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

import {

getDoc,

addDoc,

collection,

doc,

updateDoc,

arrayUnion,

setDoc,

query,

where,

onSnapshot,

document,

getDocs,

orderBy,

limit,

} from "@firebase/firestore";

import { DataGrid, GridRowsProp, GridColDef, GridToolbar } from "@mui/x-data-grid";

import {

Typography,

Container,

Grid,

TextField,

Box,

FormControl,

InputLabel,

Select,

MenuItem,

Accordion,

AccordionSummary,

AccordionDetails,

ToggleButton,

ToggleButtonGroup,

FormGroup,

Checkbox,

FormControlLabel,

// checked,

// handleChange,

} from "@mui/material";

import { db } from "../../firebaseConfig/firebase";

import moment from "moment";

// const sgMail = require("@sendgrid/mail");

// sgMail.setApiKey("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");

// const msg = {

// to: "saifs252525@gmail.com",

// from: "advanceinsonline@gmail.com", // Use the email address or domain you verified above

// subject: "Sending with Twilio SendGrid is Fun",

// text: "and easy to do anywhere, even with Node.js",

// html: "<strong>and easy to do anywhere, even with Node.js</strong>",

// };

// //ES6

// sgMail.send(msg).then(

// () => {},

// (error) => {

// console.error(error);

// if (error.response) {

// console.error(error.response.body);

// }

// }

// );

const DashboardModule = () => {

const [vendorArray, setVendorArray] = useState([]);

const [lifeArray, setLifeArray] = useState([]);

const [medicareArray, setMedicareArray] = useState([]);

const [autoArray, setAutoArray] = useState([]);

const [dateSelected, setDateSelected] = useState();

const [lifeChecked, setLifeChecked] = useState(true);

const [medicareChecked, setMedicareChecked] = useState(false);

const [autoChecked, setAutoChecked] = useState(false);

useEffect(async () => {

const lifeObject = await getDocs(collection(db, "lifeClients"));

const autoObject = await getDocs(collection(db, "autoClients"));

const medicareObject = await getDocs(collection(db, "medicareClients"));

let lifeLocalArray = [],

autoLocalArray = [],

medicareLocalArray = [];

if (!lifeObject.empty) {

//("Clients Found!")

lifeObject.forEach((doc) => {

// doc.data() is never undefined for query doc snapshots

// //(doc.id, " => ", doc.data())

const id = doc.id;

const clientObj = {

id: doc.id,

fname: doc.data().fname,

lname: doc.data().lname,

gender: doc.data().gender,

phone: doc.data().phone,

height: doc.data().height,

weight: doc.data().weight,

isMarried: doc.data().is\_married,

coverageAmount: doc.data().coverage\_amount,

coverageTime: doc.data().coverage\_time,

hasCondition: doc.data().is\_health\_condition,

isTobacco: doc.data().is\_tobacco,

address: doc.data().address,

dob: doc.data().dob,

si: doc.data().stateInitial,

city: doc.data().city,

zip: doc.data().zip,

email: doc.data().email,

ip: doc.data().clientIP,

jid: doc.data().jornayaID,

createdAt: new Date(doc.data().createdAt.seconds \* 1000),

};

lifeLocalArray.push(clientObj);

});

setLifeArray(lifeLocalArray);

} else {

//("There are no Clients yet!")

}

const emptyArray = [];

const vendors = await getDocs(collection(db, "vendorApi"));

vendors.forEach((doc) => {

let obj = {

name: doc.data().name,

url: doc.data().url,

};

emptyArray.push(obj);

});

//function to post data from server using fetch with application/json and cors enabled

for (let v = 0; v <= emptyArray.length; v++) {

let resp = await postData(emptyArray[v]?.url, [{ Munem: 1 }]);

if (!typeof resp === "object") {

resp

.then((n) => n.json())

.then((resp) => {

resp = resp;

// console.log(resp );

})

.catch((n) => {

console.error(n);

});

}

console.log(resp);

}

}, []);

useEffect(()=>{

const q = query(collection(db, "autoClients"), orderBy("createdAt", "desc"), limit(100));

// onSnapshot(collection(db, "autoClients"), (snapshot) => {

onSnapshot(q, (snapshot) => {

snapshot.docChanges().forEach((change) => {

// //(

// "Time stamp is now " +

// new Date(change.doc.data().createdAt.seconds \* 1000).toLocaleString(

// "en-US",

// { timeZone: "US/Central" }

// )

// )

if (change.type === "added") {

const clientObj = {

id: change.doc.id,

secondDriver: change.doc.data().secondDriverAdded ? "Yes" : "No",

secondCar: change.doc.data().secondCarAdded ? "Yes" : "No",

fcmake: change.doc.data().firstVehicle.Make == null ? "N/A" : change.doc.data().firstVehicle.Make,

fcmodel: change.doc.data().firstVehicle.Model == null ? "N/A" : change.doc.data().firstVehicle.Model,

fcyear: change.doc.data().firstVehicle.Year == null ? "N/A" : change.doc.data().firstVehicle.Year,

scmake: change.doc.data().secondVehicle.Make == null ? "N/A" : change.doc.data().secondVehicle.Make,

scmodel: change.doc.data().secondVehicle.Model == null ? "N/A" : change.doc.data().secondVehicle.Model,

scyear: change.doc.data().secondVehicle.Year == null ? "N/A" : change.doc.data().secondVehicle.Year,

insured: change.doc.data().insured,

insuranceName: change.doc.data().insuranceName == null ? "N/A" : change.doc.data().insuranceName,

insuranceTime: change.doc.data().insuranceTime == null ? "N/A" : change.doc.data().insuranceName,

fdfname: change.doc.data().firstDriver.Fname == null ? "N/A" : change.doc.data().firstDriver.Fname,

fdlname: change.doc.data().firstDriver.Lname == null ? "N/A" : change.doc.data().firstDriver.Lname,

fdaccident: change.doc.data().firstDriver.AtFaultAccident == null ? "N/A" : change.doc.data().firstDriver.AtFaultAccident,

fddob: change.doc.data().firstDriver.DOB == null ? "N/A" : change.doc.data().firstDriver.DOB,

fdgender: change.doc.data().firstDriver.Gender == null ? "N/A" : change.doc.data().firstDriver.Gender,

fdmarried: change.doc.data().firstDriver.Married == null ? "N/A" : change.doc.data().firstDriver.Married,

fddui: change.doc.data().firstDriver.DUI == null ? "N/A" : change.doc.data().firstDriver.DUI,

sdfname: change.doc.data().secondDriver.Fname == null ? "N/A" : change.doc.data().secondDriver.Fname,

sdlname: change.doc.data().secondDriver.Lname == null ? "N/A" : change.doc.data().secondDriver.Lname,

sdaccident: change.doc.data().secondDriver.AtFaultAccident == null ? "N/A" : change.doc.data().secondDriver.AtFaultAccident,

sddob: change.doc.data().secondDriver.DOB == null ? "N/A" : change.doc.data().secondDriver.DOB,

sdgender: change.doc.data().secondDriver.Gender == null ? "N/A" : change.doc.data().secondDriver.Gender,

sdmarried: change.doc.data().secondDriver.Married == null ? "N/A" : change.doc.data().secondDriver.Married,

sddui: change.doc.data().secondDriver.DUI == null ? "N/A" : change.doc.data().secondDriver.DUI,

name: change.doc.data().name,

gender: change.doc.data().gender,

phone: change.doc.data().phone,

address: change.doc.data().address,

dob: change.doc.data().dob,

si: change.doc.data().stateInitial,

city: change.doc.data().city,

zip: change.doc.data().zip,

email: change.doc.data().email,

ip: change.doc.data().clientIP,

jid: change.doc.data().jornayaID,

createdAt:

new Date(change.doc.data().createdAt.seconds \* 1000).toLocaleString("en-US", { timeZone: "US/Central" }) + "\tUS/Central Timezone",

sourceUrl: "https://advancedinsonline.com/auto-insurance",

};

// console.log(clientObj,"Client Auto Object");

setAutoArray((oldArray) => [...oldArray, clientObj]);

}

});

});

},[])

useEffect(() => {

// loadDataFromDB()

onSnapshot(collection(db, "medicareClients"), (snapshot) => {

snapshot.docChanges().forEach((change) => {

if (change.type === "added") {

const clientObj = {

id: change.doc.id,

fname: change.doc.data().fname,

lname: change.doc.data().lname,

gender: change.doc.data().gender,

phone: change.doc.data().phone,

address: change.doc.data().address,

dob: change.doc.data().dob,

si: change.doc.data().stateInitial,

city: change.doc.data().city,

zip: change.doc.data().zip,

email: change.doc.data().email,

ip: change.doc.data().clientIP,

jid: change.doc.data().jornayaID,

createdAt:

new Date(change.doc.data().createdAt.seconds \* 1000).toLocaleString("en-US", { timeZone: "US/Central" }) + "\tUS/Central Timezone",

sourceUrl: "https://advancedinsonline.com/medicare",

};

setMedicareArray((oldArray) => [...oldArray, clientObj]);

}

});

});

}, []);

const columns = [

{ field: "id", headerName: "ID", width: 200 },

{ field: "fname", headerName: "First Name", width: 150 },

{ field: "lname", headerName: "Last Name", width: 150 },

{ field: "gender", headerName: "Gender", width: 150 },

{ field: "hasCondition", headerName: "Has Condition", width: 150 },

{ field: "coverageAmount", headerName: "Coverage Amount", width: 150 },

{ field: "coverageTime", headerName: "Coverage Time", width: 150 },

{ field: "isMarried", headerName: "Married", width: 150 },

{ field: "isTobacco", headerName: "Tobbaco Usage", width: 150 },

{ field: "height", headerName: "Height", width: 150 },

{ field: "weight", headerName: "Weight", width: 150 },

{ field: "phone", headerName: "Phone", width: 150 },

{ field: "address", headerName: "Address", width: 300 },

{ field: "dob", headerName: "DOB", width: 150 },

{ field: "si", headerName: "State", width: 150 },

{ field: "city", headerName: "City", width: 150 },

{ field: "zip", headerName: "ZIP", width: 150 },

{ field: "email", headerName: "Email", width: 250 },

{ field: "ip", headerName: "IP Address", width: 150 },

{ field: "jid", headerName: "Jornaya ID", width: 300 },

{ field: "createdAt", headerName: "Creation TS", minWidth: 300 },

{ field: "sourceUrl", headerName: "Source Url", minWidth: 300 },

];

//function to post data to server using fetch with application/json and cors enabled

let postData = function (endppoint, data) {

return fetch(endppoint, {

method: "POST",

body: JSON.stringify(data),

headers: {

"Content-Type": "application/json",

"Access-Control-Allow-Origin": "\*",

},

});

};

//function to post data from server using fetch with application/json and cors enabled

// let postDataForm = function (endppoint, data) {

// return fetch(endppoint, {

// method: "POST",

// body: data,

// headers: {

// "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8"

// }

// });

// }

// //function to post data to server using fetch with application/x-www-form-urlencoded and cors enabled

// //example usage

// Sanjeev Aassori R12:53 AM

// resp.then(n => n.json()).then(resp => {

// console.log(resp);

// }).catch(n => {

// console.error(n);

// });

// uzf-iams-phx

//example usage

//

//function to post data to server using fetch with application/x-www-form-urlencoded and cors enabled

// let postDataForm = (endppoint, data) => {

// return await fetch(endppoint, {

// ≥ method: "POST",

// body: data,

// headers: {

// "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8"

// }

// });

// }

const MedicareColumns= [

{ field: "id", headerName: "ID", width: 200 },

{ field: "fname", headerName: "First Name", width: 150, editable: true },

{ field: "lname", headerName: "Last Name", width: 150, editable: true },

{ field: "gender", headerName: "Gender", width: 150, editable: true },

{ field: "phone", headerName: "Phone", width: 150, editable: true },

{ field: "address", headerName: "Address", width: 300, editable: true },

{ field: "dob", headerName: "DOB", width: 150, editable: true },

{ field: "si", headerName: "State", width: 150, editable: true },

{ field: "city", headerName: "City", width: 150, editable: true },

{ field: "zip", headerName: "ZIP", width: 150, editable: true },

{ field: "email", headerName: "Email", width: 250, editable: true },

{ field: "ip", headerName: "IP Address", width: 150, editable: true },

{ field: "jid", headerName: "Jornaya ID", width: 300, editable: true },

{ field: "createdAt", headerName: "Creation TS", minWidth: 300 },

{ field: "sourceUrl", headerName: "Source Url", minWidth: 300 },

];

const AutoColumns = [

{ field: "id", headerName: "ID", minWidth: 150 },

{ field: "fdfname", headerName: "FD First Name", minWidth: 150 },

{ field: "fdlname", headerName: "FD Last Name", minWidth: 150 },

{ field: "fdaccident", headerName: "FD Accident", minWidth: 150 },

{ field: "fddob", headerName: "FD DOB", minWidth: 150 },

{ field: "fdgender", headerName: "FD Gender", minWidth: 150 },

{ field: "fdmarried", headerName: "FD Married", minWidth: 150 },

{ field: "fddui", headerName: "FD DUI", minWidth: 150 },

{ field: "fcmake", headerName: "FC Make", minWidth: 150 },

{ field: "fcmodel", headerName: "FC Model", minWidth: 150 },

{ field: "fcyear", headerName: "FC Year", minWidth: 150 },

{ field: "secondDriver", headerName: "Second Driver Added", minWidth: 150 },

{ field: "sdfname", headerName: "SD First Name", minWidth: 150 },

{ field: "sdlname", headerName: "SD Last Name", minWidth: 150 },

{ field: "sdaccident", headerName: "SD Accident", minWidth: 150 },

{ field: "sddob", headerName: "SD DOB", minWidth: 150 },

{ field: "sdgender", headerName: "SD Gender", minWidth: 150 },

{ field: "sdmarried", headerName: "SD Married", minWidth: 150 },

{ field: "sddui", headerName: "SD DUI", minWidth: 150 },

{ field: "secondCar", headerName: "Second Car Added", minWidth: 150 },

{ field: "scmake", headerName: "SC Make", minWidth: 150 },

{ field: "scmodel", headerName: "SC Model", minWidth: 150 },

{ field: "scyear", headerName: "SC Year", minWidth: 150 },

{ field: "insured", headerName: "Prev. Insured", minWidth: 150 },

{

field: "insuranceName",

headerName: "Prev. Insurance Name",

minWidth: 150,

},

{

field: "insuranceTime",

headerName: "Time with prev. Insurance",

minWidth: 150,

},

{ field: "phone", headerName: "Phone", minWidth: 150 },

{ field: "address", headerName: "Address", minWidth: 150 },

{ field: "si", headerName: "State", minWidth: 150 },

{ field: "city", headerName: "City", minWidth: 150 },

{ field: "zip", headerName: "ZIP", minWidth: 150 },

{ field: "email", headerName: "Email", minWidth: 150 },

{ field: "ip", headerName: "IP Address", minWidth: 150 },

{ field: "jid", headerName: "Jornaya ID", minWidth: 320 },

{ field: "createdAt", headerName: "Creation TS", minWidth: 300 },

{ field: "sourceUrl", headerName: "Source Url", minWidth: 300 },

];

console.log(autoArray,"Auto Array")

useEffect(() => {

if (dateSelected && lifeChecked) {

const filteredArray = lifeArray.filter((item) => moment(item.createdAt).format("DD MM YYYY") === moment(dateSelected).format("DD MM YYYY"));

setLifeArray(filteredArray);

}

if (dateSelected && autoChecked) {

const filteredArray = autoArray.filter((item) => moment(item.createdAt).format("DD MM YYYY") === moment(dateSelected).format("DD MM YYYY"));

setAutoArray(filteredArray);

}

if (dateSelected && medicareChecked) {

const filteredArray = medicareArray.filter((item) => moment(item.createdAt).format("DD MM YYYY") === moment(dateSelected).format("DD MM YYYY"));

setMedicareArray(filteredArray);

}

}, [dateSelected]);

return (

<div>

<h1> Dashboard Module </h1>

{/\* <a href="mailto:abc@example.com?subject = Feedback&body = Message">Send Feedback</a> \*/}

<Checkbox

checked={lifeChecked}

defaultChecked

onChange={(event) => {

setLifeChecked(event.target.checked);

setMedicareChecked(false);

setAutoChecked(false);

}}

inputProps={{ "aria-label": "controlled" }}

/>

Life

<Checkbox

checked={medicareChecked}

onChange={(event) => {

setLifeChecked(false);

setMedicareChecked(event.target.checked);

setAutoChecked(false);

}}

inputProps={{ "aria-label": "controlled" }}

/>

Medicare

<Checkbox

checked={autoChecked}

onChange={(event) => {

setLifeChecked(false);

setMedicareChecked(false);

setAutoChecked(event.target.checked);

}}

inputProps={{ "aria-label": "controlled" }}

/>

Auto

<Container>

<Box sx={styles.searchdiv}>

<TextField

InputProps={{ disableUnderline: true }}

inputMode="numeric"

type={"date"}

onChange={(event) => setDateSelected(event.target.value)}

sx={{

width: "100%",

// height: "100%",

"& .MuiInputBase-input": {

// height: "100%",

border: "none",

outline: "none",

},

"& .MuiFilledInput-input": {

// height: "100%",

backgroundColor: "white",

border: "none",

outline: "none",

},

// "&&&:before": {

// borderBottom: "none",

// outline: "none",

// },

// "&&:after": {

// borderBottom: "none",

// outline: "none",

// },

}}

id="filled-basic"

label="Type Date to search"

variant="filled"

/>

</Box>

</Container>

{lifeArray.length > 0 && lifeChecked && (

<div style={{ height: 650, width: "100%" }}>

<DataGrid rows={lifeArray} columns={columns} components={{ Toolbar: GridToolbar }} />

</div>

)}

{medicareArray.length > 0 && medicareChecked && (

<div style={{ height: 650, width: "100%" }}>

<DataGrid rows={medicareArray} columns={MedicareColumns} components={{ Toolbar: GridToolbar }} />

</div>

)}

{autoArray.length > 0 && autoChecked && (

<div style={{ height: 650, width: "100%" }}>

<DataGrid rows={autoArray} columns={AutoColumns} components={{ Toolbar: GridToolbar }} />

</div>

)}

</div>

);

};

const styles = {

gridContainer: {

display: "flex",

flexDirection: "column",

alignItems: "center",

justifyContent: "center",

},

generalContainer: {

display: "flex",

flexDirection: "column",

alignItems: "center",

justifyContent: "center",

},

summaryContainer: {

width: "100%",

display: "flex",

flexDirection: "row",

justifyContent: "center",

alignItems: "center",

},

summaryTitleContainer: {

width: "20%",

},

summaryDataContainer: {

width: "80%",

display: "flex",

flexDirection: "row",

justifyContent: "flex-start",

alignItems: "center",

},

generalBtn: {

width: "100%",

height: "100%",

maxHeight: 70,

padding: "15px 0px",

fontSize: { lg: 22, md: 18, sm: 18, xs: 15 },

fontWeight: "bold",

color: "white",

},

// generalBtnText: {

// fontSize: { lg: 22, md: 18, sm: 18, xs: 15 },

// fontWeight: "bold",

// color: "white",

// },

generalSingleBtn: {

width: "100%",

maxWidth: 400,

height: "100%",

maxHeight: 70,

padding: "15px 0px",

marginTop: 5,

fontSize: { lg: 22, md: 18, sm: 18, xs: 15 },

fontWeight: "bold",

},

headerStyle: {

fontSize: { lg: 48, md: 48, sm: 30, xs: 26 },

fontWeight: "bold",

textAlign: "center ",

marginBottom: "25px",

marginTop: "25px",

color: "#000e31",

},

searchdiv: {

width: "100%",

height: 65,

backgroundColor: "grey",

marginBottom: "20px",

boxShadow: 10,

},

subHeadingStyle: {

fontSize: 30,

textAlign: "center ",

},

};

export default DashboardModule;