**Dhanashreesurgicals**

**Technology Stack**

**Backend:**NodeJs – Version 16 [Not Upgradable]  
ExpressJs – Version 4.18.1  
Sequelize – Version 6.21.0  
Packages:

  "dependencies": {

    "axios": "^0.27.2",

    "bcryptjs": "^2.4.3",

    "body-parser": "^1.20.0",

    "cors": "^2.8.5",

    "express": "^4.18.1",

    "firebase-admin": "^11.0.0",

    "jsonwebtoken": "^8.5.1",

    "mysql2": "^2.3.3",

    "node-schedule": "^2.1.0",

    "nodemon": "^2.0.16",

    "sequelize": "^6.21.0"

  }

**Front-End:**Uses mantine framework for frontend development

**Need**: react 18.2.0 mantine 5.4.0

Dependencies Package.json

 "dependencies": {

    "@emotion/react": "^11.10.4",

    "@mantine/carousel": "^5.4.0",

    "@mantine/core": "^5.4.0",

    "@mantine/dates": "^5.4.0",

    "@mantine/dropzone": "^5.4.0",

    "@mantine/form": "^5.4.0",

    "@mantine/hooks": "^5.4.0",

    "@mantine/modals": "^5.4.0",

    "@mantine/notifications": "^5.4.0",

    "@mantine/nprogress": "^5.4.0",

    "@mantine/prism": "^5.4.0",

    "@mantine/rte": "^5.4.0",

    "@mantine/spotlight": "^5.4.0",

    "@tabler/icons": "^1.96.0",

    "@testing-library/jest-dom": "^5.16.5",

    "@testing-library/react": "^13.4.0",

    "@testing-library/user-event": "^13.5.0",

    "axios": "^0.27.2",

    "dayjs": "^1.11.5",

    "embla-carousel-react": "^7.0.3",

    "jspdf": "^2.5.1",

    "jspdf-autotable": "^3.5.25",

    "react": "^18.2.0",

    "react-apexcharts": "^1.4.0",

    "react-csv": "^2.2.2",

    "react-dom": "^18.2.0",

    "react-router-dom": "^6.2.0",

    "react-scripts": "5.0.1",

    "read-excel-file": "^5.5.0",

    "tabler-icons-react": "^1.55.0",

    "web-vitals": "^2.1.4"

  },

**Working progress**

🡪 Clients 🡪 Group🡪 HSN 🡪 Product 🡪 Invoice 🡪Reports 🡪 Proforma🡪Delivery Challan

**App entry point**

**1.0 Index.js**

Index.js stores main Render call from React-Dom It imports the App.js component Which tells React where to render it

import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App";

import "./assets/style.css";

import { HashRouter } from "react-router-dom"; // for import react dom navigation components

import { MantineProvider } from "@mantine/core"; //for import mantine required functions and theme

import { ModalsProvider } from "@mantine/modals";

import { createBrowserHistory } from "history";

import reportWebVitals from "./reportWebVitals";

const root = ReactDOM.createRoot(document.getElementById("root"));

const history = createBrowserHistory({ window });

root.render(

  <HashRouter history={history}>

    <MantineProvider withNormalizeCSS withGlobalStyles>

      <ModalsProvider>

        <App />

      </ModalsProvider>

    </MantineProvider>

  </HashRouter>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

**1.1 App.js**

This is the file for App component . App Component is the main component in React which acts as a container for all the other components. It contains the routes for navigation

App.js file

import React from "react";

import { Route, Routes } from "react-router-dom";

import { NotificationsProvider } from "@mantine/notifications"; // For show the notification import

import { ColorSchemeProvider, MantineProvider } from "@mantine/core";

import { useLocalStorage } from "@mantine/hooks";

import ProtectedRoute from "./components/ProtectRoute";

import ErrorBoundary from "./components/Error";

import Sidebar from "./components/Sidebar";

import Login from "./pages/auth/Login";

// Import pages

import Home from "./pages/Home";

import Client from "./pages/Client";

import Product from "./pages/Product";

import CreateInvoice from "./pages/invoice/CreateInvoice";

import Invoice from "./pages/invoice/Invoices";

import PrintInvoice from "./pages/invoice/PrintInvoice";

import SaveInvoice from "./pages/invoice/SaveInvoice";

import SaveEdit from "./pages/invoice/SaveEdit";

import CreateProforma from "./pages/invoice/CreatePerformaInvoice";

import ProformaInvoice from "./pages/invoice/ProformaInvoice";

import PrintProforma from "./pages/invoice/PrintProforma";

import DeliveryChallan from "./pages/invoice/DeliveryChallan";

import WeekReport from "./pages/report/Week";

import GSTReport from "./pages/report/GST";

import Monthly from "./pages/report/Monthly";

import Daily from "./pages/report/Daily";

import Manages from "./pages/settings/Manage";

import Tax from "./pages/settings/Tax";

import Profile from "./pages/settings/Profile";

import Account from "./pages/settings/Account";

import Groups from "./pages/Group";

import ChallanPrint from "./pages/invoice/ChallanPrint";

function App() {

  const toggleColorScheme = (value) =>

    setColorScheme(value || (colorScheme === "dark" ? "light" : "dark"));

  const [colorScheme, setColorScheme] = useLocalStorage({

    key: "mantine-color-scheme",

    defaultValue: "light",

  });

  const [color] = useLocalStorage({

    key: "color",

    defaultValue: "#043c64",

  });

  const LightenDarkenColor = (col, amt) => {

    var usePound = false;

    if (col[0] === "#") {

      col = col.slice(1);

      usePound = true;

    }

    var num = parseInt(col, 16);

    var r = (num >> 16) + amt;

    if (r > 255) r = 255;

    else if (r < 0) r = 0;

    var b = ((num >> 8) & 0x00ff) + amt;

    if (b > 255) b = 255;

    else if (b < 0) b = 0;

    var g = (num & 0x0000ff) + amt;

    if (g > 255) g = 255;

    else if (g < 0) g = 0;

    return (usePound ? "#" : "") + (g | (b << 8) | (r << 16)).toString(16);

  };

  return (

    // Colorscheme provider is used for toggling between light and dark modes

    <ColorSchemeProvider

      colorScheme={colorScheme}

      toggleColorScheme={toggleColorScheme}

    >

      {/\* Mantine provider is used for customizing our own theme \*/}

      <MantineProvider

        theme={{

          "::-webkit-scrollbar": {

            backgroundColor: LightenDarkenColor(color, 120),

            width: "5px",

            height: "10px",

            borderRadius: 5,

          },

          "::-webkit-scrollbar-thumb": {

            backgroundColor: LightenDarkenColor(color, 120),

            borderRadius: 5,

            // "#D50000"

          },

          fontFamily: "Inter",

          fontWeight: 300,

          colorScheme,

          colors: {

            zevcore: [

              LightenDarkenColor(color, 140),

              LightenDarkenColor(color, 130),

              LightenDarkenColor(color, 120),

              LightenDarkenColor(color, 110),

              LightenDarkenColor(color, 100),

              LightenDarkenColor(color, 90),

              LightenDarkenColor(color, 50),

              LightenDarkenColor(color, 80),

              LightenDarkenColor(color, 50),

              LightenDarkenColor(color, 20),

            ],

          },

        }}

      >

        <NotificationsProvider autoClose={3000}>

          <ErrorBoundary>

            <Routes>

              <Route path="/login" element={<Login />} />

              <Route element={<Sidebar />}>

                <Route

                  exact

                  path="/"

                  element={

                    <ProtectedRoute>

                      <Home />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/client"

                  element={

                    <ProtectedRoute>

                      <Client />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/group"

                  element={

                    <ProtectedRoute>

                      <Groups />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/products"

                  element={

                    <ProtectedRoute>

                      <Product />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/create\_invoice"

                  element={

                    <ProtectedRoute>

                      <CreateInvoice />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/invoice"

                  element={

                    <ProtectedRoute>

                      <Invoice />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/save\_invoice"

                  element={

                    <ProtectedRoute>

                      <SaveInvoice />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/edit\_invoice/:id"

                  element={

                    <ProtectedRoute>

                      <SaveEdit />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/create\_proforma"

                  element={

                    <ProtectedRoute>

                      <CreateProforma />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/proforma"

                  element={

                    <ProtectedRoute>

                      <ProformaInvoice />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/week\_report"

                  element={

                    <ProtectedRoute>

                      <WeekReport />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/gst\_report"

                  element={

                    <ProtectedRoute>

                      <GSTReport />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/month\_report"

                  element={

                    <ProtectedRoute>

                      <Monthly />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/daily\_report"

                  element={

                    <ProtectedRoute>

                      <Daily />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/manages"

                  element={

                    <ProtectedRoute>

                      <Manages />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/tax"

                  element={

                    <ProtectedRoute>

                      <Tax />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/profile"

                  element={

                    <ProtectedRoute>

                      <Profile />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/account"

                  element={

                    <ProtectedRoute>

                      <Account />

                    </ProtectedRoute>

                  }

                />

                <Route

                  exact

                  path="/delivery\_challan"

                  element={

                    <ProtectedRoute>

                      <DeliveryChallan />

                    </ProtectedRoute>

                  }

                />

              </Route>

              <Route

                exact

                path="/invoice\_print/:id"

                element={

                  <ProtectedRoute>

                    <PrintInvoice />

                  </ProtectedRoute>

                }

              />

              <Route

                exact

                path="/invoice\_proforma/:id"

                element={

                  <ProtectedRoute>

                    <PrintProforma />

                  </ProtectedRoute>

                }

              />

              <Route

                exact

                path="/challan\_print/:id"

                element={

                  <ProtectedRoute>

                    <ChallanPrint />

                  </ProtectedRoute>

                }

              />

            </Routes>

          </ErrorBoundary>

        </NotificationsProvider>

      </MantineProvider>

    </ColorSchemeProvider>

  );

}

export default App;

**2.0 Sidebar**

Main sidebar for application.it will contain all the navigation list and dark light mode toggle button and user profile details settings and logout

Without the sidebar application will not work

import {

  ActionIcon,

  AppShell,

  Burger,

  Header,

  MediaQuery,

  Navbar,

  ScrollArea,

  useMantineTheme,

  Menu,

  Text,

  Button,

  Group,

  Code,

  TextInput,

  Divider,

  SegmentedControl,

  Center,

  Box,

  HoverCard,

  Transition,

  Paper,

  Indicator,

} from "@mantine/core"; //for import mantine required functions and theme

import React, { useState, useEffect } from "react";

import {

  MoonStars,

  Sun,

  Search,

  Settings,

  UserExclamation,

  Logout,

  X,

  WindowMinimize,

  WindowMaximize,

  Refresh,

  Minus,

  Archive,

  ChartLine,

  User,

  Dashboard,

  RotateClockwise2,

  Cash,

  BoxMultiple,

  Download,

  Bell,

} from "tabler-icons-react";

import { IconSearch } from "@tabler/icons";

import { useLocalStorage, useClickOutside } from "@mantine/hooks";

import { useModals } from "@mantine/modals";

import { useNavigate, Outlet } from "react-router-dom"; // for import react dom navigation components

import {

  SpotlightProvider,

  useSpotlight,

  openSpotlight,

} from "@mantine/spotlight";

// For imports the logo images for navigation

import Icon from "../assets/images/icon.png";

// Import thte linksgroup

import { LinksGroup } from "./NavLinks";

import useStyles from "./Style"; // Import the mantine custome styles from the compoents

//for   made mantine theme style change and write custome theme here

import Home from "../pages/Home";

import Navigation from "./Navigation";

// For control the search of naviagation list

function SpotlightControl(props) {

  const spotlight = useSpotlight();

  return (

    <Group position="center" color="dark">

      <TextInput

        variant="filled"

        placeholder="Search"

        onClick={spotlight.openSpotlight}

        size="xs"

        width={50}

        shortcut={["ctrl + k", "mod + K", "⌘ + K"]}

        icon={<IconSearch size={12} stroke={1.5} />}

        rightSectionWidth={70}

        rightSection={<Code>Ctrl + K</Code>}

        styles={{ rightSection: { pointerEvents: "none" } }}

        mb="sm"

      />

    </Group>

  );

}

const scaleY = {

  in: { opacity: 1, transform: "scaleY(1)" },

  out: { opacity: 0, transform: "scaleY(0)" },

  common: { transformOrigin: "top" },

  transitionProperty: "transform, opacity",

};

// Side bar navigation list by array

const mockdata = [

  { label: "Dashboard", icon: Dashboard, link: "/" },

  {

    label: "Sales",

    icon: ChartLine,

    initiallyOpened: true,

    links: [

      { label: "Add Sale", link: "/add\_sale" },

      { label: "Sales", link: "/sale" },

      { label: "Add Estimate", link: "/add\_estimate" },

      { label: "Estimates", link: "/estimate" },

      { label: "KPI", link: "/kpi" },

      // { label: "HSN Report", link: "/hsn" },

    ],

  },

  {

    label: "Inventory",

    icon: Archive,

    initiallyOpened: true,

    links: [

      { label: "Vendors", link: "/vendors" },

      { label: "Stock In", link: "/stock\_in" },

      { label: "Stock View", link: "/stock" },

      { label: "Purchase Order", link: "/po" },

    ],

  },

  {

    label: "Cash Ledger",

    icon: Box,

    initiallyOpened: true,

    links: [

      { label: "Cash Ledger ", link: "/cash" },

      { label: "Search Cash Ledger", link: "/search\_cash" },

    ],

  },

  {

    label: "Products",

    icon: Box,

    initiallyOpened: true,

    links: [

      { label: "Brands", link: "/brands" },

      { label: "Groups", link: "/groups" },

      { label: "SubGroups", link: "/subgroups" },

      { label: "Products", link: "/products" },

    ],

  },

  { label: "Customers", icon: User, link: "/customers" },

  { label: "Settings", icon: Settings, link: "/manages" },

];

// Sidebar start

function Sidebar({ children }) {

  const [opened, setOpened] = useState(false);

  const [customerDrawer, setCustomerDrawer] = useState(false);

  const [stockDrawer, setStockDrawer] = useState(false);

  const [minimize, setMinimize] = useState(false);

  const [name, setName] = useState("");

  const theme = useMantineTheme();

  const clickOutsideRef = useClickOutside(() => setName(""));

  const [colorScheme, setColorScheme] = useLocalStorage({

    key: "mantine-color-scheme",

    defaultValue: "light",

  });

  const toggleColorScheme = (value) =>

    setColorScheme(value || (colorScheme === "dark" ? "light" : "dark"));

  const dark = colorScheme === "dark";

  const { classes } = useStyles();

  const links = mockdata.map((item) => (

    <LinksGroup {...item} key={item.label} />

  ));

  const [note, setNote] = useState(true);

  const [pCount, setPCount] = useLocalStorage({

    key: "depletion",

    defaultValue: 0,

  });

  const [notificationDrawer, setNotificationDrawer] = useState(false);

  useEffect(() => {

    let mounted = true;

    const fetchData = async () => {};

    fetchData();

    return () => {

      mounted = false;

    };

  });

  const modals = useModals();

  let navigate = useNavigate();

  const actions = [

    {

      title: "Dashboard",

      onTrigger: () => navigate("/"),

    },

    {

      title: "Client",

      onTrigger: () => navigate("/client"),

    },

    { title: "Group", onTrigger: () => navigate("/group") },

    { title: "Product", onTrigger: () => navigate("/products") },

    { title: "Proforma", onTrigger: () => navigate("/proforma") },

    { title: "Create Proforma", onTrigger: () => navigate("/create\_proforma") },

    { title: "Invoice", onTrigger: () => navigate("/create\_invoice") },

    { title: "Week Report", onTrigger: () => navigate("/week\_report") },

    { title: "Daily Report", onTrigger: () => navigate("/daily\_report") },

    { title: "Month report", onTrigger: () => navigate("/month\_report") },

    {

      title: "Delivery Challan",

      onTrigger: () => navigate("/delivery\_challan"),

    },

    { title: "Profile Setting", onTrigger: () => navigate("/profile") },

    { title: "Account Settings", onTrigger: () => navigate("/account") },

    { title: "HSN Tax", onTrigger: () => navigate("/tax") },

  ];

  return (

    <div>

      <AppShell

        styles={(theme) => ({

          main: {

            backgroundColor:

              theme.colorScheme === "dark"

                ? theme.colors.dark[8]

                : theme.colors.gray[0],

          },

        })}

        // navbarOffsetBreakpoint controls when navbar should no longer be offset with padding-left

        navbarOffsetBreakpoint="sm"

        // fixed prop on AppShell will be automatically added to Header and Navbar

        fixed

        header={

          <Header

            height={42}

            style={{

              borderBottom: "0px",

            }}

            className="border-bottom noprint"

          >

            <div

              style={{

                display: "flex",

                flexDirection: "column",

              }}

            >

              <div

                style={{

                  backgroundColor:

                    theme.colorScheme === "dark" ? "#2d2d2d" : "#EEEEEE",

                  display: "flex",

                  justifyContent: "space-between",

                  padding: 5,

                  paddingTop: 6,

                }}

              >

                <div

                  className="no-drag"

                  style={{ display: "flex", flexDirection: "row" }}

                >

                  <Navigation />

                </div>

                <div style={{ display: "flex", flexDirection: "row" }}>

                  <ActionIcon

                    className="zc-mt-1 no-drag"

                    color={dark ? "yellow" : "blue"}

                    onClick={() => toggleColorScheme()}

                    title="Toggle dark light"

                  >

                    {dark ? (

                      <Sun strokeWidth="0.8" size={20} />

                    ) : (

                      <MoonStars strokeWidth="0.8" size={20} />

                    )}

                  </ActionIcon>

                  {/\* <ActionIcon

                    className="zc-mt-1 no-drag"

                    color="dark"

                    onClick={() => setNotificationDrawer(!notificationDrawer)}

                    title="Page refresh"

                  >

                    <Indicator disabled={note} color="red">

                      <Bell strokeWidth="0.8" size={20} />

                    </Indicator>

                  </ActionIcon> \*/}

                  {/\* For Profile Dropdown list \*/}

                  <Menu shadow="md" className="no-drag zc-mr-1" width={200}>

                    <Menu.Target>

                      <ActionIcon

                        className="zc-mt-1"

                        color="dark"

                        type="button"

                        title="Setting"

                      >

                        <Settings strokeWidth="0.8" size={20} />

                      </ActionIcon>

                    </Menu.Target>

                    {/\* Profile setting page \*/}

                    <Menu.Dropdown>

                      {/\* <Menu.Item

                        icon={<UserExclamation size={14} />}

                        onClick={() => navigate("/profile")}

                      >

                        Profile Settings

                      </Menu.Item> \*/}

                      <Menu.Item

                        icon={<Settings size={14} />}

                        onClick={() => navigate("/manages")}

                      >

                        Settings

                      </Menu.Item>

                      {/\* For logout button \*/}

                      <Menu.Item

                        onClick={() => {

                          modals.openConfirmModal({

                            title: "Confirm Logout ",

                            children: (

                              <Text size="sm">Do you want to logout.</Text>

                            ),

                            labels: {

                              confirm: "Confirm",

                              cancel: "Cancel",

                            },

                            confirmProps: { color: "red" },

                            onCancel: () => console.log("Cancel"),

                            onConfirm: () => {

                              localStorage.clear();

                              navigate("/login");

                            },

                          });

                        }}

                        color="red"

                        icon={<Logout size={14} />}

                      >

                        Log Out

                      </Menu.Item>

                    </Menu.Dropdown>

                  </Menu>

                  <SpotlightProvider

                    className="no-drag"

                    actions={actions}

                    searchPlaceholder="Search..."

                    shortcut={["ctrl + k", "mod + K", "⌘ + K"]}

                    nothingFoundMessage="Nothing found..."

                  >

                    <div

                      style={{ width: 150 }}

                      className="no-drag"

                      onClick={openSpotlight}

                    >

                      <TextInput

                        variant="filled"

                        placeholder="Search"

                        onClick={openSpotlight}

                        size="xs"

                        pt={2}

                        pb={1}

                        shortcut={["ctrl + k", "mod + K", "⌘ + K"]}

                        icon={<IconSearch size={12} stroke={1.5} />}

                        rightSectionWidth={70}

                        rightSection={<Code>Ctrl + K</Code>}

                      />

                    </div>

                  </SpotlightProvider>

                </div>

              </div>

            </div>

          </Header>

        }

      >

        <Outlet />

      </AppShell>

      <div style={{ display: "none" }}>

        <Home schemeDrawer={notificationDrawer} />

      </div>

    </div>

  );

}

export default Sidebar;

**3.0 Login**

For admin login page

**3.0a Login form (auth🡪 Login.jsx)**

{/\* For login form inputs \*/}

        <form onSubmit={form.onSubmit((values) => LoginUser(values))}>

          <TextInput

            value={form.values.email}

            label="Email or Phone No"

            placeholder="Email or Phone No"

            size="md"

            {...form.getInputProps("email")}

          />

          <PasswordInput

            label="Password"

            placeholder="Your password"

            mt="md"

            size="md"

            value={form.values.password}

            {...form.getInputProps("password")}

          />

          <Button

            mt="xl"

            type="submit"

            fullWidth

            color="zevcore"

            loading={submitLoading}

          >

            Login

          </Button>

        </form>

// Submit login by email and password

  const LoginUser = async (e) => {

    setSubmitLoading(true);

    // submit login to api

    const response = await handelLogin(e);

    if (response.status == 200) {

      setTimeout(() => {

        // After login success store this data in local storage

        localStorage.setItem("notification", "yes");

        localStorage.setItem("token", response.data.accessToken);

        localStorage.setItem("isAuthenticated", "true");

        localStorage.setItem("role", response.data.roles);

        // window.location.reload();

        navigate("/");

      }, 300);

      showNotification({

        color: "green",

        title: "Login....",

        message: "You have login successfully.. 😉",

        icon: <Check />,

      });

      setSubmitLoading(false);

    } else {

      // Error show notification

      showNotification({

        color: "red",

        title: "Login Error",

        message: response.data.message,

        icon: <X />,

      });

      setSubmitLoading(false);

    }

  };

**3.0a Backend route (auth.routes.js)**

 app.post("/api/auth/signin", controller.signin); //for login user

**3.0b backend controller (auth.controller.js)**

//For signin users verifying and give new token

exports.signin = (req, res) => {

  //first find user with email id

  User.findOne({

    where: {

      email: req.body.email,

    },

  })

    .then((user) => {

      //if user not found with the email check with phone number

      if (!user) {

        User.findOne({

          where: {

            phone\_number: req.body.email,

          },

        }).then((user) => {

          //both email and phone number not found send user not found

          if (!user) {

            return res.status(404).send({ message: "User Not found." });

          } else {

            //if user get by phone number check password

            var passwordIsValid = bcrypt.compareSync(

              req.body.password,

              user.password

            );

            //if password not valid send invalid password

            if (!passwordIsValid) {

              return res.status(401).send({

                accessToken: null,

                message: "Invalid Password!",

              });

            }

            //if password verify give new token with secrete key

            var token = jwt.sign(

              { id: user.value, name: user.label },

              config.secret

            );

            //for send success and user details along with it token

            res.status(200).send({

              id: user.value,

              name: user.name,

              roles: user.role,

              accessToken: token,

            });

          }

        });

      } else {

        //if user get by phone number check password

        var passwordIsValid = bcrypt.compareSync(

          req.body.password,

          user.password

        );

        //if password not valid send invalid password

        if (!passwordIsValid) {

          return res.status(401).send({

            accessToken: null,

            message: "Invalid Password!",

          });

        }

        //if password verify give new token with secrete key

        var token = jwt.sign(

          { id: user.value, name: user.name, role: user.role },

          config.secret

        );

        //for send success and user details along with it token

        res.status(200).send({

          id: user.value,

          name: user.name,

          roles: user.role,

          accessToken: token,

        });

      }

    })

    .catch((err) => {

      res.status(500).send({ message: err.message });

    });

};

**4 Group**

**4.0 View group**

const response = await handleGetGroup();

        // On response setting the data to variable

        if (response.status === 200) {

          setVariables({

            ...variables,

            data: response.data.data,

            skeletonLoading: false,

          });

          const datas = dataSlice({

            data: response.data.data,

            page: 1,

            total: 10,

          });

          setSortedData(datas);

        }

 // Table data arrabnge by using function and loop throw each data rrange to table body

  const rows = sortedData.map((row, index) => (

    <tr key={row.label}>

      <td>{activePage \* total - total + index + 1}</td>

      <td>{row.label}</td>

      <td>{new Date(row.createdAt).toLocaleDateString()}</td>

      {/\* For action drop down edit and delete the data \*/}

      <td justifycontent="right" align="right">

        <Menu shadow="sm" size="xs">

          <Menu.Target>

            <ActionIcon

              color="zevcore"

              type="button"

              style={{ marginLeft: 10 }}

              size="xs"

            >

              <Dots />

            </ActionIcon>

          </Menu.Target>

          <Menu.Dropdown>

            <Menu.Item

              onClick={() => handleEdit(row.value)}

              icon={<Pencil size={14} />}

            >

              Edit

            </Menu.Item>

            <Menu.Item

              onClick={() => openConfirmModal(row.value)}

              icon={<Trash size={14} />}

            >

              Delete

            </Menu.Item>

          </Menu.Dropdown>

        </Menu>

      </td>

    </tr>

  ));

**4.0a Backend route (group.routes.js)**

 app.get("/api/group", [authJwt.verifyToken], controller.findAll); //get all the group data

**4.0b Backend controller(group.controller.js)**

//Find All group

exports.findAll = (req, res) => {

  try {

    //for get all users with the user decencies

    Group.findAll()

      .then((data) => {

        //send the success res to

        res.status(200).send({

          data: data,

        });

      })

      .catch((err) => {

        //send the error res to

        res.status(500).send({

          message: "Error retrieving all group",

          detail: err,

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**4.1 Add group**

 {/\* Group adding form name \*/}

          <form onSubmit={form.onSubmit((values) => AddGroup(values))}>

            <TextInput

              required

              variant="filled"

              value={form.values.label}

              label="Group Name"

              placeholder="Group Name"

              {...form.getInputProps("label")}

            />

            <Button

              mt="xl"

              type="submit"

              fullWidth

              color="zevcore"

              loading={variables.submitLoading}

            >

              Submit

            </Button>

          </form>

// Add new group data

  const AddGroup = async (e) => {

    setVariables({ ...variables, submitLoading: true });

    const response = await handleAddGroup(e);

    // Check for response for actions

    if (response.status === 200) {

      notificationHelper({

        color: "green",

        title: "Success",

        message: "Group added successfully",

      });

      form.reset();

      setVariables({

        ...variables,

        submitLoading: false,

        data: response.data.data,

      });

      const datas = dataSlice({

        data: response.data.data,

        page: activePage,

        total: total,

      });

      setSortedData(datas);

    } else {

      notificationHelper({

        color: "red",

        title: "Failed! Please enter correct details",

        message: response.data.message,

      });

      setVariables({ ...variables, submitLoading: false });

    }

  };

**4.1a Backend route**

  app.post("/api/group", [authJwt.verifyToken], controller.add); //create new group

**4.1b Backend controller**

// For create group

exports.add = async (req, res) => {

  try {

    //add the variable values to req data

    const detail = {

      label: req.body.label,

    };

    //create new group

    const datas = await Group.create(detail);

    //after create new group get all the group data with decencies

    const data = await Group.findAll();

    //for promise req get

    Promise.all(data)

      .then((data) => {

        res.send({ data });

      })

      .catch((err) => {

        res.status(500).send({

          message: "Error retrieving data ",

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**4.2 Edit group**

 {/\* Group Editing form name \*/}

          <form onSubmit={formEdit.onSubmit((values) => EditGroup(values))}>

            <TextInput

              required

              variant="filled"

              value={formEdit.values.label}

              label="Group Name"

              placeholder="Group Name"

              {...formEdit.getInputProps("label")}

            />

            <Button

              mt="xl"

              type="submit"

              fullWidth

              color="zevcore"

              loading={variables.submitLoading}

            >

              Submit

            </Button>

          </form>

// Edit group data compoennt

  const EditGroup = async (e) => {

    setVariables({ ...variables, submitLoading: true });

    const response = await handleEditGroup(e);

    // Check for respose data for actions

    if (response.status === 200) {

      notificationHelper({

        color: "green",

        title: "Success",

        message: "Group updated successfully",

      });

      setVariables({

        ...variables,

        submitLoading: false,

        data: response.data.data,

      });

      const datas = dataSlice({

        data: response.data.data,

        page: activePage,

        total: total,

      });

      setSortedData(datas);

    } else {

      notificationHelper({

        color: "red",

        title: "Failed! Please enter correct details",

        message: response.data.message,

      });

      setVariables({ ...variables, submitLoading: false });

    }

  };

**4.2a Backend route**

  app.patch("/api/group", [authJwt.verifyToken], controller.update); //update group data

**4.2b Backend controller**

// For update group

exports.update = async (req, res) => {

  try {

    //for store req data to variables

    const detail = {

      label: req.body.label,

    };

    //update the Group data with value id

    const datas = await Group.update(detail, {

      where: { value: req.body.value },

    });

    const data = await Group.findAll();

    //for promise req get

    Promise.all(data)

      .then((data) => {

        res.send({ data });

      })

      .catch((err) => {

        res.status(500).send({

          message: "Error retrieving data ",

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**4.3 Delete group**

 //For delete confirm modal show                                               Delete

  const openConfirmModal = (e) => {

    setVariables({ ...variables, deleteIndex: e });

    modals.openConfirmModal({

      title: "Do you want to delete this group value",

      labels: { confirm: "Confirm", cancel: "Cancel" },

      confirmProps: { color: "red" },

      onCancel: () => console.log("Cancel"),

      onConfirm: () => handleConfirmDelete(e),

    });

  };

  //   For delete db data from table and db

  const handleConfirmDelete = async (e) => {

    const response = await handleDeleteGroup(e);

    // Check the response for notification and actions

    if (response.status === 200) {

      notificationHelper({

        color: "green",

        title: "Success",

        message: "Group deleted successfully",

      });

      var filter = variables.data;

      filter = filter.filter((img) => img.value !== e);

      setVariables({

        ...variables,

        submitLoading: false,

        data: filter,

      });

      const datas = dataSlice({

        data: filter,

        page: activePage,

        total: total,

      });

      setSortedData(datas);

    } else {

      notificationHelper({

        color: "red",

        title: "Failed! Please enter correct details",

        message: response.data.message,

      });

      setVariables({ ...variables, submitLoading: false });

    }

  };

**4.3a Backend route**

 app.delete("/api/group/:value", [authJwt.verifyToken], controller.delete); //delete group

**4.3b Backend controller**

// For delete group

exports.delete = async (req, res) => {

  try {

    //group for delete the

    Group.destroy({

      where: { value: req.params.value },

    })

      .then((data) => {

        //send the success res to

        res.status(200).send({

          data: data,

        });

      })

      .catch((err) => {

        //send the error res to

        res.status(500).send({

          message: "Error deleting group",

          detail: err,

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**5 HSN (pages🡪settings🡪Tax.js)**

**Adding hsn for product tax percentage**

**5.0 View tax**

//   For get all the tax data list

        const response = await handleGetHSNTax();

        // On Respose setting the data to variable

        if (response.status === 200) {

          setVariables({

            ...variables,

            data: response.data.data,

            skeletonLoading: false,

          });

          const datas = dataSlice({

            data: response.data.data,

            page: 1,

            total: 10,

          });

          setSortedData(datas);

        }

// Table data arrabnge by using function and loop throw each data rrange to table body

  const rows = sortedData.map((row, index) => (

    <tr key={row.label}>

      <td>{activePage \* total - total + index + 1}</td>

      <td>{row.label}</td>

      <td>{row.tax}</td>

      <td>{new Date(row.createdAt).toLocaleDateString()}</td>

      {/\* For action drop down edit and delete the data \*/}

      <td justifycontent="right" align="right">

        <Menu shadow="sm" size="xs">

          <Menu.Target>

            <ActionIcon

              color="zevcore"

              type="button"

              style={{ marginLeft: 10 }}

              size="xs"

            >

              <Dots />

            </ActionIcon>

          </Menu.Target>

          <Menu.Dropdown>

            <Menu.Item

              onClick={() => handleEdit(row.value)}

              icon={<Pencil size={14} />}

            >

              Edit

            </Menu.Item>

            <Menu.Item

              onClick={() => openConfirmModal(row.value)}

              icon={<Trash size={14} />}

            >

              Delete

            </Menu.Item>

          </Menu.Dropdown>

        </Menu>

      </td>

    </tr>

  ));

**5.0a Backend route (hsn\_tax.routes.js)**

  app.get("/api/hsn\_tax", [authJwt.verifyToken], controller.findAll); //get all the hsn\_tax data

**5.0b Backend controller (hsn\_tax.controller.js)**

//Find All hsn

exports.findAll = (req, res) => {

  try {

    //for get all hsn

    HSNTax.findAll()

      .then((data) => {

        //send the success res to

        res.status(200).send({

          data: data,

        });

      })

      .catch((err) => {

        //send the error res to

        res.status(500).send({

          message: "Error retrieving all hsn",

          detail: err,

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**5.1 Add hsn\_tax**

  {/\* HSNTax adding form name \*/}

          <form onSubmit={form.onSubmit((values) => AddHSNTax(values))}>

            <TextInput

              variant="filled"

              value={form.values.label}

              label="HSN Name"

              placeholder="HSN Name"

              {...form.getInputProps("label")}

            />

            <NumberInput

              variant="filled"

              value={form.values.tax}

              label="Tax Value"

              placeholder="Tax Value"

              {...form.getInputProps("tax")}

            />

            <Button

              mt="xl"

              type="submit"

              fullWidth

              color="zevcore"

              loading={variables.submitLoading}

            >

              Submit

            </Button>

          </form>

  // Add new tax data

  const AddHSNTax = async (e) => {

    setVariables({ ...variables, submitLoading: true });

    const response = await handleAddHSNTax(e);

    // Check for response for actions

    if (response.status === 200) {

      notificationHelper({

        color: "green",

        title: "Success",

        message: "HSN Tax added successfully",

      });

      form.reset();

      setVariables({

        ...variables,

        submitLoading: false,

        data: response.data.data,

      });

      const datas = dataSlice({

        data: response.data.data,

        page: activePage,

        total: total,

      });

      setSortedData(datas);

    } else {

      notificationHelper({

        color: "red",

        title: "Failed! Please enter correct details",

        message: response.data.message,

      });

      setVariables({ ...variables, submitLoading: false });

    }

  };

**5.1a Backend route**

  app.post("/api/hsn\_tax", [authJwt.verifyToken], controller.add); //create new hsn\_tax

**5.1b Backend controller**

// For create hsn

exports.add = async (req, res) => {

  try {

    //add the variable values to req data

    const detail = {

      label: req.body.label,

      tax: req.body.tax,

    };

    //create new hsn

    const datas = await HSNTax.create(detail);

    //after create new hsn get all the hsn data with decencies

    const data = await HSNTax.findAll();

    //for promise req get

    Promise.all(data)

      .then((data) => {

        res.send({ data });

      })

      .catch((err) => {

        res.status(500).send({

          message: "Error retrieving data ",

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**5.2 Edit hsn\_tax**

 {/\* HSNTax adding form name \*/}

          <form onSubmit={formEdit.onSubmit((values) => EditHSNTax(values))}>

            <TextInput

              variant="filled"

              value={formEdit.values.label}

              label="HSN Name"

              placeholder="HSN Name"

              {...formEdit.getInputProps("label")}

            />

            <NumberInput

              variant="filled"

              value={formEdit.values.tax}

              label="Tax Value"

              placeholder="Tax Value"

              {...formEdit.getInputProps("tax")}

            />

            <Button

              mt="xl"

              type="submit"

              fullWidth

              color="zevcore"

              loading={variables.submitLoading}

            >

              Submit

            </Button>

          </form>

// Edit tax data compoennt

  const EditHSNTax = async (e) => {

    setVariables({ ...variables, submitLoading: true });

    const response = await handleEditHSNTax(e);

    // Check for respose data for actions

    if (response.status === 200) {

      notificationHelper({

        color: "green",

        title: "Success",

        message: "HSN Tax updated successfully",

      });

      setVariables({

        ...variables,

        submitLoading: false,

        data: response.data.data,

      });

      const datas = dataSlice({

        data: response.data.data,

        page: activePage,

        total: total,

      });

      setSortedData(datas);

    } else {

      notificationHelper({

        color: "red",

        title: "Failed! Please enter correct details",

        message: response.data.message,

      });

      setVariables({ ...variables, submitLoading: false });

    }

  };

**5.2a Backend route**

  app.patch("/api/hsn\_tax", [authJwt.verifyToken], controller.update); //update hsn\_tax data

**5.2b Backend controller**

// For update hsn

exports.update = async (req, res) => {

  try {

    //for store req data to variables

    const detail = {

      label: req.body.label,

      tax: req.body.tax,

    };

    //update the hsn data with value id

    const datas = await HSNTax.update(detail, {

      where: { value: req.body.value },

    });

    const data = await HSNTax.findAll();

    //for promise req get

    Promise.all(data)

      .then((data) => {

        res.send({ data });

      })

      .catch((err) => {

        res.status(500).send({

          message: "Error retrieving data ",

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**5.3 Delete hsn\_tax**

//For delete confirm modal show                                               Delete

  const openConfirmModal = (e) => {

    setVariables({ ...variables, deleteIndex: e });

    modals.openConfirmModal({

      title: "Do you want to delete this tax value",

      labels: { confirm: "Confirm", cancel: "Cancel" },

      confirmProps: { color: "red" },

      onCancel: () => console.log("Cancel"),

      onConfirm: () => handleConfirmDelete(e),

    });

  };

  //   For delete db data from table and db

  const handleConfirmDelete = async (e) => {

    const response = await handleDeleteHSNTax(e);

    // Check the response for notification and actions

    if (response.status === 200) {

      notificationHelper({

        color: "green",

        title: "Success",

        message: "HSN Tax deleted successfully",

      });

      var filter = variables.data;

      filter = filter.filter((img) => img.value !== e);

      setVariables({

        ...variables,

        submitLoading: false,

        data: filter,

      });

      const datas = dataSlice({

        data: filter,

        page: activePage,

        total: total,

      });

      setSortedData(datas);

    } else {

      notificationHelper({

        color: "red",

        title: "Failed! Please enter correct details",

        message: response.data.message,

      });

      setVariables({ ...variables, submitLoading: false });

    }

  };

**5.3a Backend route**

 app.delete("/api/hsn\_tax/:value", [authJwt.verifyToken], controller.delete); //delete hsn\_tax

**5.3b Backend controller**

// For delete hsn

exports.delete = async (req, res) => {

  try {

    //hsn for delete the

    HSNTax.destroy({

      where: { value: req.params.value },

    })

      .then((data) => {

        //send the success res to

        res.status(200).send({

          data: data,

        });

      })

      .catch((err) => {

        //send the error res to

        res.status(500).send({

          message: "Error deleting hsn",

          detail: err,

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**6 Product (pages🡪Product.js)**

**6.0 View Product**

const response = await handleGetProduct();

        // On Respose setting the data to variable

        if (response.status === 200) {

          setVariables((variables) => {

            return {

              ...variables,

              data: response.data.data,

              skeletonLoading: false,

            };

          });

          const datas = dataSlice({

            data: response.data.data,

            page: 1,

            total: 10,

          });

          setSortedData(datas);

        }

  // Table data arrabnge by using function and loop throw each data rrange to table body

  const rows = sortedData.map((row, index) => (

    <tr key={row.label}>

      <td>{activePage \* total - total + index + 1}</td>

      <td>{row.group.label}</td>

      <td>{row.label}</td>

      <td>{row.sku\_code}</td>

      <td>{row.hsn\_tax.label}</td>

      <td>{row.quantity}</td>

      <td>{row.price}</td>

      <td>{new Date(row.createdAt).toLocaleDateString()}</td>

      {/\* For action drop down edit and delete the data \*/}

      <td justifycontent="right" align="right">

        <Menu shadow="sm" size="xs">

          <Menu.Target>

            <ActionIcon

              color="zevcore"

              type="button"

              style={{ marginLeft: 10 }}

              size="xs"

            >

              <Dots />

            </ActionIcon>

          </Menu.Target>

          <Menu.Dropdown>

            <Menu.Item

              onClick={() => handleEdit(row.value)}

              icon={<Pencil size={14} />}

            >

              Edit

            </Menu.Item>

            <Menu.Item

              onClick={() => openConfirmModal(row.value)}

              icon={<Trash size={14} />}

            >

              Delete

            </Menu.Item>

          </Menu.Dropdown>

        </Menu>

      </td>

    </tr>

  ));

**6.0a Backend route (product.routes.js)**

 app.get("/api/product", [authJwt.verifyToken], controller.findAll); //get all the product data

**6.0b Backend controller (product.controller.js)**

//Find All product

exports.findAll = (req, res) => {

  try {

    //for get all users with the user decencies

    Product.findAll({

      include: [

        {

          model: Group,

          required: false,

        },

        {

          model: HSNTax,

          required: false,

        },

      ],

      order: [["value", "DESC"]],

    })

      .then((data) => {

        //send the success res to

        res.status(200).send({

          data: data,

        });

      })

      .catch((err) => {

        //send the error res to

        res.status(500).send({

          message: "Error retrieving all product",

          detail: err,

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**6.1 Add Product**

{/\* Product adding form name \*/}

              <form onSubmit={form.onSubmit((values) => AddProduct(values))}>

                <Select

                  variant="filled"

                  classNames={{ item: classes.selectItem }}

                  label="Select Group"

                  searchable

                  required

                  clearable

                  placeholder="Select Group"

                  {...form.getInputProps("group\_id")}

                  data={variables.groupList}

                />

                <TextInput

                  required

                  variant="filled"

                  value={form.values.label}

                  label="Product Name"

                  placeholder="Product Name"

                  {...form.getInputProps("label")}

                />

                <TextInput

                  variant="filled"

                  value={form.values.sku\_code}

                  label="Product SKU"

                  placeholder="Product SKU"

                  {...form.getInputProps("sku\_code")}

                />

                <Select

                  variant="filled"

                  classNames={{ item: classes.selectItem }}

                  label="Select HSN"

                  searchable

                  required

                  clearable

                  placeholder="Select HSN"

                  {...form.getInputProps("hsn\_id")}

                  data={variables.hsnList}

                />

                <NumberInput

                  variant="filled"

                  value={form.values.quantity}

                  label="Quantity"

                  placeholder="Quantity"

                  {...form.getInputProps("quantity")}

                />

                <NumberInput

                  variant="filled"

                  value={form.values.price}

                  label="Price"

                  placeholder="Price"

                  {...form.getInputProps("price")}

                />

                <TextInput

                  variant="filled"

                  value={form.values.lot\_number}

                  label="Product Lot Number"

                  placeholder="Product Lot Number"

                  {...form.getInputProps("lot\_number")}

                />

                <Button

                  mt="xl"

                  mb={60}

                  type="submit"

                  fullWidth

                  color="zevcore"

                  loading={variables.submitLoading}

                >

                  Submit

                </Button>

              </form>

 // Add new product data

  const AddProduct = async (e) => {

    setVariables({ ...variables, submitLoading: true });

    const response = await handleAddProduct(e);

    // Check for response for actions

    if (response.status === 200) {

      notificationHelper({

        color: "green",

        title: "Success",

        message: "Product added successfully",

      });

      form.reset();

      setVariables({

        ...variables,

        submitLoading: false,

        data: response.data.data,

      });

      const datas = dataSlice({

        data: response.data.data,

        page: activePage,

        total: total,

      });

      setSortedData(datas);

    } else {

      notificationHelper({

        color: "red",

        title: "Failed! Please enter correct details",

        message: response.data.message,

      });

      setVariables({ ...variables, submitLoading: false });

    }

  };

**6.1a Backend route**

 app.post("/api/product", [authJwt.verifyToken], controller.add); //create new product

**6.1b Backend controller**

// For create product

exports.add = async (req, res) => {

  try {

    //add the variable values to req data

    const detail = {

      label: req.body.label,

      sku\_code: req.body.sku\_code,

      hsn\_id: req.body.hsn\_id,

      group\_id: req.body.group\_id,

      quantity: req.body.quantity,

      price: req.body.price,

      lot\_number: req.body.lot\_number,

    };

    //create new product

    const datas = await Product.create(detail);

    //after create new product get all the product data with decencies

    const data = await Product.findAll({

      include: [

        {

          model: Group,

          required: false,

        },

        {

          model: HSNTax,

          required: false,

        },

      ],

      order: [["value", "DESC"]],

    });

    //for promise req get

    Promise.all(data)

      .then((data) => {

        res.send({ data });

      })

      .catch((err) => {

        res.status(500).send({

          message: "Error retrieving data ",

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**6.2 Edit product**

 {/\* Product Editing form name \*/}

              <form

                onSubmit={formEdit.onSubmit((values) => EditProduct(values))}

              >

                <Select

                  variant="filled"

                  classNames={{ item: classes.selectItem }}

                  label="Select Group"

                  searchable

                  required

                  clearable

                  placeholder="Select Group"

                  {...formEdit.getInputProps("group\_id")}

                  data={variables.groupList}

                />

                <TextInput

                  required

                  variant="filled"

                  value={formEdit.values.label}

                  label="Product Name"

                  placeholder="Product Name"

                  {...formEdit.getInputProps("label")}

                />

                <TextInput

                  variant="filled"

                  value={formEdit.values.sku\_code}

                  label="Product SKU"

                  placeholder="Product SKU"

                  {...formEdit.getInputProps("sku\_code")}

                />

                <Select

                  variant="filled"

                  classNames={{ item: classes.selectItem }}

                  label="Select HSN"

                  searchable

                  required

                  clearable

                  placeholder="Select HSN"

                  {...formEdit.getInputProps("hsn\_id")}

                  data={variables.hsnList}

                />

                <NumberInput

                  variant="filled"

                  value={formEdit.values.quantity}

                  label="Quantity"

                  placeholder="Quantity"

                  {...formEdit.getInputProps("quantity")}

                />

                <NumberInput

                  variant="filled"

                  value={formEdit.values.price}

                  label="Price"

                  placeholder="Price"

                  {...formEdit.getInputProps("price")}

                />

                <TextInput

                  variant="filled"

                  value={formEdit.values.lot\_number}

                  label="Product Lot Number"

                  placeholder="Product Lot Number"

                  {...formEdit.getInputProps("lot\_number")}

                />

                <Button

                  mt="xl"

                  mb={60}

                  type="submit"

                  fullWidth

                  color="zevcore"

                  loading={variables.submitLoading}

                >

                  Update

                </Button>

              </form>

  // Edit product data compoennt

  const EditProduct = async (e) => {

    setVariables({ ...variables, submitLoading: true });

    const response = await handleEditProduct(e);

    // Check for respose data for actions

    if (response.status === 200) {

      notificationHelper({

        color: "green",

        title: "Success",

        message: "Product updated successfully",

      });

      setVariables({

        ...variables,

        submitLoading: false,

        data: response.data.data,

      });

      const datas = dataSlice({

        data: response.data.data,

        page: activePage,

        total: total,

      });

      setSortedData(datas);

      setRefreshTable(new Date());

    } else {

      notificationHelper({

        color: "red",

        title: "Failed! Please enter correct details",

        message: response.data.message,

      });

      setVariables({ ...variables, submitLoading: false });

    }

  };

**6.3a Backend route**

  app.patch("/api/product", [authJwt.verifyToken], controller.update); //update product data

**6.3b Backend controller**

// For update product

exports.update = async (req, res) => {

  try {

    //for store req data to variables

    const detail = {

      label: req.body.label,

      sku\_code: req.body.sku\_code,

      hsn\_id: req.body.hsn\_id,

      group\_id: req.body.group\_id,

      quantity: req.body.quantity,

      price: req.body.price,

      lot\_number: req.body.lot\_number,

    };

    //update the Product data with value id

    const datas = await Product.update(detail, {

      where: { value: req.body.value },

    });

    const data = await Product.findAll({

      include: [

        {

          model: Group,

          required: false,

        },

        {

          model: HSNTax,

          required: false,

        },

      ],

      order: [["value", "DESC"]],

    });

    //for promise req get

    Promise.all(data)

      .then((data) => {

        res.send({ data });

      })

      .catch((err) => {

        res.status(500).send({

          message: "Error retrieving data ",

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**6.4 Delete product**

 //For delete confirm modal show                                               Delete

  const openConfirmModal = (e) => {

    setVariables({ ...variables, deleteIndex: e });

    modals.openConfirmModal({

      title: "Do you want to delete this product value",

      labels: { confirm: "Confirm", cancel: "Cancel" },

      confirmProps: { color: "red" },

      onCancel: () => console.log("Cancel"),

      onConfirm: () => handleConfirmDelete(e),

    });

  };

  //   For delete db data from table and db

  const handleConfirmDelete = async (e) => {

    const response = await handleDeleteProduct(e);

    // Check the response for notification and actions

    if (response.status === 200) {

      notificationHelper({

        color: "green",

        title: "Success",

        message: "Product deleted successfully",

      });

      var filter = variables.data;

      filter = filter.filter((img) => img.value !== e);

      setVariables({

        ...variables,

        submitLoading: false,

        data: filter,

      });

      const datas = dataSlice({

        data: filter,

        page: activePage,

        total: total,

      });

      setSortedData(datas);

    } else {

      notificationHelper({

        color: "red",

        title: "Failed! Please enter correct details",

        message: response.data.message,

      });

      setVariables({ ...variables, submitLoading: false });

    }

  };

**6.3a Backend route**

 app.delete("/api/product/:value", [authJwt.verifyToken], controller.delete); //delete product

**6.3b Backend controller**

// For delete product

exports.delete = async (req, res) => {

  try {

    //product for delete the

    Product.destroy({

      where: { value: req.params.value },

    })

      .then((data) => {

        //send the success res to

        res.status(200).send({

          data: data,

        });

      })

      .catch((err) => {

        //send the error res to

        res.status(500).send({

          message: "Error deleting product",

          detail: err,

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**7 Invoice add (pages🡪invoice🡪CreateInvoice.jsx)**

 {/\* For adding customer and product from this left side \*/}

      <Grid pb={0} gutter="xs" classNames="zc-hide-scroll">

        <Grid.Col xs={4}>

          <Card p={0} className="border">

            <div

              style={{

                display: "flex",

                padding: 6,

                margin: 0,

                justifyContent: "space-between",

                background: "#043c64",

                borderTopRightRadius: 3,

                borderTopLeftRadius: 3,

              }}

            >

              <Text color="#ffffff" weight={500} style={{ paddingTop: -15 }}>

                Create Invoice

              </Text>

            </div>

            <div style={{ padding: 15 }}>

              {/\* For add new customer or selct customer \*/}

              <Grid grow gutter="xs">

                <Grid.Col span={12}>

                  <div

                    style={{

                      display: "flex",

                      marginTop: 0,

                      justifyContent: "space-between",

                    }}

                  >

                    <Text style={{ fontSize: 14 }} weight={500}>

                      Select Customer

                    </Text>

                    <Button

                      color="zevcore"

                      onClick={() =>

                        setVariables({ ...variables, addDrawer: true })

                      }

                      style={{ fontSize: 10, padding: 2, height: 15 }}

                    >

                      Add <Plus size={10} />

                    </Button>

                  </div>

                  <Select

                    variant="filled"

                    fullWidth

                    size="xs"

                    required

                    dropdownPosition="bottom"

                    searchable

                    clearable

                    placeholder="Select Customer"

                    data={variables.customer}

                    value={customer}

                    onChange={(e) => {

                      setCustomer(e);

                    }}

                    mr={2}

                  />

                </Grid.Col>

                <Grid.Col span={6}>

                  <DatePicker

                    variant="filled"

                    size="xs"

                    value={variables.invoiceDate}

                    onChange={(e) => {

                      setVariables({

                        ...variables,

                        invoiceDate: e,

                      });

                    }}

                    label="Invoice Date"

                    placeholder="Pick date"

                    required

                  />

                </Grid.Col>

                <Grid.Col span={6}>

                  <TextInput

                    size="xs"

                    variant="filled"

                    value={form.values.reference}

                    label="Reference"

                    placeholder="Reference"

                    {...form.getInputProps("reference")}

                  />

                </Grid.Col>

                <Grid.Col span={6}>

                  <TextInput

                    size="xs"

                    variant="filled"

                    value={form.values.patient\_name}

                    label="Patient Name"

                    placeholder="Patient Name"

                    {...form.getInputProps("patient\_name")}

                  />

                </Grid.Col>

                <Grid.Col span={6}>

                  <TextInput

                    size="xs"

                    variant="filled"

                    value={form.values.ip\_num}

                    label="IP No."

                    placeholder="IP No."

                    {...form.getInputProps("ip\_num")}

                  />

                </Grid.Col>

                <Grid.Col span={6}>

                  <TextInput

                    size="xs"

                    variant="filled"

                    value={form.values.dl\_num}

                    label="DL No."

                    placeholder="DL No."

                    {...form.getInputProps("dl\_num")}

                  />

                </Grid.Col>

                <Grid.Col span={6}>

                  <TextInput

                    size="xs"

                    variant="filled"

                    value={form.values.dc\_num}

                    label="DC No."

                    placeholder="DC No."

                    {...form.getInputProps("dc\_num")}

                  />

                </Grid.Col>

                <Grid.Col span={6}>

                  <TextInput

                    size="xs"

                    variant="filled"

                    value={form.values.po\_number}

                    label="PO Number"

                    placeholder="PO Number"

                    {...form.getInputProps("po\_number")}

                  />

                </Grid.Col>

              </Grid>

              <Divider

                pt={2}

                mb={0}

                label="Product Select"

                labelPosition="center"

              />

              {/\* Adding product for cart \*/}

              <form

                ref={formRef}

                onKeyPress={(e) => {

                  if (e.key === "Enter") {

                    e.preventDefault();

                  }

                }}

                onSubmit={formBarcode.onSubmit((values) => AddProduct(values))}

              >

                <Grid grow gutter="xs">

                  <Grid.Col span={6}>

                    <Select

                      size="xs"

                      variant="filled"

                      label="Product Type"

                      placeholder="Product one"

                      data={variables.groupList}

                      value={formBarcode.values.product\_type}

                      {...formBarcode.getInputProps("product\_type")}

                    />

                  </Grid.Col>

                  <Grid.Col span={6}>

                    {formBarcode.values.product\_type == "Implant" ? (

                      <>

                        <div

                          style={{

                            display: "flex",

                            marginTop: 0,

                            marginTop: 6,

                            justifyContent: "space-between",

                          }}

                        >

                          <Text style={{ fontSize: 12 }} weight={500}>

                            Select Product

                          </Text>

                          {/\* Button for adding new product \*/}

                          <Button

                            color="zevcore"

                            onClick={() =>

                              setVariables({ ...variables, addDrawer2: true })

                            }

                            style={{ fontSize: 10, padding: 2, height: 15 }}

                          >

                            Add <Plus size={10} />

                          </Button>

                        </div>

                        <Select

                          ref={input3}

                          size="xs"

                          variant="filled"

                          fullWidth

                          placeholder="Select Product"

                          data={variables.productList.filter(

                            (raw) =>

                              raw.group == formBarcode.values.product\_type

                          )}

                          searchable

                          maxDropdownHeight={400}

                          nothingFound="Nobody here"

                          value={formBarcode.values.product\_id}

                          onChange={(e) => {

                            var pp = variables.productList.find(

                              (img) => Number(img.value) === Number(e)

                            );

                            formBarcode.setFieldValue(

                              "price",

                              Number(pp.price)

                            );

                            formBarcode.setFieldValue("product", pp.label);

                            formBarcode.setFieldValue("product\_id", e);

                          }}

                        />

                      </>

                    ) : (

                      <>

                        <div

                          style={{

                            display: "flex",

                            marginTop: 0,

                            marginTop: 6,

                            justifyContent: "space-between",

                          }}

                        >

                          <Text style={{ fontSize: 12 }} weight={500}>

                            Select Product

                          </Text>

                          <Button

                            color="zevcore"

                            onClick={() =>

                              setVariables({ ...variables, addDrawer2: true })

                            }

                            style={{ fontSize: 10, padding: 2, height: 15 }}

                          >

                            Add <Plus size={10} />

                          </Button>

                        </div>

                        <Select

                          ref={input3}

                          size="xs"

                          variant="filled"

                          fullWidth

                          placeholder="Select Product"

                          data={variables.productList.filter(

                            (raw) =>

                              raw.group == formBarcode.values.product\_type

                          )}

                          searchable

                          maxDropdownHeight={400}

                          nothingFound="Nobody here"

                          value={formBarcode.values.product\_id}

                          onChange={(e) => {

                            var pp = variables.productList.find(

                              (img) => Number(img.value) === Number(e)

                            );

                            formBarcode.setFieldValue(

                              "price",

                              Number(pp.price)

                            );

                            formBarcode.setFieldValue("product", pp.label);

                            formBarcode.setFieldValue("product\_id", e);

                          }}

                        />

                      </>

                    )}

                  </Grid.Col>

                  <Grid.Col span={6}>

                    <NumberInput

                      ref={input5}

                      size="xs"

                      variant="filled"

                      value={formBarcode.values.lot\_number}

                      label="Product Lot Number"

                      placeholder="Product Lot Number"

                      {...formBarcode.getInputProps("lot\_number")}

                    />

                  </Grid.Col>

                  <Grid.Col span={6}>

                    <NumberInput

                      ref={input5}

                      size="xs"

                      variant="filled"

                      value={formBarcode.values.qty}

                      label="Product Quantity"

                      placeholder="Product Quantity"

                      {...formBarcode.getInputProps("qty")}

                    />

                  </Grid.Col>

                  <Grid.Col span={6}>

                    <NumberInput

                      ref={input4}

                      size="xs"

                      variant="filled"

                      value={formBarcode.values.price}

                      label="Product Price"

                      placeholder="Product Price"

                      {...formBarcode.getInputProps("price")}

                    />

                  </Grid.Col>

                  <Grid.Col span={6}>

                    <NumberInput

                      ref={input4}

                      size="xs"

                      variant="filled"

                      value={

                        (formBarcode.values.price -

                          formBarcode.values.discount) \*

                        formBarcode.values.qty

                      }

                      label="Total Price"

                      placeholder="Total Price"

                    />

                  </Grid.Col>

                  <Grid.Col span={6}>

                    <NumberInput

                      ref={input7}

                      size="xs"

                      variant="filled"

                      label="Enter Discount Value"

                      value={formBarcode.values.discount}

                      {...formBarcode.getInputProps("discount")}

                      placeholder="Enter whole number"

                    ></NumberInput>

                  </Grid.Col>

                </Grid>

                <div style={{ display: "flex", marginTop: 15 }}>

                  <Button

                    size="xs"

                    fullWidth

                    type="submit"

                    onClick={() => setDetailModal(true)}

                    color="zevcore"

                  >

                    Challan Detail

                  </Button>

                  <Button

                    ml={5}

                    size="xs"

                    fullWidth

                    type="submit"

                    color="zevcore"

                  >

                    Submit

                  </Button>

                </div>

              </form>

            </div>

          </Card>

        </Grid.Col>

        {/\* For show the product list and delete product and add invoice here \*/}

        <Grid.Col xs={8}>

          <Card className="border" pt={2}>

            <div style={{ paddingBottom: 10 }}>

              <div className="invoiceItemHeight">

                {/\* Display all the products list here \*/}

                <table className="tableBorder ">

                  <thead>

                    <tr>

                      <th>Sl.No</th>

                      <th>Description / Item</th>

                      <th>Qty</th>

                      <th>Price</th>

                      <th>Discount</th>

                      <th>Total</th>

                      <th>Lot Number</th>

                      <th>Delete</th>

                    </tr>

                  </thead>

                  <tbody key={tableRefresh}>

                    {cart.length > 0 ? (

                      <>

                        {cart.map((row, index) => (

                          <tr className="products">

                            <td>{index + 1}</td>

                            <td>{row.product.label}</td>

                            <td>{row.qty}</td>

                            <td>{row.price}</td>

                            <td>{row.discount}</td>

                            <td>{row.total}</td>

                            <td>{row.lot\_number}</td>

                            <td

                              onClick={() => DeleteProduct(row.product.value)}

                            >

                              <Button

                                color="red"

                                size="xs"

                                style={{ fontSize: 8, padding: 2, height: 15 }}

                              >

                                Delete

                              </Button>

                            </td>

                          </tr>

                        ))}

                      </>

                    ) : null}

                  </tbody>

                </table>

              </div>

            </div>

            <Divider my="xs" />

            <Grid gutter="xs">

              <Grid.Col span={8}>

                <Grid>

                  <Grid.Col span={4} pb={0} pt={1}>

                    {" "}

                    <TextInput

                      size="xs"

                      variant="filled"

                      value={variables.amountPaid}

                      label="Amount Paid"

                      onChange={(e) =>

                        setVariables({

                          ...variables,

                          amountPaid: e.target.value,

                        })

                      }

                      placeholder="Amount Paid"

                    />

                  </Grid.Col>

                  <Grid.Col span={4} pb={0} pt={1}>

                    <TextInput

                      size="xs"

                      ml={5}

                      variant="filled"

                      readOnly

                      value={

                        cart.reduce(function (sum, current) {

                          return +Number(sum) + Number(current.total);

                        }, 0) +

                        cart.reduce(function (sum, current) {

                          return +Number(sum) + Number(current.tax);

                        }, 0) -

                        Number(variables.amountPaid)

                      }

                      label="Balance"

                      placeholder="Balance"

                    />

                  </Grid.Col>

                </Grid>

                <div

                  style={{

                    marginTop: 32,

                    display: "flex",

                    justifyContent: "space-between",

                  }}

                >

                  <div>

                    <Button

                      disabled={cart.length == 0 ? true : false}

                      color="zevcore"

                      size="xs"

                      onClick={() => AddSaveData()}

                    >

                      Save Invoice

                    </Button>

                    <Button

                      ml={5}

                      color="zevcore"

                      size="xs"

                      onClick={() =>

                        setVariables({ ...variables, saveModal: true })

                      }

                    >

                      View Saved

                    </Button>

                  </div>

                  <Button

                    color="zevcore"

                    disabled={cart.length == 0 ? true : false}

                    size="xs"

                    onClick={() => AddSaleData()}

                  >

                    Add Invoice

                  </Button>

                </div>

              </Grid.Col>

              <Grid.Col span={4} mt={4}>

                <table className="tableBorder">

                  <tbody>

                    <tr>

                      <td>Sub-Total</td>

                      <td>

                        ₹

                        {cart.reduce(function (sum, current) {

                          return +Number(sum) + Number(current.total);

                        }, 0)}

                      </td>

                    </tr>

                    <tr>

                      <td>CGST</td>

                      <td>

                        ₹

                        {cart

                          .reduce(function (sum, current) {

                            return +Number(sum) + Number(current.tax);

                          }, 0)

                          .toFixed(2) / 2}

                      </td>

                    </tr>

                    <tr>

                      <td>SGST</td>

                      <td>

                        ₹

                        {cart

                          .reduce(function (sum, current) {

                            return +Number(sum) + Number(current.tax);

                          }, 0)

                          .toFixed(2) / 2}

                      </td>

                    </tr>

                    <tr

                      style={{

                        backgroundColor: "#043c64",

                        fontWeight: 500,

                        color: "#ffffff",

                      }}

                    >

                      <td>Grand Total</td>

                      <td>

                        {" "}

                        ₹

                        {Math.round(

                          cart.reduce(function (sum, current) {

                            return +Number(sum) + Number(current.total);

                          }, 0) +

                            cart.reduce(function (sum, current) {

                              return +Number(sum) + Number(current.tax);

                            }, 0)

                        )}

                      </td>

                    </tr>

                  </tbody>

                </table>

              </Grid.Col>

            </Grid>

          </Card>

        </Grid.Col>

      </Grid>

// Add the new invoice with data

  const AddSaleData = async () => {

    // Once the invoice add check for date validation for

    var first = new Date(lastInvoice).setHours(0, 0, 0, 0);

    var second = new Date(variables.invoiceDate).setHours(0, 0, 0, 0);

    var third = new Date().setHours(0, 0, 0, 0);

    var check = 0;

    if (lastInvoice != "") {

      if (first <= second && second <= third) {

        check = 0;

      } else {

        check = 1;

      }

    }

    if (check == 0) {

      var customerId = null;

      var customerGST = null;

      var customerIID = null;

      if (customer !== null && customer !== "") {

        var data = variables.customer.find((img) => img.value === customer);

        customerId = data.label;

        customerGST = data.gstin;

        customerIID = customer;

      }

      // Adding the invoice data list

      const reg = {

        discount\_amount: cart

          .reduce(function (sum, current) {

            return +Number(sum) + Number(current.discount);

          }, 0)

          .toFixed(2),

        sub\_totalamt: cart.reduce(function (sum, current) {

          return +Number(sum) + Number(current.total);

        }, 0),

        totalTax: Number(

          cart.reduce(function (sum, current) {

            return +Number(sum) + Number(current.tax);

          }, 0)

        ).toFixed(2),

        sale\_type: saleTypeValue,

        customer: customerId,

        customer\_id: customerIID,

        customerGST: customerGST,

        reference: form.values.reference,

        po\_number: form.values.po\_number,

        patient\_name: form.values.patient\_name,

        ip\_num: form.values.ip\_num,

        dl\_num: form.values.dl\_num,

        dc\_num: form.values.dc\_num,

        date: form.values.invoiceDate,

        total\_amount:

          cart.reduce(function (sum, current) {

            return +Number(sum) + Number(current.total);

          }, 0) +

          cart.reduce(function (sum, current) {

            return +Number(sum) + Number(current.tax);

          }, 0),

        list: cart,

        due\_amount:

          cart.reduce(function (sum, current) {

            return +Number(sum) + Number(current.total);

          }, 0) +

          cart.reduce(function (sum, current) {

            return +Number(sum) + Number(current.tax);

          }, 0) -

          Number(variables.amountPaid),

        paid: variables.amountPaid,

        date: variables.invoiceDate,

        challan: formDetails.values.label,

        doctor\_name: formDetails.values.doctor\_name,

      };

      // handle the invoice add to database

      const response = await handleAddCreateInvoice(reg);

      if (response.status === 200) {

        navigate("/invoice\_print/" + response.data[0].value);

        notificationHelper({

          color: "green",

          title: "Success",

          message: "Invoice added successfully",

        });

      } else {

        notificationHelper({

          color: "red",

          title: "Failed! Please enter correct details",

          message: response.data.message,

        });

      }

    } else {

      alert("Please select date correctly");

    }

  };

**7.0a Backend route**

 app.post("/api/tax\_invoice", [authJwt.verifyToken], controller.add); //create new tax\_invoice

**7.0b Backend controller**

// For create tax\_invoice

exports.add = async (req, res) => {

  // console.log(req.body);

  try {

    var check = 1;

    // For check the client credit limit

    if (

      req.body.customer\_id != "" &&

      req.body.customer\_id != null &&

      typeof req.body.customer\_id != "undefined"

    ) {

      const client = await Client.findByPk(req.body.customer\_id);

      if (Number(client.credit\_limit) > 0) {

        if (Number(client.credit\_limit) >= Number(req.body.due\_amount)) {

          check = 0;

        }

      }

    }

    if (check == 1) {

      const last = await TaxInvoice.findAll({

        limit: 1,

        order: [["invoice\_date", "DESC"]],

      });

      var invoice = "";

      //check for last invoice null

      if (last.length > 0) {

        invoice = Number(last[0].invoice\_num) + 1; //add id with last invoice

      } else {

        invoice = 2051; //or just add 1

      }

      //add the variable values to req data

      const detail = {

        invoice\_num: invoice,

        invoice\_date: req.body.date,

        client\_name: req.body.customer,

        client\_gst: req.body.customerGST,

        client\_id: req.body.customer\_id,

        reference: req.body.reference,

        patient\_name: req.body.patient\_name,

        ip\_num: req.body.ip\_num,

        dl\_num: req.body.dl\_num,

        po\_number: req.body.po\_number,

        dc\_num: req.body.dc\_num,

        total\_amount: req.body.dc\_num,

        total\_amount: req.body.total\_amount,

        due\_amount: req.body.due\_amount,

        discount\_amount: req.body.discount\_amount,

        sub\_totalamt: req.body.sub\_totalamt,

        gst\_amount: req.body.totalTax,

      };

      //create new invoice data

      const invoice\_create = await TaxInvoice.create(detail);

      var list = req.body.list;

      for (i = 0; i < list.length; i++) {

        const details2 = {

          invoice\_id: invoice\_create.value,

          invoice\_date: req.body.date,

          product\_name: list[i].product.label,

          hsn: list[i].product.hsn\_code,

          sku: list[i].product.sku\_code,

          mrp\_display: list[i].product.price,

          quantity: list[i].qty,

          lot\_number: list[i].lot\_number,

          price: list[i].price,

          total\_amount: list[i].total,

          type: list[i].type,

        };

        await TaxInvoiceTransaction.create(details2);

      }

      // For update credit limit

      // const cls = {

      //   credit\_limit: Number(client.credit\_limit) - Number(req.body.due\_amount),

      // };

      // const clients = await Client.update(cls, {

      //   where: { value: req.body.customer\_id },

      // });

      if (

        req.body.challan != "" &&

        req.body.challan != null &&

        typeof req.body.challan != "undefined"

      ) {

        const detail456 = {

          label: req.body.challan,

          doctor\_name: req.body.doctor\_name,

          invoice\_id: invoice\_create.value,

          client\_id: req.body.customer\_id,

        };

        //create new Challan

        const datas789 = await DeliveryChallan.create(detail456);

      }

      //after create new tax\_invoice get all the tax\_invoice data with decencies

      const data = await TaxInvoice.findByPk(invoice\_create.value);

      //for promise req get

      Promise.all([data])

        .then((data) => {

          res.send(data);

        })

        .catch((err) => {

          res.status(500).send({

            message: "Error retrieving data ",

          });

        });

    } else {

      res.status(500).send({

        message: "Customer credit limit exceeded ",

      });

    }

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**7.1 View invoice in printing section**

  // Start printing the page

    <div style={{ padding: 10 }}>

      <div

        style={{

          textAlign: "right",

        }}

        className="noprint"

      >

        {/\* for get the back screen \*/}

        <Button

          type="submit"

          color="zevcore"

          variant="outline"

          mt={15}

          onClick={() => navigate(-1)}

        >

          Back

        </Button>

      </div>

      <>

        {data != "" ? (

          <div className="invoiceContainer">

            <table className="invoiceTable">

              <tbody>

                <tr>

                  <td className="tableText tableHeader" colspan="12">

                    <Text weight={600} size="md">

                      TAX Invoice

                    </Text>

                  </td>

                </tr>

              </tbody>

              <tbody>

                {/\* Company details \*/}

                <tr>

                  <td colspan="5" rowspan="3">

                    <Text

                      color="#b90707"

                      weight={600}

                      size="xl"

                      component="span"

                    >

                      Dhanashree Surgicals

                    </Text>

                    <br />

                    <Text weight={500}>

                      #185 B1, 1st Stage Bisilumaramma Temple Road,

                    </Text>

                    <Text weight={500}> Gangothri Layout, Mysore-570009 </Text>

                    <Text weight={500}>

                      Phone: 9448386150, 9972450797, 9886655586

                    </Text>

                  </td>

                </tr>

                <tr>

                  {/\* Invoice details \*/}

                  <td colspan="7">

                    {" "}

                    <Text weight={600}>

                      Invoice No: {data.invoice\_num} &nbsp;

                    </Text>{" "}

                  </td>

                </tr>

                <tr>

                  <td colspan="7">

                    <Text>

                      {" "}

                      <Text weight={600} component="span">

                        {" "}

                        Date:

                      </Text>{" "}

                      {new Date(data.invoice\_date).toLocaleDateString("en-UK")}

                    </Text>

                  </td>

                </tr>

                {/\* Customer details name and gst \*/}

                <tr>

                  <td colspan="5" rowspan="4">

                    <Text size="md" weight={600}>

                      To,{data.client\_name}&nbsp;

                    </Text>

                    <Text>

                      {" "}

                      <Text weight={600} component="span">

                        {" "}

                        Ref:

                      </Text>{" "}

                      {data.reference}

                    </Text>

                    <Text>

                      {" "}

                      <Text weight={600} component="span">

                        {" "}

                        GST:

                      </Text>{" "}

                      {data.client\_gst}

                    </Text>

                    <Text>

                      {" "}

                      <Text weight={600} component="span">

                        {" "}

                        DL No:

                      </Text>{" "}

                      {data.dl\_num}

                    </Text>

                  </td>

                </tr>

                <tr>

                  <td colspan="7">

                    {" "}

                    <Text>

                      {" "}

                      <Text weight={600} component="span">

                        {" "}

                        DC No:

                      </Text>{" "}

                      {data.dc\_num}

                    </Text>

                  </td>

                </tr>

                <tr>

                  <td colspan="7">

                    <Text>

                      {" "}

                      <Text weight={600} component="span">

                        {" "}

                        DL No

                      </Text>{" "}

                      : KA/MYSU/1/20B-709 <br />{" "}

                      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:

                      KA/MYSU/1/21B-666

                    </Text>

                  </td>

                </tr>

                <tr>

                  <td colspan="7">

                    <Text>

                      {" "}

                      <Text weight={600} component="span">

                        {" "}

                        GST:

                      </Text>{" "}

                      29BBEPS6455H1Z5

                    </Text>

                  </td>

                </tr>

                {/\* Patient Details \*/}

                <tr>

                  <td colspan="5" rowspan="2">

                    <Text weight={600}>Patient: {data.patient\_name}</Text>

                  </td>

                </tr>

                <tr>

                  <td colspan="7">

                    {" "}

                    <Text weight={600}>IP No : {data.ip\_num}</Text>

                    <Text weight={600}>PO Number : {data.po\_number}</Text>

                  </td>

                </tr>

              </tbody>

              {/\* List of invoice products with hsn and lot\_number \*/}

              <tbody>

                <tr>

                  <td className="tableHeader">Sl.No.</td>

                  <td className="tableHeader2" colspan="4">

                    Particulars

                  </td>

                  <td className="tableHeader3">Lot&nbsp;No</td>

                  <td className="tableHeader3">HSN</td>

                  <td className="tableHeader">MRP</td>

                  <td className="tableHeader" colspan="2">

                    Rate

                  </td>

                  <td className="tableHeader">Qty&nbsp;</td>

                  <td className="tableHeader4" colspan="2">

                    Amount

                  </td>

                </tr>

              </tbody>

              <tbody className="invoiceProducts">

                {data.tax\_invoice\_transactions.map((row, index) => (

                  <tr className="tableBottomBorderNo">

                    <td>{index + 1}</td>

                    <td colspan="4">

                      <span className="tableSpan"> {row.product\_name}</span>

                    </td>

                    <td>

                      <span className="tableSpan">{row.lot\_number}</span>

                    </td>

                    <td>

                      <span className="tableSpan">{row.hsn}</span>

                    </td>

                    <td>

                      <span className="tableSpan">{row.mrp\_display}</span>

                    </td>

                    <td colspan="2">

                      <span className="tableSpan">{row.price}</span>

                    </td>

                    <td>

                      <span className="tableSpan">{row.quantity}</span>

                    </td>

                    <td colspan="2">

                      <span className="tableSpan">{row.total\_amount}</span>

                    </td>

                  </tr>

                ))}

              </tbody>

              {/\* show the total of quantity and total with discount \*/}

              <tbody>

                <tr className="tableBottomBorderNo tableTopBorder">

                  <td></td>

                  <td colspan="4" className="tableTextRight tableTextBold">

                    Total Quantiy | Sub-Total

                  </td>

                  <td></td>

                  <td></td> <td></td>

                  <td colspan="2"></td>

                  <td className="tableTextBold">

                    {data.tax\_invoice\_transactions.length}

                  </td>

                  <td className="tableTextBold" colspan="2">

                    {data.sub\_totalamt}

                  </td>

                </tr>

                <tr className="tableBottomBorderNo">

                  <td></td>

                  <td colspan="4" className="tableTextRight tableTextBold">

                    CGST

                  </td>

                  <td></td>

                  <td></td> <td></td>

                  <td colspan="2"></td>

                  <td></td>

                  <td className="tableTextBold" colspan="2">

                    {Number(Number(data.gst\_amount) / 2).toFixed(2)}

                  </td>

                </tr>

                <tr className="tableBottomBorderNo">

                  <td></td>

                  <td className="tableTextRight tableTextBold" colspan="4">

                    SCGST

                  </td>

                  <td></td>

                  <td></td> <td></td>

                  <td colspan="2"></td>

                  <td></td>

                  <td className="tableTextBold" colspan="2">

                    {Number(Number(data.gst\_amount) / 2).toFixed(2)}

                  </td>

                </tr>

                <tr>

                  <td></td>

                  <td className="tableTextRight tableTextBold" colspan="4">

                    R-off/Discount

                  </td>

                  <td></td>

                  <td></td> <td></td>

                  <td colspan="2"></td>

                  <td></td>

                  <td className="tableTextBold" colspan="2">

                    (-){data.discount\_amount}

                  </td>

                </tr>

                <tr>

                  <td></td>

                  <td className="tableTextRight tableTextBold" colspan="4">

                    Grand Total

                  </td>

                  <td></td>

                  <td></td>

                  <td></td> <td></td>

                  <td colspan="2"></td>

                  <td colspan="2">

                    <Text weight={600} size="sm">

                      {data.total\_amount}

                    </Text>

                  </td>

                </tr>

              </tbody>

              <tbody>

                <tr>

                  <td colspan="12">

                    Amount (in words): <br />{" "}

                    <span className="tableSpan">

                      {price\_in\_words2(data.total\_amount)}

                    </span>

                  </td>

                </tr>

              </tbody>

**7.1a Backend route**

  app.get("/api/tax\_invoice/:id", [authJwt.verifyToken], controller.findById); //get all the tax\_invoice data

**7.1b Controller**

// Find one tax\_invoice by id

exports.findById = (req, res) => {

  try {

    const id = req.params.id;

    TaxInvoice.findByPk(id, {

      order: [["invoice\_date", "DESC"]],

      include: [

        {

          model: TaxInvoiceTransaction,

          required: false,

        },

      ],

    })

      .then((data) => {

        //For send the success to res

        res.send(data);

      })

      .catch((err) => {

        //for send the error to res

        res.status(500).send({

          message: "Error retrieving tax\_invoice with id=" + id + "",

          detail: err,

        });

      });

  } catch (error) {

    res.status(500).send({

      message: "Failed to perform request",

      detail: error,

    });

  }

};

**8 Proforma invoice**

Refer 7.0 for invoice same as proforma but tax data will be hide