App.css

\*{  
 margin: 0;  
 padding: 0;  
}  
  
body{  
 font-family: "Inter", sans-serif;  
 font-optical-sizing: auto;  
 font-style: normal;  
 overflow-x: hidden;  
 min-height: 100svh;  
}  
  
::-webkit-scrollbar-track{  
 background-color: var(--background-secondary);  
}  
  
::-webkit-scrollbar{  
 width: 12px;  
}  
  
::-webkit-scrollbar-thumb{  
 background-color: var(--secondary);  
}  
  
/\* Color Variables \*/  
body{  
 --inputText: #252525;  
 --text-primary: #3b3b3b;  
 --secondary: #676767;  
 --border: #262626;  
 --accent-color: #417BD1;  
 --background-primary: #ffffff;  
 --background-secondary: #ececec;  
 --border-design: 1.3px solid #262626;  
 --border-sahdow: 0px 4px 4px #26262675;  
}  
  
.Container{  
 background-color: var(--background-secondary);  
 min-height: 100svh;  
}  
  
.inactiveBg{  
 pointer-events: none;  
 filter: blur(2px);  
}  
  
.modalOverlay {  
 position: fixed;  
 top: 0;  
 left: 0;  
 width: 100%;  
 height: 100%;  
 background-color: rgba(0, 0, 0, 0.1);  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 z-index: 1000;  
}  
  
.modal {  
 background-color:var(--background-primary);  
 border-radius: 4px;  
 box-shadow: var(--border-sahdow);  
 z-index: 1100;  
 padding: 20px;  
 gap: 12px;  
}  
  
.fullModal{  
 background-color:var(--background-primary);  
 z-index: 1100;  
 gap: 12px;  
 height: 100svh;  
}  
  
.modalTitle{  
 background-color: var(--accent-color);  
 font-size: 18px;  
 color: var(--background-primary);  
 font-weight: bold;  
 text-transform: uppercase;  
 padding: 24px 60px;  
}  
  
.btnBox{  
 gap: 12px;  
 justify-content: right;  
 margin: 12px 0;  
}  
  
.flex{  
 display: flex;  
}  
  
.flex-row{  
 flex-direction: row;  
}  
  
.flex-column{  
 flex-direction: column;  
}  
  
.center{  
 justify-content: center;  
 align-self: center;  
}  
  
.spaceAround{  
 justify-content: space-around;  
}  
  
.spaceBetween{  
 justify-content: space-between;  
}  
  
.centerBoth{  
 justify-content: center;  
 align-items: center;  
}  
  
.talign\_right{  
 text-align: right;  
}  
  
.alignItems\_Center{  
 align-items: center;  
}  
  
.imageSection{  
 background: url(./Assets/LoginBg.svg);  
 background-size: contain;  
 background-repeat: no-repeat;  
 background-position: center center;  
 flex-basis: 67%;  
}  
  
.formSection{  
 background-color: white;  
 flex-grow: 1;  
 padding: 12px 36px;  
}  
  
  
#logoImg{  
 height: 80px;  
 width: fit-content;  
 margin: 8px 0;  
}  
  
.titleText{  
 font-size: 28px;  
 color: var(--text-primary);  
 font-weight: 600;  
}  
  
.subText{  
 font-size: 14px;  
 color: var(--secondary);  
}  
  
input{  
 padding: 16px;  
 font-size: 14px;  
 border-radius: 6px;  
 border: none;  
 outline: none;  
 border: var(--border-design);  
 color: var(--inputText);  
 font-weight: 300;  
}  
  
input::placeholder, textarea::placeholder{  
 font-weight: 300;  
}  
  
input:focus, textarea:focus{  
 border:2px solid var(--accent-color);  
}  
  
button{  
 border: none;  
 outline: none;  
 cursor: pointer  
}  
  
#formBox{  
 gap: 16px;  
}  
  
.blue{  
 background-color: var(--accent-color);  
}  
  
.black{  
 background-color: var(--text-primary);  
}  
  
.red{  
 background-color: #A20202;  
}  
.green{  
 background-color: #349A38;  
}  
.orange{  
 background-color: #FA6900;  
}  
.darkBlue{  
 background-color: rgb(0, 0, 63);  
}  
.col-blue{  
 color: var(--accent-color);  
}  
.col-white{  
 color: var(--background-primary);  
}  
.circle{  
 border-radius: 50%;  
 padding: 6px;  
}  
.btn{  
 padding: 12px;  
 border-radius: 4px;  
 color: var(--background-primary);  
 text-transform: uppercase;  
 font-weight: bold;  
 font-size: 16px;  
 box-shadow: 0px 4px 4px 0px rgba(0, 0, 0, 0.25);  
}  
  
.btnMini{  
 padding: 12px;  
 border-radius: 2px;  
 color: var(--background-primary);  
 text-transform: uppercase;  
 font-weight: bold;  
 font-size: 12px;  
 box-shadow: 0px 4px 4px 0px rgba(0, 0, 0, 0.25);  
}  
  
.btnHalf{  
 width: fit-content;  
}  
  
.right{  
 align-self: flex-end;  
}  
  
.bold{  
 font-weight: bold;  
}  
  
#navBar{  
 padding: 12px 24px;  
}  
  
#navbarRight{  
 gap: 8px;  
}  
  
.navButtons{  
 color: var(--background-primary);  
 padding: 12px;  
 border-radius: 4px;  
 box-shadow: var(--border-sahdow);  
 height: fit-content;  
 align-self: center;  
 cursor: pointer;  
}  
  
#profileLogo{  
 border: 5px solid var(--background-secondary);  
 border-radius: 50%;  
 height: 60px;  
 width: 60px;  
 background:url(./Assets/sampleAvatar.svg);  
 background-size:cover;  
 background-position: center center;  
 margin: 0 12px;  
 cursor: pointer;  
 box-shadow: var(--border-sahdow);  
}  
  
#dashboardContainer{  
 height: 100svh;  
}  
  
#bodyContainer{  
 flex-grow: 1;  
 margin: 24px;  
 gap: 24px;  
}  
  
#mainSection, #actionSection{  
 background-color: var(--background-primary);  
 flex-grow: 1;   
 border-radius: 8px;  
 box-shadow: var(--border-sahdow);  
 padding: 12px 0;  
 height: 75svh;  
}  
  
#mainSection{  
 flex-basis: 70%;  
 gap: 12px;  
}  
  
#actionSection{  
 gap: 12px;  
}  
  
#headerSection{  
 padding: 12px 36px  
}  
  
  
.headerTitleCard{  
 gap: 12px;  
}  
  
.titleTextNormal{  
 color: var(--accent-color);  
 text-transform: uppercase;  
 font-size: 14px;  
}  
  
.headerText{  
 color: var(--secondary);  
 font-size: 14px;  
}  
  
#displayTable{  
 overflow-y: auto;  
}  
  
#headerRow{   
 border-top: 1px solid var(--border);  
 border-bottom: 1px solid var(--border);  
 background-color: var(--background-primary) !important;  
 margin-bottom: 12px;  
}  
  
.row{  
 padding: 8px 12px;  
}  
  
.rowText{  
 text-transform: uppercase;  
 font-size: 14px;  
}  
  
.row:nth-child(odd){  
 background-color: var(--background-secondary) !important;  
}  
  
.talign\_center{  
 text-align: center;  
}  
  
.align\_flex\_end{  
align-self: flex-end;  
}  
  
.error{  
 text-transform: capitalize !important;  
}  
  
#dateTimeSection{  
 text-align-last: right;  
 margin: 4px 12px;  
 gap: 4px;  
}  
  
.dateText{  
 font-size: 22px;  
 color: var(--border);  
}  
  
.timeText{  
 color: #7436F6;  
}  
  
.userText{  
 font-size: 12px;  
 color: var(--secondary);  
}  
  
.wishText{  
 background-color: var(--background-secondary);  
 padding: 12px;  
 font-weight: bold;  
 text-align:center;  
}  
  
.no-rt-border{  
 border-top-right-radius: 0% !important;  
 border-bottom-right-radius: 0% !important;  
}  
  
.no-lt-border{  
 border-top-left-radius: 0% !important;  
 border-bottom-left-radius: 0% !important;  
}  
  
#actionBox{  
 gap: 2px;  
}  
  
#actionBox p{  
 margin: 0 12px;  
 font-size: 14px;  
}  
  
.actionTitle{  
 color: var(--accent-color);  
 text-transform: uppercase;  
 font-weight: bold;  
 margin-top: 12px !important;  
}  
  
.actionOption{  
 color: var(--text-primary);  
 cursor: pointer;  
 transition: all 0.2s ease-in-out;  
}  
  
.actionOption:hover{  
 color: var(--border);  
 font-weight: bold;  
}  
  
#tDateBtn{  
 margin: 12px;  
}  
  
.inputBoxContainer{  
 width: 50%;  
}  
  
.inputBox{  
 gap: 24px;  
 padding-left: 24px;  
 margin: 6px 0;  
}  
  
.gap12{  
 gap: 12px;  
}  
  
.labelText{  
 color: var(--text-primary);  
 font-size: 18px;  
}  
  
.fullModal .row{  
 cursor: pointer;  
}  
  
.fullModal .row:hover{  
 transform: scale(1.01);  
}  
  
#SalesTableSection{  
 flex-basis: 100%;  
 flex-grow: 1;  
}  
  
#SalesTableSection table thead{  
 margin: 8px 0;  
}  
  
.salesTable{  
 margin: 0 12px;  
 text-align:center;  
 border-spacing: 0 12px;  
}  
  
.salesTable input{  
 padding: 10px 8px;  
 text-align: right;  
}  
  
.salesTable select{  
 padding: 12px;  
}  
  
  
#SalesTableSection table tbody tr:nth-child(odd){  
 background-color: #ececec82;  
}  
  
.totalAmount{  
 padding: 12px;  
 font-weight: bold;  
}  
  
.addRowButton{  
 margin: 12px;  
}  
  
#reportTable{  
 color: var(--inputText);  
 font-size: 18px;  
 font-weight: bold;  
}  
  
#reportTable td{  
 padding: 12px;  
}  
  
.row-important{  
 flex-direction: row;  
}  
  
i{  
 margin: 0 4px;  
}  
  
.gap-12{  
 gap: 12px;  
}  
  
#fileInput{  
 border: none;  
 margin: 0;  
 padding: 4px;  
}  
  
/\* Media Query for Responsiveness \*/  
@media screen and (min-width: 320px) and (max-width: 768px) {  
 .imageSection{  
 display: none;  
 }  
  
 .flex-row{  
 flex-direction: column;  
 }  
}  
  
/\* Absolute Sizing \*/  
#navbarLogoImage{  
 height: 60px;  
}

App.js

import './App.css';  
import { BrowserRouter,Routes,Route } from 'react-router-dom';  
import Dashboard from './pages/Dashboard'  
import Login from './pages/Login';  
import Signup from './pages/Signup';  
import { UserContextProvider } from './contexts/userContext';  
import { DateProvider } from './contexts/dateContext';  
import CompanyPage from './pages/CompanyPage';  
import SelectModalWrapper from './Components/SelectModalWrapper';  
import Sales from './pages/Sales';  
import Purchase from './pages/Purchase';  
import Report from './pages/Report';  
  
function App() {  
 return (  
 <UserContextProvider>  
 <DateProvider>  
 <BrowserRouter>  
 <Routes>  
 <Route path='/' element = {<Dashboard />}></Route>  
 <Route path='/login' element = {<Login/>}></Route>  
 <Route path='/createUser' element = {<Signup/>}></Route>  
 <Route path='/create/Company' element = {<CompanyPage type='Create' />} ></Route>  
 <Route path='/alter/Company' element = {<SelectModalWrapper type={'Alter'} />} ></Route>  
 <Route path='/delete/Company' element = {<SelectModalWrapper type={'Delete'} />} ></Route>  
 <Route path='/select/Company' element = {<SelectModalWrapper type={'Select'} />} ></Route>  
 <Route path='/sales' element = {<Sales />} ></Route>  
 <Route path='/purchase' element = {<Purchase />} ></Route>  
 <Route path='/report' element = {<Report />} ></Route>  
 </Routes>  
 </BrowserRouter>  
 </DateProvider>  
 </UserContextProvider>  
 );  
}  
  
export default App;

index.js

import React from 'react';  
import ReactDOM from 'react-dom/client';  
import App from './App';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(  
 <React.StrictMode>  
 <App />  
 </React.StrictMode>  
);

ActionBox.jsx

import React, { useEffect, useContext, useState } from 'react'  
import { useNavigate } from 'react-router-dom';  
import { UserContext } from '../contexts/userContext';  
import Modal from './Modal';  
import ItemModal from '../pages/ItemModal';  
import PaymnetRecieptModal from '../pages/PaymentRecieptModal';  
  
export default function ActionBox() {  
 const nav = useNavigate();  
 const {userName,userId} = useContext(UserContext)  
 const [currentDate, setcurrentDate] = useState("")  
 const [currentTime, setcurrentTime] = useState("")  
 const [isModalOpen, setIsModalOpen] = useState(false)  
 const [isItemModalOpen, setIsItemModalOpen] = useState(false)  
 const [isPModalOpen, setIsPModalOpen] = useState(false)  
 const [isRModalOpen, setIsRModalOpen] = useState(false)  
  
  
 useEffect(()=>{  
 const now = new Date()  
 const options = { day: 'numeric', month: 'long', year: 'numeric' };  
 const formattedDate = now.toLocaleDateString('en-IN', options)  
 const istTime = new Date(now.getTime());  
 const formattedTime = istTime.toTimeString().split(" ")[0].slice(0, 5);  
  
 setcurrentDate(formattedDate)  
 setcurrentTime(formattedTime)  
 },[])  
   
 return (  
 <div id="actionSection" className="flex flex-column">  
 <div id="dateTimeSection" className="flex flex-column">  
 <p className="dateText bld">{currentDate}</p>  
 <p className="timeText bold">{currentTime}</p>  
 <p className="userText">Logged In as {userName}</p>  
 </div>  
 <p className="wishText">Good Morning!</p>  
 <div id="actionBox" className="flex flex-column">  
 <p className="actionTitle">Company</p>  
 <p className="actionOption" onClick={()=> nav(`/select/Company`)}>Select Company</p>  
 <p className="actionOption" onClick={()=> nav(`/create/Company`)}>Create Company</p>  
 <p className="actionOption" onClick={()=> nav(`/alter/Company`)}>Alter Company</p>  
 <p className="actionOption" onClick={()=> nav(`/delete/Company`)}>Delete Company</p>  
  
 <p className="actionTitle">Items</p>  
 <p className="actionOption" onClick={()=> setIsItemModalOpen(true)}>Manage Items</p>  
  
 <p className="actionTitle">Vouchers</p>  
 <p className="actionOption" onClick={()=> nav(`/sales`)}>Sales Voucher</p>  
 <p className="actionOption" onClick={()=> nav(`/purchase`)}>Purchase Voucher</p>  
 <p className="actionOption" onClick={()=> setIsPModalOpen(true)}>Payment</p>  
 <p className="actionOption" onClick={()=> setIsRModalOpen(true)}>Reciept</p>  
  
 <button id="tDateBtn" className="btnMini blue" onClick={()=>setIsModalOpen(true)}><i className='fa-solid fa-clock'></i> Transaction Date</button>   
 </div>  
   
 {isModalOpen && (  
 <Modal onClose = {()=>setIsModalOpen(false)} />  
 )}  
  
 {isItemModalOpen && (  
 <ItemModal onClose={()=>setIsItemModalOpen(false)} userId={userId} />  
 )}  
  
 {isPModalOpen && (  
 <PaymnetRecieptModal onClose={()=>setIsPModalOpen(false)} userId={userId} action={"Payment"} />  
 )}  
  
 {isRModalOpen && (  
 <PaymnetRecieptModal onClose={()=>setIsRModalOpen(false)} userId={userId} action={"Reciept"} />  
 )}  
  
  
 </div>  
 )  
}

Modal.jsx

import React, { useContext, useState } from 'react'  
import { DateContext } from '../contexts/dateContext'  
  
export default function Modal({onClose}) {  
 const {date,setDate} = useContext(DateContext)  
 const [inputDate, setInputDate] = useState(date)  
  
 const changeDate = ()=>{  
 if(inputDate){  
 alert("Transaction Date changed to: " + inputDate)  
 setDate(inputDate)  
 onClose()  
 }else{  
 alert("Please Choose a Valid Date")  
 }  
 }  
  
 return (  
 <div className="modalOverlay">  
 <div className="modal flex flex-column">  
 <p className="titleText">Transaction Date</p>  
 <input type="date" value={inputDate} onChange={(e)=>{setInputDate(e.target.value)}} />  
 <div className="btnBox flex felx-row">  
 <button className="btnMini btnHalf blue" onClick={changeDate}>Submit</button>   
 <button className="btnMini btnHalf red" onClick={onClose}>Cancel</button>   
 </div>  
 </div>  
 </div>  
 )  
}

Navbar.jsx

import { useNavigate } from "react-router-dom";  
import logo from "../Assets/logo.svg"  
  
export default function NavBar(){  
 const nav = useNavigate()  
 const logout = () =>{  
 localStorage.removeItem("uid")  
 localStorage.removeItem("uname")  
 localStorage.removeItem("Company")  
 localStorage.removeItem("Items")  
 localStorage.removeItem("tDate")  
 nav("/login")  
 }  
  
 return(  
 <div id="navBar" className="flex flex-row spaceBetween">  
 <div id="navbarLeft">  
 <img src={logo} alt="Logo Clerky" id="navbarLogoImage" />  
 </div>  
 <div id="navbarRight" className="flex row-important center">  
 <div className="navButtons red" onClick={()=>nav("/report")}>  
 <i className="fa-solid fa-chart-pie"></i>  
 </div>  
 <div className="navButtons green" onClick={()=> nav("/")}>  
 <i className="fa-solid fa-home"></i>  
 </div>  
 <div className="navButtons darkBlue">  
 <i className="fa-solid fa-refresh" onClick={()=>window.location.reload()}></i>  
 </div>  
  
 <div id="profileLogo"></div>  
  
 <div className="navButtons orange" onClick={logout}>  
 <i className="fa-solid fa-door-open"></i>  
 </div>  
 </div>  
 </div>  
 );  
}

SelectModal.jsx

import React, { useEffect } from "react";  
import { useNavigate } from "react-router-dom";  
  
export default function SelectModal({ companies, onSelect }) {  
 const navigate = useNavigate();  
  
 useEffect(() => {  
 const handleKeyDown = (event) => {  
 if (event.key === "Escape") {  
 navigate("/");  
 }  
 };  
 document.addEventListener("keydown", handleKeyDown)  
 return () => {  
 document.removeEventListener("keydown", handleKeyDown);  
 };  
 }, [navigate]);  
 return (  
 <div className="modalOverlay">  
 <div className="fullModal flex flex-column">  
 <p className="modalTitle">Select Company</p>  
 {companies.length > 0 ? (  
 companies.map((company) => (  
 <div  
 key={company.\_id}  
 className="row flex flex-row spaceBetween"  
 onClick={() => onSelect(company)}  
 >  
 <p className="rowText bold">{company.Name}</p>  
 <p className="rowText bold">{company.LastTransactionDate}</p>  
 </div>  
 ))  
 ) : (  
 <p className="rowText bold error talign\_center">No Companies Found</p>  
 )}  
 </div>  
 </div>  
 );  
}

SelectModalWrapper.jsx

import React, { useState, useEffect, useContext } from "react";  
import SelectModal from "../Components/SelectModal";  
import CompanyPage from "../pages/CompanyPage";  
import { UserContext } from "../contexts/userContext";  
import { useNavigate } from "react-router-dom";  
  
export default function SelectModalWrapper({ type }) {  
 const [companies, setCompanies] = useState([]);  
 const [selectedCompany, setSelectedCompany] = useState(null);  
 const {userId} = useContext(UserContext);  
 const nav = useNavigate();  
  
 useEffect(() => {  
 const fetchCompanies = async () => {  
 try{  
 const response = await fetch(`http://localhost:5000/getUser/${userId}`);  
 const data = await response.json()  
 if (response.ok) {  
 console.log(data.message)  
 setCompanies(data.Companies || []);  
 }  
 } catch (error) {  
 console.error("Error fetching User Comapny:", error);  
 }  
 };  
  
 fetchCompanies();  
 }, [userId]);  
  
 const handleDelete = async () => {  
 if (!selectedCompany) return;  
  
 const confirmDelete = window.confirm(  
 `Are you sure you want to delete the company "${selectedCompany.Name}"?`  
 );  
  
 if (confirmDelete) {  
 try {  
 const response = await fetch("http://localhost:5000/deleteCompany", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ userId, companyName: selectedCompany.Name }),  
 });  
  
 const data = await response.json();  
 alert(data.message);  
   
 if(response.ok){  
 setCompanies((prev) =>  
 prev.filter((company) => company.Name !== selectedCompany.Name)  
 );  
 setSelectedCompany(null);  
 }  
   
 } catch (error) {  
 console.error("Error Calling Delete API ", error);  
 }  
 }  
 };  
  
 if (selectedCompany) {  
 if(type === "Select"){  
 localStorage.setItem('Company',selectedCompany.Name)  
 nav('/')  
 }else if(type !== "Delete"){  
 return <CompanyPage type={type} company={selectedCompany} />;  
 }else{  
 handleDelete()  
 }  
 }  
  
 console.log(companies);  
  
 return (  
 <SelectModal  
 companies={companies}  
 onSelect={(company) => setSelectedCompany(company)}  
 />  
 );  
}

dateContext.js

import React, { createContext, useEffect, useState } from 'react';  
  
export const DateContext = createContext();  
  
export const DateProvider = ({ children }) => {  
 const [date, setDate] = useState(() => {  
 return localStorage.getItem("tDate") || new Date().toISOString().split("T")[0];  
 });  
  
 useEffect(() => {  
 localStorage.setItem("tDate", date);  
 }, [date]);  
  
 return (  
 <DateContext.Provider value={{ date, setDate }}>  
 {children}  
 </DateContext.Provider>  
 );  
};

userContext.js

import React, { createContext, useState, useEffect } from 'react';  
  
export const UserContext = createContext();  
  
export const UserContextProvider = ({ children }) => {  
 const [userName, setUserName] =useState(() => {  
 return localStorage.getItem("uname") || "";  
 });  
 const [userId, setUserId] = useState(() => {  
 return localStorage.getItem("uid") || "";  
 });  
  
 useEffect(() => {  
 localStorage.setItem("uname", userName);  
 }, [userName]);  
  
 useEffect(() => {  
 localStorage.setItem("uid", userId);  
 }, [userId]);  
  
  
 return (  
 <UserContext.Provider value={{userName, setUserName, userId, setUserId}}>  
 {children}  
 </UserContext.Provider>  
 );  
};

BulkModal.jsx

import React, {useEffect, useState, useRef } from "react";  
import \* as XLSX from "xlsx";  
  
export default function BulkModal({ onClose, userId, type, itemNameArray, onBulkData }) {  
 const fileInputRef = useRef();  
 const handleFileUpload = (e) => {  
 const file = e.target.files[0];  
 if (!file) return;  
  
 const allowedTypes = [  
 "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",  
 "text/csv",  
 "application/vnd.ms-excel"  
 ];  
  
 if (!allowedTypes.includes(file.type)) {  
 alert("Please upload a valid .xlsx or .csv file.");  
 return;  
 }  
  
 const reader = new FileReader();  
 reader.onload = (evt) => {  
 const data = evt.target.result;  
 const workbook = XLSX.read(data, { type: "binary" });  
  
 const sheetName = workbook.SheetNames[0];  
 const sheet = workbook.Sheets[sheetName];  
 const jsonData = XLSX.utils.sheet\_to\_json(sheet, { header: 1 });  
  
 const [headers, ...rows] = jsonData;  
 const expectedHeaders = ["Item Name", "Quantity", "Price"];  
  
 if (  
 !headers ||  
 headers.length !== 3 ||  
 !expectedHeaders.every((header, idx) => headers[idx] === header)  
 ) {  
 alert("Invalid file format. Headers must be: Item Name, Quantity, Price");  
 return;  
 }  
  
 const validRows = [];  
 const invalidItems = [];  
  
 rows.forEach((row, idx) => {  
 const [itemName, quantity, price] = row;  
 if (!itemNameArray.includes(itemName)) {  
 invalidItems.push(itemName);  
 return;  
 }  
   
 if(quantity == "" || price == ""){  
 return;  
 }  
  
 validRows.push({  
 itemName,  
 quantity: parseFloat(quantity) || 0,  
 price: parseFloat(price) || 0,  
 amount: (parseFloat(quantity) || 0) \* (parseFloat(price) || 0),  
 });  
 });  
  
 if (invalidItems.length > 0) {  
 alert("These items were not found and ignored:\n" + invalidItems.join(", "));  
 }  
  
 if (validRows.length > 0) {  
 onBulkData(validRows);  
 onClose(); // Close the modal after processing  
 }  
 };  
  
 reader.readAsBinaryString(file);  
 };  
  
 const handleDownloadTemplate = () => {  
 const data = itemNameArray.map((name) => ({  
 "Item Name": name,  
 Quantity: "",  
 Price: "",  
 }));  
  
 const worksheet = XLSX.utils.json\_to\_sheet(data, {  
 header: ["Item Name", "Quantity", "Price"],  
 });  
  
 const workbook = XLSX.utils.book\_new();  
 XLSX.utils.book\_append\_sheet(workbook, worksheet, "Template");  
  
 XLSX.writeFile(workbook, "bulk\_upload\_template.xlsx");  
 };  
  
 return (  
 <div className="modalOverlay">  
 <div className="modal flex flex-column">  
 <div className="flex flex-row spaceBetween">  
 <p className="titleText">{"Bulk " + type + " Upload"}</p>  
 <button className="btnMini btnHalf blue" onClick={onClose}>  
 X  
 </button>  
 </div>  
 <div className="flex flex-row gap-12">  
 <button className="btnMini btnHalf orange" onClick={handleDownloadTemplate}>Download Template</button>  
 </div>  
 <div className="flex flex-row gap-12">  
 <input type="file" id="fileInput" accept=".xlsx,.csv"  
 ref={fileInputRef}  
 onChange={handleFileUpload} />  
 </div>  
 </div>  
 </div>  
 );  
}

CompanyPage.jsx

import React, { useState, useEffect, useContext } from "react";  
import Navbar from "../Components/Navbar";  
import ActionBox from "../Components/ActionBox";  
import { UserContext } from "../contexts/userContext";  
import { DateContext } from "../contexts/dateContext";  
  
export default function CompanyPage({ type, company }) {  
 const {date} = useContext(DateContext)  
 const [companyData, setCompanyData] = useState({  
 Name: "",  
 Address: "",  
 State: "",  
 Mobile: "",  
 Email: "",  
 Currency: "",  
 LastTransactionDate: date,  
 Balance: 0  
 });  
  
 const currencies = ["INR", "USD", "GBP", "EUR"];  
 const {userId} = useContext(UserContext)  
  
 useEffect(() => {  
 if (company) {  
 setCompanyData({  
 Name: company.Name,  
 Address: company.Address || "",  
 State: company.State || "",  
 Mobile: company.Mobile || "",  
 Email: company.Email || "",  
 Currency: company.Currency || "",  
 LastTransactionDate: company.LastTransactionDate || "",  
 Balance: company.Balance || 0  
 });  
 }  
 }, [company]);  
  
 const handleChange = (e) => {  
 const { name, value } = e.target;  
 setCompanyData((prev) => ({ ...prev, [name]: value }));  
 };  
  
 const handleSubmit = async () => {  
 const url = (type === 'Create') ? "http://localhost:5000/createCompany" : "http://localhost:5000/alterCompany"  
 const response = await fetch(url, {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({userId,companyData}),  
 });  
 const data = await response.json()  
 alert(data.message)  
 setCompanyData({  
 Name: "",  
 Address: "",  
 State: "",  
 Mobile: "",  
 Email: "",  
 Currency: "",  
 LastTransactionDate: date,  
 Balance: 0  
 })  
 };  
  
 return (  
 <div id="dashboardContainer" className={`Container flex flex-column`}>  
 <Navbar />  
 <div id="bodyContainer" className={`flex flex-row`}>  
 <div id="mainSection" className="flex flex-column">  
 <div id="displayTable" className="flex flex-column">  
 <div id="headerRow" className="row flex flex-row spaceBetween">  
 <p className="titleTextNormal bold">{type} Company</p>  
 </div>  
 <div className="inputBoxContainer">  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Company Name</p>  
 <input  
 type="text"  
 name="Name"  
 placeholder="eg. ABC Corporation"  
 value={companyData.Name}  
 onChange={handleChange}  
 />  
 </div>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Address</p>  
 <input  
 type="text"  
 name="Address"  
 placeholder="eg. 248 Diego Street, LA"  
 value={companyData.Address}  
 onChange={handleChange}  
 />  
 </div>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">State</p>  
 <input  
 type="text"  
 name="State"  
 placeholder="Select State"  
 value={companyData.State}  
 onChange={handleChange}  
 />  
 </div>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Mobile No</p>  
 <input  
 type="tel"  
 name="Mobile"  
 placeholder="eg. 9876543210"  
 value={companyData.Mobile}  
 onChange={handleChange}  
 />  
 </div>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Email</p>  
 <input  
 type="email"  
 name="Email"  
 placeholder="eg. info@abc.com"  
 value={companyData.Email}  
 onChange={handleChange}  
 />  
 </div>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Currency</p>  
 <input  
 list="currencyOptions"  
 name="Currency"  
 placeholder="Base Currency"  
 value={companyData.Currency}  
 onChange={handleChange}  
 />  
 <datalist id="currencyOptions">  
 {currencies.map((currency) => (  
 <option key={currency} value={currency}>  
 {currency}  
 </option>  
 ))}  
 </datalist>  
 </div>  
 <div className="btnBox flex flex-row">  
 <button  
 className="btnMini btnHalf orange"  
 onClick={() =>  
 setCompanyData({  
 Name: "",  
 Address: "",  
 State: "",  
 Mobile: "",  
 Email: "",  
 Currency: "",  
 LastTransactionDate: date,  
 Balance: 0  
 })  
 }  
 >  
 Reset  
 </button>  
 <button  
 type="submit"  
 className="btnMini btnHalf blue"  
 onClick={handleSubmit}  
 >  
 Submit  
 </button>  
 </div>  
 </div>  
 </div>  
 </div>  
 <ActionBox title={"Company"} />  
 </div>  
 </div>  
 );  
}

Dashboard.jsx

import Navbar from "../Components/Navbar";  
import ActionBox from "../Components/ActionBox";  
import { UserContext } from "../contexts/userContext";  
import { DateContext } from "../contexts/dateContext";  
import { useNavigate } from "react-router-dom";  
import { useContext, useEffect, useState } from "react";  
  
export default function Dashboard() {  
 const nav = useNavigate();  
 const { userId } = useContext(UserContext);  
 const { date } = useContext(DateContext);  
 const [companies, setCompanies] = useState([]);  
 const [loading, setLoading] = useState(true);  
  
 useEffect(() => {  
 if (!userId) {  
 nav("/login");  
 return;  
 }  
  
 const fetchCompanies = async () => {  
 try {  
 const response = await fetch(`http://localhost:5000/getUser/${userId}`);  
 const data = await response.json();  
  
 if (response.ok) {  
 console.log(data.message)  
 setCompanies(data.Companies || []);  
 }  
 } catch (error) {  
 console.error("Error fetching User Compnay:", error);  
 } finally {  
 setLoading(false);  
 }  
 };  
  
 fetchCompanies();  
 }, [userId, nav]);  
  
 return (  
 <div id="dashboardContainer" className="Container flex flex-column">  
 <Navbar />  
 <div id="bodyContainer" className="flex flex-row">  
 <div id="mainSection" className="flex flex-column">  
 <div id="headerSection" className="flex flex-row spaceBetween">  
 <div className="headerTitleCard flex flex-column">  
 <p className="titleTextNormal">Current Period</p>  
 <p className="headerText bold">01 Apr 2025 - 31 Mar 2026</p>  
 </div>  
 <div className="headerTitleCard flex flex-column">  
 <p className="titleTextNormal">Transaction Date</p>  
 <p className="headerText bold">{date}</p>  
 </div>  
 </div>  
 <div id="displayTable" className="flex flex-column">  
 <div id="headerRow" className="row flex flex-row spaceBetween">  
 <p className="titleTextNormal bold">Company Name</p>  
 <p className="titleTextNormal bold">Last Transaction Date</p>  
 </div>  
  
 {loading && (  
 <p className="rowText bold talign\_center">Loading...</p>  
 )}  
  
 {!loading && companies.length === 0 && (  
 <p className="rowText bold error talign\_center">No Companies Created</p>  
 )}  
  
 {companies.map((company, index) => (  
 <div  
 key={index}  
 className="row flex flex-row spaceBetween"  
 >  
 <p className="rowText bold">M/s {company.Name}</p>  
 <p className="rowText">{company.LastTransactionDate ? company.LastTransactionDate : "No Vouchers Entered"}</p>  
 </div>  
 ))}  
 </div>  
 </div>  
 <ActionBox />  
 </div>  
 </div>  
 );  
}

ItemModal.jsx

import React, {useEffect, useState } from "react";  
import { useNavigate } from "react-router-dom";  
  
export default function ItemModal({ onClose, userId }) {  
 const [action, setAction] = useState("");  
 const [itemName, setItemName] = useState("");  
 const [qtyInStock, setQtyInStock] = useState(0);  
 const [selectedDeleteItem, setSelectedDeleteItem] = useState("");  
 const [selectedAlterItem, setSelectedAlterItem] = useState("");  
 const [items, setItems] = useState([]);  
  
 const handleAction = (selectedAction) => {  
 setAction(selectedAction);  
 };  
  
 const companyName = localStorage.getItem("Company");  
 const nav = useNavigate();  
  
 useEffect(() => {  
 const fetchCompanies = async () => {  
 if (!companyName) {  
 alert("Select a Company First");  
 nav(`/select/Company`);  
 return;  
 } else {  
 if (action === "alter" || action === "delete") {  
 try {  
 const response = await fetch(`http://localhost:5000/fetchItems`, {  
 method: 'POST',  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ userId: userId, companyName: companyName }),  
 });  
  
 const data = await response.json();  
 console.log(data.message)  
 if (response.ok && data.items) {  
 setItems(data.items);  
 }  
 } catch (error) {  
 console.log("Error Fetching Items " + error);  
 }  
 }  
 }  
 };  
  
 fetchCompanies();  
 }, [action, companyName, userId, nav]);  
  
 const onSubmit = async() =>{  
 const url = (action === "create") ? "http://localhost:5000/createItems" : "http://localhost:5000/updateItems"  
 try {  
 const response = await fetch(url, {  
 method: 'POST',  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ userId: userId, companyName: companyName, itemName: itemName, qtyInStock: qtyInStock }),  
 });  
  
 const data = await response.json();  
 alert(data.message)  
 } catch (error) {  
 console.log("Error Creating/Updating Items " + error);  
 }  
 }  
  
 const onDelete = async () =>{  
 try {  
 const response = await fetch("http://localhost:5000/deleteItems", {  
 method: 'POST',  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ userId: userId, companyName: companyName, itemName: selectedDeleteItem }),  
 });  
  
 const data = await response.json();  
 alert(data.message)  
 } catch (error) {  
 console.log("Error Deleting Items " + error);  
 }  
 }  
  
 useEffect(() => {  
 const fetchItemDetails = () => {  
 const item = items.find((item) => item.Name === selectedAlterItem);  
 setItemName(item?.Name || "");  
 setQtyInStock(item?.Qty || 0);  
 };  
 if (selectedAlterItem) {  
 fetchItemDetails();  
 }  
 }, [selectedAlterItem, items]);  
  
 const handleSubmit = () => {  
 if (action === "create" || action === "alter") {  
 if (itemName.trim() === "" || qtyInStock <= 0) {  
 alert("Please provide valid item details.");  
 return;  
 }  
 onSubmit();  
 setItemName("");  
 setQtyInStock(0);  
 setSelectedAlterItem("")  
 handleAction("");  
 }  
 };  
  
 const handleDelete = () => {  
 if (!selectedDeleteItem) {  
 alert("Please select an item to delete.");  
 return;  
 }  
 onDelete()  
 setSelectedDeleteItem("")  
 handleAction("");  
 };  
  
 return (  
 <div className="modalOverlay">  
 <div className="modal flex flex-column">  
 <div className="flex flex-row spaceBetween">  
 <p className="titleText">Manage Items</p>  
 <button className="btnMini btnHalf blue" onClick={onClose}>  
 X  
 </button>  
 </div>  
  
 {action === "create" ? (  
 <>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Item Name</p>  
 <input  
 type="text"  
 name="Name"  
 placeholder="eg. ABC Corporation"  
 value={itemName}  
 onChange={(e) => setItemName(e.target.value)}  
 />  
 </div>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Qty In Stock</p>  
 <input  
 type="number"  
 name="Qty"  
 value={qtyInStock}  
 onChange={(e) => setQtyInStock(parseInt(e.target.value) || 0)}   
 />  
 </div>  
 <div className="btnBox flex flex-row">  
 <button className="btnMini btnHalf blue" onClick={handleSubmit}>  
 Submit  
 </button>  
 <button className="btnMini btnHalf red" onClick={() => handleAction("")}>  
 Cancel  
 </button>  
 </div>  
 </>  
 ) : action === "alter" ? (  
 <>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Select Item</p>  
 <input  
 list="items"  
 name="Items"  
 placeholder="Select Item to Alter"  
 value={selectedAlterItem}  
 onChange={(e) => setSelectedAlterItem(e.target.value)}  
 />  
 <datalist id="items">  
 {items.map((item) => (  
 <option key={item.Name} value={item.Name}>  
 {item.Name}  
 </option>  
 ))}  
 </datalist>  
 </div>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Item Name</p>  
 <input  
 type="text"  
 name="Name"  
 placeholder="eg. ABC Corporation"  
 value={itemName}  
 onChange={(e) => setItemName(e.target.value)}  
 />  
 </div>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Qty In Stock</p>  
 <input  
 type="number"  
 name="Qty"  
 value={qtyInStock}  
 onChange={(e) => setQtyInStock(parseInt(e.target.value) || 0)} // Parse to number  
 />  
 </div>  
 <div className="btnBox flex flex-row">  
 <button className="btnMini btnHalf blue" onClick={handleSubmit}>  
 Submit  
 </button>  
 <button className="btnMini btnHalf red" onClick={() => handleAction("")}>  
 Cancel  
 </button>  
 </div>  
 </>  
 ) : action === "delete" ? (  
 <>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Select Item</p>  
 <input  
 list="items"  
 name="Items"  
 placeholder="Select Item to Delete"  
 value={selectedDeleteItem}  
 onChange={(e) => setSelectedDeleteItem(e.target.value)}  
 />  
 <datalist id="items">  
 {items.map((item) => (  
 <option key={item.Name} value={item.Name}>  
 {item.Name}  
 </option>  
 ))}  
 </datalist>  
 </div>  
 <div className="btnBox flex flex-row">  
 <button className="btnMini btnHalf red" onClick={handleDelete}>  
 Delete  
 </button>  
 <button className="btnMini btnHalf blue" onClick={() => handleAction("")}>  
 Cancel  
 </button>  
 </div>  
 </>  
 ) : (  
 <div className="btnBox flex flex-row">  
 <button  
 className="btnMini btnHalf green"  
 onClick={() => handleAction("create")}  
 >  
 Create Item  
 </button>  
 <button  
 className="btnMini btnHalf orange"  
 onClick={() => handleAction("alter")}  
 >  
 Alter Item  
 </button>  
 <button  
 className="btnMini btnHalf red"  
 onClick={() => handleAction("delete")}  
 >  
 Delete Item  
 </button>  
 </div>  
 )}  
 </div>  
 </div>  
 );  
}

Login.jsx

import { useNavigate } from "react-router-dom";  
import "../App.css";  
import Logo from "../Assets/logo.svg";  
import { useState, useContext, useEffect } from "react";  
import {UserContext} from "../contexts/userContext";  
  
export default function Login(){  
 const nav =useNavigate();  
 const [emailuser, setEmailuser] = useState("")  
 const [Password, setPassword] = useState("")  
 const {setUserId, setUserName} = useContext(UserContext);  
  
 useEffect(()=>{  
 if(localStorage.getItem("uid")){  
 setUserId(localStorage.getItem("uid"))  
 setUserName(localStorage.getItem("uname"))  
 nav("/")  
 }  
 })  
  
 const authenticate = async ()=>{  
 try{  
 const response = await fetch("http://localhost:5000/loginUser",{  
 method: 'POST',  
 headers:{'Content-Type': 'application/json'},  
 body: JSON.stringify({usernameEmail: emailuser, password: Password})  
 })  
   
 const data = await response.json()  
  
 if(response.ok){  
 alert(data.message)  
 console.log(data.user)  
 }  
  
 if(data.user){  
 setUserId(data.user.\_id)  
 setUserName(data.user.Username)  
 localStorage.setItem("uid",data.user.\_id)  
 localStorage.setItem("uname",data.user.Username)  
 nav("/")  
 }  
  
 }catch(error){  
 console.error("Error Calling Login API " + error)  
 alert("Failed Fetch")  
 }  
 }  
   
 return(  
 <div id="loginContainer" className="Container flex flex-row">  
 <div className="imageSection">  
 </div>  
 <div className="formSection flex flex-column spaceBetween">  
 <button className="btn btnHalf black right" onClick={()=>nav("/createUser")}>New to Clerky</button>  
 <div id="logoBox" className="flex flex-column">  
 <img src={Logo} alt="Logo" id="logoImg" />  
 <p className="titleText">Welcome to Clerky</p>  
 <p className="subText">Every Boss's Favourite</p>  
 </div>  
 <div id="formBox" className="flex flex-column">  
 <input type="text" placeholder="Email or Username" onChange={(e)=>{setEmailuser(e.target.value)}} value={emailuser} />  
 <input type="password" placeholder="Password" onChange={(e) => {setPassword(e.target.value)}} value={Password} />  
 <button className="btn blue" onClick={authenticate}>Login</button>   
 </div>  
 <p className="subText center">Please Sign In to Continue</p>  
 </div>  
 </div>  
 );  
}

PaymentRecieptModal.jsx

import React, {useContext, useEffect, useState } from "react";  
import {DateContext} from "../contexts/dateContext";  
  
export default function PaymnetRecieptModal({ onClose, userId, action }) {  
 const [amount, setAmount] = useState(0)  
 const {date} = useContext(DateContext)  
  
 const companyName = localStorage.getItem("Company");  
   
 const onSubmit = async() =>{  
 const transactionID = `TXN-${Date.now()}`;  
 const url = (action === "Payment") ? "http://localhost:5000/removeBalance" : "http://localhost:5000/addBalance"  
 try {  
 const response = await fetch(url, {  
 method: 'POST',  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ userId: userId, companyName: companyName, amount: amount, date: date, transactionID: transactionID }),  
 });  
  
 const data = await response.json();  
 alert(data.message)  
 } catch (error) {  
 console.log("Error Creating/Updating Items " + error);  
 }  
 }  
  
 const handleSubmit = () => {  
 if (amount<0) {  
 alert("Please provide valid Amount.");  
 return;  
 }  
  
 onSubmit();  
 setAmount(0)  
 onClose();  
 };  
  
 return (  
 <div className="modalOverlay">  
 <div className="modal flex flex-column">  
 <div className="flex flex-row spaceBetween">  
 <p className="titleText">{action} Voucher</p>  
 <button className="btnMini btnHalf blue" onClick={onClose}>  
 X  
 </button>  
 </div>  
 <>  
 <div className="inputBox flex flex-row alignItems\_Center spaceBetween">  
 <p className="labelText bold">Amount</p>  
 <input  
 type="number"  
 name="Amount"  
 placeholder= {action === "Payment" ? "Enter Amount Paid" : "Enter Amount Recieved"}  
 value={amount}  
 onChange={(e) => setAmount(e.target.value)}  
 />  
 </div>  
  
 <div className="btnBox flex flex-row">  
 <button className="btnMini btnHalf blue" onClick={handleSubmit}>  
 Submit  
 </button>  
 <button className="btnMini btnHalf red" onClick={() => setAmount(0)}>  
 Reset  
 </button>  
 </div>  
 </>  
 </div>  
 </div>  
 );  
}

Purchase.jsx

import Navbar from "../Components/Navbar";  
import ActionBox from "../Components/ActionBox";  
import { UserContext } from "../contexts/userContext";  
import { DateContext } from "../contexts/dateContext";  
import { useNavigate } from "react-router-dom";  
import { useContext, useEffect, useState } from "react";  
import BulkModal from "../pages/BulkModal";  
  
export default function Purchase() {  
 const nav = useNavigate();  
 const { userId } = useContext(UserContext);  
 const { date } = useContext(DateContext);  
 const companyName = localStorage.getItem("Company");  
 const [itemNames, setItemNames] = useState([]);  
 const [rows, setRows] = useState([]);  
 const [totalAmount, setTotalAmount] = useState(0);  
 const [bulkModal, setbulkModal] = useState(false);  
   
 useEffect(() => {  
 const fetchCompanies = async () => {  
 if (!companyName) {  
 alert("Select a Company First");  
 nav(`/select/Company`);  
 return;  
 } else {  
 try {  
 const response = await fetch(`http://localhost:5000/fetchItems`, {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ userId: userId, companyName: companyName }),  
 });  
  
 const data = await response.json();  
 if (response.ok && data.items) {  
 setItemNames(data.items.map((item) => item.Name));  
 }  
 } catch (error) {  
 console.log("Error Fetching Items " + error);  
 }  
 }  
 };  
  
 fetchCompanies();  
 }, [companyName, userId, nav]);  
  
 const addRow = () => {  
 setRows([...rows, { itemName: "", quantity: 0, price: 0, amount: 0 }]);  
 };  
  
 const updateRow = (index, field, value) => {  
 const updatedRows = [...rows];  
   
 if (field === "quantity" || field === "price") {  
 if (!updatedRows[index].itemName) {  
 alert("Please select an item before entering quantity or price.");  
 return;  
 }  
 }  
   
 updatedRows[index][field] = value;  
   
 if (field === "quantity" || field === "price") {  
 updatedRows[index].amount = updatedRows[index].quantity \* updatedRows[index].price;  
 }  
   
 setRows(updatedRows);  
 calculateTotalAmount(updatedRows);  
 };  
   
  
 const calculateTotalAmount = (rows) => {  
 const total = rows.reduce((acc, row) => acc + (row.amount || 0), 0);  
 setTotalAmount(total);  
 };  
  
 const removeRow = (index) => {  
 const updatedRows = rows.filter((\_, i) => i !== index);  
 setRows(updatedRows);  
 calculateTotalAmount(updatedRows);  
 };  
  
 const handleSubmit = async () => {  
 if (!companyName) {  
 alert("Select a Company First");  
 return;  
 }  
  
 const transactionID = `TXN-${Date.now()}`;  
  
 const payload = {  
 userId: userId,  
 companyName: companyName,  
 transactionID: transactionID,  
 date: date,  
 rows: rows,  
 totalAmount: totalAmount,  
 };  
  
 try {  
 const response = await fetch("http://localhost:5000/submitPurchase", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify(payload),  
 });  
 const data = await response.json();  
 if (response.ok) {  
 alert(data.message);  
 if(data.success){  
 resetForm();  
 }  
 }  
 } catch (error) {  
 console.log("Error Submitting Transaction:", error);  
 }  
 };  
  
 const resetForm = () => {  
 setRows([]);  
 setTotalAmount(0);  
 };  
  
 const handleBulkData = (bulkRows) => {  
 setRows(bulkRows);  
 calculateTotalAmount(bulkRows);  
 };  
  
  
 return (  
 <div id="dashboardContainer" className="Container flex flex-column">  
 <Navbar />  
 <div id="bodyContainer" className="flex flex-row">  
 {bulkModal && (<BulkModal onClose={()=>setbulkModal(false)} userId={userId} type="Purchase" itemNameArray={itemNames} onBulkData={handleBulkData} />)}   
 <div id="mainSection" className="flex flex-column">  
 <div id="headerSection" className="flex flex-row spaceBetween">  
 <div className="headerTitleCard flex flex-column">  
 <p className="titleTextNormal">Transaction Date</p>  
 <p className="headerText bold">{date}</p>  
 </div>  
 <div className="headerTitleCard flex flex-column">  
 <button className="btnMini btnHalf blue" onClick={()=>setbulkModal(true)}>  
 <i className="fa-solid fa-upload"></i> Bulk Upload  
 </button>  
 </div>  
 </div>  
  
 <div id="displayTable" className="flex flex-column">  
 <div id="headerRow" className="row flex flex-row spaceBetween">  
 <p className="titleTextNormal bold">Purchase Voucher</p>  
 </div>  
 <div className="flex flex-column spaceBetween" id="SalesTableSection">  
 <table className="salesTable">  
 <thead>  
 <tr>  
 <th className="col-blue">Item Name</th>  
 <th className="col-blue">Quantity</th>  
 <th className="col-blue">Price</th>  
 <th className="col-blue">Amount</th>  
 <th></th>  
 </tr>  
 </thead>  
 <tbody>  
 {rows.map((row, index) => (  
 <tr key={index}>  
 <td>  
 <select  
 value={row.itemName}  
 onChange={(e) => updateRow(index, "itemName", e.target.value)}  
 >  
 <option value="">Select Item</option>  
 {itemNames.map((itemName) => (  
 <option key={itemName} value={itemName}>  
 {itemName}  
 </option>  
 ))}  
 </select>  
 </td>  
 <td>  
 <input  
 type="number"  
 value={row.quantity}  
 onChange={(e) => updateRow(index, "quantity", parseFloat(e.target.value) || 0)}  
 />  
 </td>  
 <td>  
 <input  
 type="number"  
 value={row.price}  
 onChange={(e) => updateRow(index, "price", parseFloat(e.target.value) || 0)}  
 />  
 </td>  
 <td>{row.amount.toFixed(2)}</td>  
 <td>  
 <button className="btnHalf red col-white circle" onClick={() => removeRow(index)}><i className="fa-solid fa-minus"></i></button>  
 </td>  
 </tr>  
 ))}  
 </tbody>  
 </table>  
  
 <button className="addRowButton btnMini green col-white centerBoth" onClick={addRow}>  
 <i className="fa-solid fa-plus"></i> New Item  
 </button>  
  
 <div className="totalAmount">  
 <p className="talign\_right">Total Amount: {totalAmount.toFixed(2)}</p>  
 </div>  
  
 <div className="flex row gap12 align\_flex\_end">  
 <button className="btnHalf btnMini blue" onClick={handleSubmit}>Purchase</button>  
 <button className="btnHalf btnMini orange" onClick={resetForm}>Reset</button>  
 </div>  
 </div>  
 </div>  
 </div>  
 <ActionBox />  
 </div>  
 </div>  
 );  
}

Report.jsx

import Navbar from "../Components/Navbar";  
import ActionBox from "../Components/ActionBox";  
import { UserContext } from "../contexts/userContext";  
import { DateContext } from "../contexts/dateContext";  
import { useNavigate } from "react-router-dom";  
import { useContext, useEffect, useState } from "react";  
  
export default function Report() {  
 const nav = useNavigate();  
 const { userId } = useContext(UserContext);  
 const { date } = useContext(DateContext);  
 const companyName = localStorage.getItem("Company");  
  
 const [itemWiseData, setItemWiseData] = useState([]);  
 const [voucherWiseData, setVoucherWiseData] = useState([]);  
 const [balance, setBalance] = useState(0);  
 const [view, setView] = useState("item");  
  
 useEffect(() => {  
 const fetchReportData = async () => {  
 if (!companyName) {  
 alert("Select a Company First");  
 nav(`/select/Company`);  
 return;  
 } else {  
 try {  
 const response = await fetch(`http://localhost:5000/fetchReports`, {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ userId, companyName }),  
 });  
  
 const data = await response.json();  
 if (response.ok && data) {  
 setBalance(data.balance);  
 setItemWiseData(data.itemWise);  
 setVoucherWiseData(data.voucherWise);  
 }  
 } catch (error) {  
 console.log("Error Fetching Reports " + error);  
 }  
 }  
 };  
  
 fetchReportData();  
 }, [companyName, userId, nav]);  
  
 return (  
 <div id="dashboardContainer" className="Container flex flex-column">  
 <Navbar />  
 <div id="bodyContainer" className="flex flex-row">  
 <div id="mainSection" className="flex flex-column">  
 <div id="headerSection" className="flex flex-row spaceBetween">  
 <div className="headerTitleCard flex flex-column">  
 <p className="titleTextNormal">Report Date</p>  
 <p className="headerText bold">{date}</p>  
 </div>  
 <div className="headerTitleCard flex flex-column">  
 <p className="titleTextNormal">Comapny Balance</p>  
 <p className="headerText bold">{balance}</p>  
 </div>  
 <div className="headerActions">  
 <button className={`btnMini ${view === "item" ? "blue" : "black"} no-rt-border`} onClick={() => setView("item")}>  
 Item-Wise View  
 </button>  
 <button className={`btnMini ${view === "voucher" ? "blue" : "black"} no-lt-border`} onClick={() => setView("voucher")}>  
 Voucher-Wise View  
 </button>  
 </div>  
 </div>  
   
 <div id="displayTable" className="flex flex-column">  
 <div id="headerRow" className="row flex flex-row center">  
 <p className="titleTextNormal bold">{view === "item" ? "Item Wise Report" : "Voucher Wise Report"}</p>  
 </div>  
 {view === "item" ? (  
 <div className="flex flex-column spaceBetween" id="SalesTableSection">  
 <table className="salesTable">  
 <thead>  
 <tr>  
 <th className="col-blue">Item</th>  
 <th className="col-blue">Sold Qty</th>  
 <th className="col-blue">Sold Amount</th>  
 <th className="col-blue">Purchase Qty</th>  
 <th className="col-blue">Purchase Amount</th>  
 </tr>  
 </thead>  
 <tbody id="reportTable">  
 {itemWiseData.map((item, index) => (  
 <tr key={index}>  
 <td>{item.itemName}</td>  
 <td>{item.totalSalesQuantity}</td>  
 <td>{item.totalSales}</td>  
 <td>{item.totalPurchaseQuantity}</td>  
 <td>{item.totalPurchase}</td>  
 </tr>  
 ))}  
 </tbody>  
 </table>  
 </div>  
 ) : (  
 <div className="flex flex-column spaceBetween" id="SalesTableSection">  
 <table className="salesTable">  
 <thead>  
 <tr>  
 <th className="col-blue">Voucher Type</th>  
 <th className="col-blue">Total Amount</th>  
 </tr>  
 </thead>  
 <tbody id="reportTable">  
 {Object.entries(voucherWiseData).map(([type, amount], index) => (  
 <tr key={index}>  
 <td>{type.charAt(0).toUpperCase() + type.slice(1)}</td>  
 <td>{Number(amount).toFixed(2)}</td>  
 </tr>  
 ))}  
 </tbody>  
 </table>  
 </div>  
 )}  
 </div>  
 </div>  
 <ActionBox />  
 </div>  
 </div>  
 );  
}

Sales.jsx

import Navbar from "../Components/Navbar";  
import ActionBox from "../Components/ActionBox";  
import BulkModal from "../pages/BulkModal";  
import { UserContext } from "../contexts/userContext";  
import { DateContext } from "../contexts/dateContext";  
import { useNavigate } from "react-router-dom";  
import { useContext, useEffect, useState } from "react";  
  
  
export default function Sales() {  
 const nav = useNavigate();  
 const { userId } = useContext(UserContext);  
 const { date } = useContext(DateContext);  
 const companyName = localStorage.getItem("Company");  
 const [itemNames, setItemNames] = useState([]);  
 const [rows, setRows] = useState([]);  
 const [totalAmount, setTotalAmount] = useState(0);  
 const [bulkModal, setbulkModal] = useState(false);  
  
 useEffect(() => {  
 const fetchCompanies = async () => {  
 if (!companyName) {  
 alert("Select a Company First");  
 nav(`/select/Company`);  
 return;  
 } else {  
 try {  
 const response = await fetch(`http://localhost:5000/fetchItems`, {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ userId: userId, companyName: companyName }),  
 });  
  
 const data = await response.json();  
 if (response.ok && data.items) {  
 setItemNames(data.items.map((item) => item.Name));  
 }  
 } catch (error) {  
 console.log("Error Fetching Items " + error);  
 }  
 }  
 };  
  
 fetchCompanies();  
 }, [companyName, userId, nav]);  
  
 const addRow = () => {  
 setRows([...rows, { itemName: "", quantity: 0, price: 0, amount: 0 }]);  
 };  
  
 const updateRow = (index, field, value) => {  
 const updatedRows = [...rows];  
   
 if (field === "quantity" || field === "price") {  
 if (!updatedRows[index].itemName) {  
 alert("Please select an item before entering quantity or price.");  
 return;  
 }  
 }  
   
 updatedRows[index][field] = value;  
   
 if (field === "quantity" || field === "price") {  
 updatedRows[index].amount = updatedRows[index].quantity \* updatedRows[index].price;  
 }  
   
 setRows(updatedRows);  
 calculateTotalAmount(updatedRows);  
 };  
   
  
 const calculateTotalAmount = (rows) => {  
 const total = rows.reduce((acc, row) => acc + (row.amount || 0), 0);  
 setTotalAmount(total);  
 };  
  
 const removeRow = (index) => {  
 const updatedRows = rows.filter((\_, i) => i !== index);  
 setRows(updatedRows);  
 calculateTotalAmount(updatedRows);  
 };  
  
 const handleSubmit = async () => {  
 if (!companyName) {  
 alert("Select a Company First");  
 return;  
 }  
  
 const transactionID = `TXN-${Date.now()}`;  
  
 const payload = {  
 userId: userId,  
 companyName: companyName,  
 transactionID: transactionID,  
 date: date,  
 rows: rows,  
 totalAmount: totalAmount,  
 };  
  
 try {  
 const response = await fetch("http://localhost:5000/submitSales", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify(payload),  
 });  
 const data = await response.json();  
 if (response.ok) {  
 alert(data.message);  
 if(data.success){  
 resetForm();  
 }  
 }  
 } catch (error) {  
 console.log("Error Submitting Transaction:", error);  
 }  
 };  
  
 const resetForm = () => {  
 setRows([]);  
 setTotalAmount(0);  
 };  
  
 const handleBulkData = (bulkRows) => {  
 setRows(bulkRows);  
 calculateTotalAmount(bulkRows);  
 };  
  
 return (  
 <div id="dashboardContainer" className="Container flex flex-column">  
 <Navbar />  
 <div id="bodyContainer" className="flex flex-row">  
 {bulkModal && (<BulkModal onClose={()=>setbulkModal(false)} userId={userId} type="Sales" itemNameArray={itemNames} onBulkData={handleBulkData} />)}   
 <div id="mainSection" className="flex flex-column">  
 <div id="headerSection" className="flex flex-row spaceBetween">  
 <div className="headerTitleCard flex flex-column">  
 <p className="titleTextNormal">Transaction Date</p>  
 <p className="headerText bold">{date}</p>  
 </div>  
 <div className="headerTitleCard flex flex-column">  
 <button className="btnMini btnHalf blue" onClick={()=>setbulkModal(true)}>  
 <i className="fa-solid fa-upload"></i> Bulk Upload  
 </button>  
 </div>  
 </div>  
  
 <div id="displayTable" className="flex flex-column">  
 <div id="headerRow" className="row flex flex-row spaceBetween">  
 <p className="titleTextNormal bold">Sales Voucher</p>  
 </div>  
 <div className="flex flex-column spaceBetween" id="SalesTableSection">  
 <table className="salesTable">  
 <thead>  
 <tr>  
 <th className="col-blue">Item Name</th>  
 <th className="col-blue">Quantity</th>  
 <th className="col-blue">Price</th>  
 <th className="col-blue">Amount</th>  
 <th></th>  
 </tr>  
 </thead>  
 <tbody>  
 {rows.map((row, index) => (  
 <tr key={index}>  
 <td>  
 <select  
 value={row.itemName}  
 onChange={(e) => updateRow(index, "itemName", e.target.value)}  
 >  
 <option value="">Select Item</option>  
 {itemNames.map((itemName) => (  
 <option key={itemName} value={itemName}>  
 {itemName}  
 </option>  
 ))}  
 </select>  
 </td>  
 <td>  
 <input  
 type="number"  
 value={row.quantity}  
 onChange={(e) => updateRow(index, "quantity", parseFloat(e.target.value) || 0)}  
 />  
 </td>  
 <td>  
 <input  
 type="number"  
 value={row.price}  
 onChange={(e) => updateRow(index, "price", parseFloat(e.target.value) || 0)}  
 />  
 </td>  
 <td>{row.amount.toFixed(2)}</td>  
 <td>  
 <button className="btnHalf red col-white circle" onClick={() => removeRow(index)}><i className="fa-solid fa-minus"></i></button>  
 </td>  
 </tr>  
 ))}  
 </tbody>  
 </table>  
  
 <button className="addRowButton btnMini green col-white centerBoth" onClick={addRow}>  
 <i className="fa-solid fa-plus"></i> New Item  
 </button>  
  
 <div className="totalAmount">  
 <p className="talign\_right">Total Amount: {totalAmount.toFixed(2)}</p>  
 </div>  
  
 <div className="flex row gap12 align\_flex\_end">  
 <button className="btnHalf btnMini blue" onClick={handleSubmit}>Sale</button>  
 <button className="btnHalf btnMini orange" onClick={resetForm}>Reset</button>  
 </div>  
 </div>  
 </div>  
 </div>  
 <ActionBox />  
 </div>  
 </div>  
 );  
}

Signup.jsx

import { useState } from "react";  
import { useNavigate } from "react-router-dom";  
import "../App.css";  
import Logo from "../Assets/logo.svg";  
  
export default function Signup() {  
 const nav = useNavigate();  
 const [username, setUsername] = useState("");  
 const [email, setEmail] = useState("");  
 const [password, setPassword] = useState("");  
  
 const createUser = async () => {  
 try {  
 const response = await fetch("http://localhost:5000/createUser", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ username, email, password }),  
 });  
  
 const data = await response.json();  
  
 if (response.ok) {  
 alert(data.message);  
 if(data.response){  
 localStorage.removeItem("uid");  
 localStorage.removeItem("uname");  
 nav("/login")  
 }  
 }  
  
 } catch (error) {  
 console.error("Error during signup: " + error);  
 alert("Failed Insert")  
 }  
 };  
  
 return (  
 <div id="loginContainer" className="Container flex flex-row">  
 <div className="imageSection"></div>  
 <div className="formSection flex flex-column spaceBetween">  
 <button className="btn btnHalf black right" onClick={() => nav("/login")}>  
 Login  
 </button>  
 <div id="logoBox" className="flex flex-column">  
 <img src={Logo} alt="Logo" id="logoImg" />  
 <p className="titleText">Welcome to Clerky</p>  
 <p className="subText">Every Boss's Favourite</p>  
 </div>  
 <div id="formBox" className="flex flex-column">  
 <input  
 type="text"  
 placeholder="Username"  
 onChange={(e) => setUsername(e.target.value)}  
 value={username}  
 />  
 <input  
 type="email"  
 placeholder="Email"  
 onChange={(e) => setEmail(e.target.value)}  
 value={email}  
 />  
 <input  
 type="password"  
 placeholder="Password"  
 onChange={(e) => setPassword(e.target.value)}  
 value={password}  
 />  
 <button className="btn blue" onClick={createUser}>  
 Signup  
 </button>  
 </div>  
 <p className="subText center">Sign Up to avail Premium</p>  
 </div>  
 </div>  
 );  
}