/\*\*

\* @**projct\_name**: insignia-pos

\* @Version: 0.1.0

\* @Author: API (Advanced Project Integration)

\* @Created\_by: Kawsar Bin Siraj

\* @Created\_at: 10/19/2022

\* @Description: This document summarizes how to insignia pos actually works!

\*/

Core Technologies:

1. **react.js**

(A JavaScript library for building user interfaces)

|  |  |
| --- | --- |
|  |  |
| * Folder Architecture |  |
|  |  |

* public
* src
  + app-helpers
    - appHelpers.js
  + assets
    - img
  + components
    - ApplicationHelmet
    - Header
    - RestaurantPos
    - RoomServicePos
  + error-boundary
    - ErrorBoundary.js
  + libraries
    - AuthLibrary.js
  + Pages
    - CardPos
    - RestaurantPos
    - RoomServicePos
    - Dashboard
    - Login
    - NotFound
  + redux-store
    - reducers
    - Store.js
  + routes
    - ProtectedRoutes.js
    - Routes.js
  + Utilities
    - Loader.js
  + App.js
  + App.scss
  + index.js
  + index.scss
* .env
* .gitignore
* package.json
* README.md
* LICENSE
* Libraries we used

"@hookform/resolvers": "^2.8.8",

"@reecelucas/react-use-hotkeys": "^1.3.3",

"@testing-library/jest-dom": "^5.16.4",

"@testing-library/react": "^11.2.7",

"@testing-library/user-event": "^12.8.3",

"axios": "^0.24.0",

"bootstrap": "^5.1.3",

"cookies-js": "^1.2.3",

"dotenv": "^10.0.0",

"lodash": "^4.17.21",

"node-sass": "^6.0.1",

"prop-types": "^15.8.1",

"react": "^17.0.2",

"react-bootstrap": "^2.2.3",

"react-dom": "^17.0.2",

"react-helmet": "^6.1.0",

"react-hook-form": "^7.29.0",

"react-icons": "^4.3.1",

"react-loading": "^2.0.3",

"react-outside-click-handler": "^1.3.0",

"react-promise-tracker": "^2.1.0",

"react-redux": "^7.2.8",

"react-router-dom": "^6.3.0",

"react-scripts": "^4.0.3",

"react-select": "^5.3.2",

"react-to-print": "^2.14.7",

"react-toastify": "^8.2.0",

"react-tooltip": "^4.2.21",

"redux": "^4.1.2",

"redux-persist": "^6.0.0",

"use-sound": "^4.0.1",

"uuid": "^8.3.2",

"web-vitals": "^1.1.2",

"yup": "^0.32.11"

* Functional Documents
* **posAccessPermissions ()**

@description: we use useSelector redux hook for getting pos access list after getting we set dashboard home

* **categoryWidgetOpen (**e, menuName**)**

@param: e, menuName

@description: this is an asynchronous method to open category and getting menu items. after getting menus it has been stored into setFoodNames state. menuName is the dependency @param to send xmlhttp request for getting menus.

* **searchCategoryOnChangeHandler (**searchValue**)**

@param: searchValue

@description: this is an asynchronous method for server side searching products by category via axios. After getting all products result stored into setProductsByCatgory state.

* **searchSubCategoryOnChangeHandler (**product**)**

@param:product

@description: this is an asynchronous method for server side searching depending on product. This method gives us a product details. After getting result modal will be open with product details. You can add to cart form this modal.

* **getAllCategoryAndSubCategory ()**

@description: for getting all product category and sub category we can use this method.

* **productOnChangeHandler ()**

@description: this method execute when product quantity has been changed. Its automatically calculated other values depends on quantity.

* **adOnsOnChangeHandler** **(addonId, price, e)**

@param: addonId, price, e

@description: this method immediately invoked when product addons change. when select or deselect any addons this method will work on that event also auto added checked addons in add to cart product bundle.

* **onSubmit (e)**

@param: e

@description. When you submit for add to cart this method will be call. Figure out every input value as object and stored it in global store.

* **getComboProducts ()**

@param:

@description. This is an asynchronous method for getting all combo products. After getting we stored with re-organize

* **getCurrentOrders ()**

@param:

@description. This method used for getting all current order

* **getCurrentOrdersByStatus ()**

@param:

@description. This method used for getting all current order by status. (Ordered, deleted)

* **makePayment (invoiceId)**

@param: invoiceId

@description. This method ensures make payment after create order and automatically updated

* **cancelOrder (invoiceId)**

@param: invoiceId

@description. This method will cancel order with the param of invoiceId.

* **makePrintPreview (invoiceId)**

@param: invoiceId

@description. This method works for print preview in the markup.

* **makeKitchenPrintPreview (invoiceId)**

@param: invoiceId

@description. This method works for print preview in the markup only for kitchen. It’s called KOT (kitchen of token).

* **editOrder (invoiceId)**

@param: invoiceId

@description. This method helps to edit order. Dispatched editable ordered data in global store and navigate main page.

* **calcGrandTotal ()**

@param:

@description. Calculate total amount, tax, service charge

* **sendInvoiceDetailsToOrder ()**

@param:

@description. Send invoice details to order. And success alert show when order is successfully created

* **newCustomerFormOnSubmit ()**

@param:

@description. To create new customer, we have used a new custom modal. This method add a new custom with this modal form submission.

* **getCustomerByType (type)**

@param: type

@description. For getting customer details by customer type you can use this method.

* **getAllTables ()**

@param:

@description. This is an asynchronous method for getting all table as a listed array

* **getAllWaiter ()**

@param:

@description. This is an asynchronous method for getting all waiter as a listed array

Know issues.

1. Authorization is not checked.
2. OOP is not followed