**React js**

React is a popular JavaScript library for building user interfaces (UIs) on the web. It was developed by Facebook and released as an open-source project in 2013. React allows developers to create reusable UI components and manage the state of an application using a declarative programming model.

Some **key features** of React include:

**Virtual DOM**: React uses a virtual representation of the actual DOM, which is faster and more efficient than traditional DOM manipulation.

**Component-based architecture**: React allows developers to break down an application into small, reusable components, which can be used across multiple projects.

**JSX**: React allows developers to write HTML-like syntax called JSX, which can be combined with JavaScript to create UI elements.

**Unidirectional data flow**: React uses a unidirectional data flow, where data flows down from a parent component to its child components. This helps to keep the application state predictable and easier to manage.

React has a large and active community, with many third-party libraries and tools available for building and deploying React applications. It is commonly used for building single-page applications (SPAs), mobile applications, and desktop applications.

**Folder structure/Architecture**

The folder structure for a React application can vary depending on the specific project and the team's preferences. However, there are some common practices that developers follow when organizing the files and folders for a React application. Here's a commonly used folder structure:



**node\_modules:** This folder contains all of the project's dependencies installed via NPM or Yarn.

**public:** This folder contains the public files that are served by the web server, such as the HTML file, favicon, and other static assets.

**src:** This folder contains the source code for the React application.

components: This folder contains reusable components that can be used throughout the application.

**containers:** This folder contains higher-level components that serve as pages or sections of the application.

**services:** This folder contains modules that interact with external services, such as APIs or authentication services.

**assets:** This folder contains non-code assets, such as images and stylesheets.

**utils:** This folder contains utility functions that can be used throughout the application.

**App.js:** This is the top-level component that is rendered by the React application.

**index.js:** This is the entry point for the application, where the React app is rendered to the DOM.

**index.css:** This is the main CSS file for the application.

**package.json**: This file contains metadata and dependencies for the project.

**README.md**: This file contains documentation and instructions for the project.

**React Hooks**

React Hooks are a feature introduced in React 16.8 that allow developers to use state and other React features in functional components. Before hooks, state could only be managed in class components using this.state and this.setState(). With hooks, developers can manage state and other React features such as context and lifecycle methods in functional components.

Some of the commonly used hooks are:

**useState():** This hook allows developers to manage state in a functional component. It takes an initial state value as an argument and returns an array with two elements, the current state value and a function to update the state.

**useEffect():** This hook allows developers to handle lifecycle methods in functional components. It takes a function as an argument and runs it after every render.

**useContext()**: This hook allows developers to consume a React context in a functional component.

**useReducer():** This hook allows developers to manage state using a reducer function instead of setState(). It is useful for managing complex state with multiple values.

**useCallback():** This hook allows developers to memoize functions to avoid unnecessary re-renders of child components.

**useMemo()**: This hook allows developers to memoize the result of a computation to avoid unnecessary re-computations.

Hooks have many benefits, including making it easier to share stateful logic between components and reducing the need for class components. They also allow developers to write more reusable and composable code.

**React Router DOM**

React Router DOM is a package that provides a routing solution for single-page applications built with React. It allows you to define routes in your application and render different components based on the URL of the page.

React Router DOM is built on top of React Router, which is a routing library for React. It provides a set of components that you can use to define the different routes in your application. The main components provided by React Router DOM are:

**<Route>:** This component defines a specific route in your application. It takes a path prop that defines the URL for the route, and a component prop that specifies the component to render when the route is matched. You can also use the children prop to render content for the route.

**<Link> and <NavLink>:** These components allow you to create links that navigate to specific routes in your application. <Link> is a basic component that renders an <a> tag with the href attribute set to the specified route. <NavLink> is similar, but it also applies an active class to the link when it matches the current route.

**<Outlet>:** This component is used to render child routes for a parent route. It is typically used in conjunction with the <Route> component, where child routes can be nested within a parent route.

**<Redirect>:** This component allows you to redirect to another route in your application. It takes a to prop that specifies the route to redirect to.

React Router DOM also provides a number of other components and APIs to help you build more complex routing logic, such as nested routes, dynamic routes, and protected routes.

Overall, React Router DOM is a powerful and flexible tool for managing the routing in your React application. It is widely used in the React community and is well-documented, making it easy to get started with.

**MUI (formerly known as Material-UI)**

MUI (formerly known as Material-UI) is a popular open-source UI library for React. It provides a set of reusable components, styles, and utilities that help developers quickly build beautiful and responsive user interfaces.

MUI is built on top of Google's Material Design guidelines, which provide a consistent visual language for UI design across different platforms and devices. MUI provides a set of components that follow these guidelines, such as buttons, cards, menus, and form controls. These components are highly customizable and can be easily styled using MUI's theme system and CSS-in-JS approach.

Some of the key features of MUI include:

**Accessibility**: MUI places a strong emphasis on accessibility, ensuring that its components are usable by everyone, regardless of their abilities or disabilities.

**Responsive design**: MUI provides a responsive design system that allows components to adapt to different screen sizes and devices.

**Customizability**: MUI's components are highly customizable, with a wide range of options for changing their appearance and behavior.

**Theming**: MUI's theme system allows developers to define a consistent set of styles for their application, making it easy to maintain a coherent design language across different components and screens.

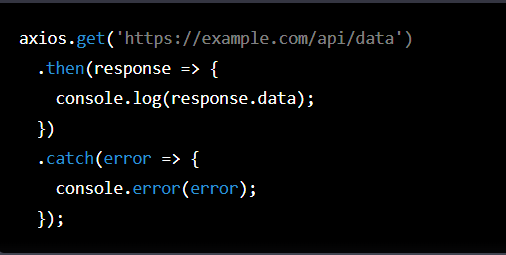
MUI also provides a number of other utilities and tools to help developers build responsive, accessible, and beautiful user interfaces, such as a grid system, typography components, and icons. Overall, MUI is a powerful and flexible library for building UIs in React, and is widely used in the React community.

**Axios**

Axios is a popular open-source JavaScript library used to make HTTP requests from a web browser or Node.js. It provides an easy-to-use API for making HTTP requests to servers, and it supports various request methods such as GET, POST, PUT, DELETE, and more.

Axios can be used in both client-side and server-side applications, and it also supports various features such as interceptors, automatic transformation of request and response data, cancellation of requests, and more.

Axios is a popular choice for developers because it provides a simpler and cleaner API for making HTTP requests compared to traditional methods such as XMLHttpRequest. It also provides a promise-based interface, making it easy to handle asynchronous requests in modern JavaScript applications.



Project

-scaleable folder structure

-Routing using react router dom

-reminder functionality

-fetching data using axios

-mui reusable component