar position="static">
      <Toolbar>
        {isMobile && (
          <IconButton color="inherit"  onClick={() => setOpen(true)}>
            <MenuIcon />
          </IconButton>
        )}
```

NAME: Lakshay Raina
UID: 23BAI70251
CLASS: 23AML-2B

```
        <Typography variant="h6"  sx={{ flexGrow: 1 }}>

          Project Manager

        </Typography>


        {!isMobile && (

          <>

            <Button color="inherit">Dashboard</Button>

            <Button color="inherit">Projects</Button>

            <Button color="inherit">Profile</Button>

          </>

        )}

      </Toolbar>


      <Drawer open={open} onClose={() => setOpen(false)}>

        <Box sx={{ width: 200, p: 2 }}>

          <Button fullWidth>Dashboard</Button>

          <Button fullWidth>Projects</Button>

          <Button fullWidth>Profile</Button>

        </Box>

      </Drawer>

    </AppBar>

  );

}
```

6) Evaluate and design a complete frontend architecture for a collaborative project management tool with real-time updates.

NAME: Lakshay Raina
UID: 23BAI70251
CLASS: 23AML-2B

Include: a) SPA structure with nested routing and protected routes b) Global state management using Redux Toolkit with middleware c) Responsive UI design using Material UI with custom theming d) Performance optimization techniques for large datasets e) Analyze scalability and recommend improvements for multi-user concurrent access.

A collaborative project management tool allows multiple users to manage projects and tasks with real-time updates. The frontend must be scalable, responsive, and efficient.

a) SPA structure with nested routing and protected routes
The application is developed as a Single Page Application (SPA) using React to provide faster navigation without page reloads.
Nested routing is used to organize related pages under a common layout, such as project boards, task lists, and settings.
Protected routes ensure only authenticated users can access sensitive pages using token-based authentication and role-based access control.

b) Global state management using Redux Toolkit with middleware
Redux Toolkit is used for centralized state management.
State is divided into slices such as authentication, projects, tasks, and UI.

c) Responsive UI design using Material UI with custom theming
The user interface is built using Material UI (MUI).
Components like AppBar, Drawer, Cards, and Grid are used.
Custom themes define colors, typography, and light/dark modes.
MUI breakpoints ensure responsiveness across mobile, tablet, and desktop devices.

d) Performance optimization techniques for large dataset
To improve performance with large datasets:
List virtualization is used to render only visible data.
Pagination and infinite scrolling reduce load time.
Memoization prevents unnecessary re-renders.
Lazy loading loads components only when required.

NAME: Lakshay Raina
UID: 23BAI70251
CLASS: 23AML-2B

e)Scalability analysis and recommendations for multi-user access
To support multiple users:
Optimistic UI updates improve responsiveness.
Role-based data access reduces unnecessary data transfer.
State caching minimizes repeated server requests.