**Creating the client**

**Step-1 :** Start by creating the react project with npm createvite@latest and name the project client

**Step-2 :** Go to the client folder and install all the dependencies npm install

**Step-3 :** Install all the dependencies required for the react project like redux-toolkit, etc.

**Step-4 :** Install the tailwind Css for shadcn cli to function (npm install tailwindcss @tailwindcss/vite)

**Step-5 :** follow the shadcn guide and makes changes in index.css and vite.config

**Step-5-1 :** Create a tailwind.config.js file and copy-paste the necessary configuration. If you're using Tailwind CSS v4

**Step-6 :** create a new file jsconfig.json (since you’re using JavaScript, not TypeScript) and copy what is inside of ts.config file in the schadcn documentation, thus it validates import alias

**Step-7 :** Setting up the React router Dom , inside the main.js wrap the app inside browser router

**Step-8 :** Setting up the redux store and slice, create a store folder in the src, create a store.js file in the store

**Step-9 :** Inside the store folder, create a folder auth-slice, then create a file index.js inside it

**Step-10 :** In auth-slice/index.js, define a basic slice auth and export the action and reducer  
**Step-11 :** In store.js, create a Redux store using configureStore from Redux Toolkit.  
Add your reducers inside the store as a global reducer, and export the configured store

**Step-12 :** Wrap the <App /> component in main.js with the Redux <Provider> and pass in the store.  
This makes the Redux store available throughout the entire application

**Step-13 :** Creating the Pages folder inside the client and then create the auth folder inside the pages folder for creating the pages

**Step-14 :** Create auth folder inside the components folder, because every component we use in page folder will be in auth folder in components

**Step-15 :** Create layout.jsx file inside the auth folder (each page will have it own layout ) in components, Create login.jsx and register.jsx inside the auth folder in Pages

Step-16 : Create a layout component in AuthLayout.jsx Inside this file, define a basic layout and include the <Outlet /> component from react-router-dom to render nested routes

Step-17 : Configure the App.jsx file by setting up your routes using react-router-dom. Import AuthLayout and nest your auth-related routes (like Login, Register, etc.) inside it using the <Routes> and <Route> components

**Step-18 :** Create shopping-view and admin-view folders inside both the pages and components directories, components used in these pages will be stored in component directories

**Step-19 :** Create a layout.jsx file inside the components/admin-view folder. Set up the basic admin layout by importing and using the AdminHeader and AdminSidebar components. At the end of the layout, include <Outlet /> to render nested route components

**Step-20 :** Create header.jsx and sidebar.jsx inside the components/admin-view folder. Use these components inside layout.jsx to complete the admin layout structure

**Step-21 :** Create another Route inside and app.jsx and configure the admin pages and components

**Step-22 :** Create a dashboard.jsx, products.jsx, features.jsx, orders.jsx inside pages/admin-view folder