Step-1: Initialize the project backend by //npm init -y , then add some modules like // npm i express mongoose bcrypt cors cookie-parser, then create a server // server.js, inside server.js import the modules, create the app, define the port and start the server

Step-2: Configure the database, connect with MongoDB using // mongoose.connect, configure the cors, and add the middlewares // app.use(express.json()), // app.use(cookieParser())

Step-3: Create a user model inside // User.js, in models folder, then create a register controller function inside // auth-controller.js, in auth/controllers folder

Step-4: Create routes inside //authRoutes.js, in auth/routes folder, then do a post route for register using router and // RegisterUser, from controller give it a path /register, then export the router

==== Moving to Client ====

Step-5: Initialize the vite project // npm create vite@latest, install modules like // npm i react-redux @reduxjs/toolkit react-router-dom, then set up the tailwind // npm install tailwindcss @tailwindcss/vite, create a // tailwind.config.js, copy and paste // tsconfig.json, in the // jsconfig.json, also configure the // vite.config.js, accordance to schadcn documentation, then do // npx shadcn@latest init, and follow the documentation

Step-6: Cover the app in the main.jsx with browser router from react router dom, then initialize the redux by creating a store folder inside src, create // store.js, file and // auth-slice, folder inside the store, create //index.js file inside it

Step-7: Create a slice using // createSlice, from redux toolkit, define the initial state and pass it in the slice with reducers, export the reducers and actions, then configure the store with //configureStore, from redux toolkit by passing the auth slice as // authReducer, export the store and wrap the app in main.jsx with provider containing store

Step-8: Create a pages folder inside client, inside it create auth folder with // login.jsx and register.jsx, file also create auth folder inside component with // layout.jsx, create a basic layout for auth pages, then configure the routes for them in main

Step-9: Create complete folder structure in pages and components folders and configure the routing for them, create // privateView, publicView, adminView, editorView, folders inside the pages and components then create // privateDoc.jsx, privateListing.jsx, publicDoc.jsx, publicListing.jsx, adminPanel.jsx, editorPanel.jsx, for folders inside pages and also create // layout.jsx, header.jsx for each folder in components, then configure the routing for each page inside the // app.jsx

Step-10: Create a common folder inside the components folder, create a // checkAuth.jsx file inside the common folder, configure the checkAuth and wrap each page route with checkAuth file to finalize the routing

Step-11: