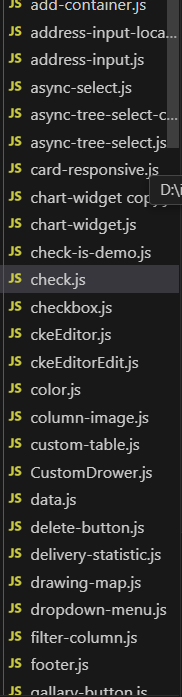
*Folder structue*

Let’s alalyze the code structure of the admin app how components, services, and utilities are organized.  
The src is the main folder to store many components of the project and they are catagorized inside it.

The first folder is the assets folder which has 6 folders inside it the audio,css,images,rive,scss, and video this are resources containing css and some assets as the name suggests.

The next folder is components which contains forms, reportss, shop, todo and upload folder inside it. It also has a lot of components outside this folders, in the root folder(components): shown in the image below.

. 

The next folder is the configs folder which has 5 components.

* app-global.js: Manages various configuration settings and constants used throughout the application.
* app-local.js: Manages localization settings, allowing the application to support multiple languages.
* I18next.js: is a configuration file for setting up internationalization (i18n) in the application using the i18next library along with react-i18next. This setup allows the application to support multiple languages and dynamically switch between them based on user preferences or browser settings.
* [menu-config.js](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "): is a configuration file that defines the menu structure for different user roles in the application. It includes routes and submenus for various roles such as admin, manager, seller, moderator, deliveryman, waiter, and parcel. Each route is defined with properties like name, icon, URL, ID, and children.
* [theme-config.js](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") file defines the theme-related settings for the application. These settings include the locale, colors, theme mode, layout direction, and other UI-related configurations.

The next folder is constants folder. The constants folder contains several configuration files that define various constants used throughout the application. These files include:

index.js: Defines various constants such as mock data, status lists, default values, and banner types.

It Provides a centralized location for defining configuration settings and constants used throughout the application.

ChartConstant.js: Defines various constants and default options for different types of charts.

It Ensures consistency in the appearance and behavior of charts throughout the application.

[shop-types.js](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "): Defines the different types of shops used in the application.

It Provides a centralized location for defining shop types, making it easier to manage and update these values as needed.

The next folder is the context folder which contains 4 files.

[context.js](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "): defines a React context and a context provider component for managing global state related to the theme and modal visibility in the application. This setup allows different components in the application to access and update the theme and modal visibility state without having to pass props down through multiple levels of the component tree.

path-logout.js: defines a React component called [PathLogout](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") that handles user redirection based on their authentication status and the active menu item. This component uses React hooks and Redux to manage state and navigation.

[protected-route.js](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "): defines a component that protects routes by checking the user's authentication status. It ensures that only authenticated users can access certain routes.

[report.js](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "): defines a React context and a context provider component for managing the state related to reports in the application. This setup allows different components in the application to access and update the report-related state without having to pass props down through multiple levels of the component tree.

The next folder is the helpers folder. The helper files in the helpers folder provide essential utility functions and custom hooks that enhance the functionality and maintainability of the application. By encapsulating common tasks and reusable logic, these helper files promote code reuse and simplify the development process. Each helper file serves a specific purpose, from managing query parameters and detecting dark themes to calculating order amounts and creating pre-configured Axios instances. This setup ensures that the application remains modular, scalable, and easy to maintain.

The next is the layout folder. Which contain the following 2 components.

app-layout.js: The AppLayout component, located in the [app-layout.js](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") file, serves as the primary layout for the application, integrating various subcomponents and managing the overall layout and content dynamically based on the user's role and application state. Utilizing Redux hooks such as useDispatch and [useSelector](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "), the component dispatches actions to fetch necessary data like languages, shops, and currencies, and selects relevant state slices from the Redux store. The component employs the [useEffect](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") hook to perform side effects, including fetching data and setting user-specific configurations upon mounting. The layout is composed of several key elements: a sidebar for navigation, a content area that leverages React Router's Outlet for nested routes, and additional components like TabMenu, SubscriptionsDate, and Footer. Depending on the user's role, it conditionally renders components such as ChatIcons and ParcelFloat, ensuring a tailored user experience. The layout's responsiveness is managed through functions like getLayoutGutter and getLayoutDirectionGutter, which adjust padding and layout direction based on the navigation state and theme settings. Overall, the AppLayout component provides a cohesive and adaptive framework that supports the application's dynamic content and user interactions.

[welcome-layout.js](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "): The [WelcomeLayout](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") component, found in the [welcome-layout.js](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") file, handles the initial setup and redirection logic for the application, ensuring users are appropriately directed based on their authentication status and the application's initialization state. This component leverages Redux hooks to access the current user and active menu state from the Redux store, and uses the [useNavigate](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") hook from React Router for programmatic navigation. Upon mounting, the [useEffect](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") hook triggers a check for the application's initialization file via the [installationService](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "), setting a loading state during this process. If the initialization file is found, the user is navigated to the root URL. The component conditionally renders a loading indicator if the application is still in the loading state, ensuring a smooth user experience. Once the loading state is resolved, authenticated users are redirected to the appropriate menu URL based on their role, while unauthenticated users are allowed to proceed with the child components. By managing these initial checks and redirections, the [WelcomeLayout](vscode-file://vscode-app/c:/Users/ermid/AppData/Local/Programs/Microsoft VS Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") component ensures that the application is properly initialized and that users are seamlessly guided to the correct starting point.

Till now I have found an empty component I don’t know if that is intentional or not so I think I have to report this. It is in the helpers folder which is CheckforEquality.js .