```
import React, { useState, useEffect, useRef } from "react";
import { MapContainer, TileLayer, Marker, Popup } from "react-leaflet";
import "leaflet/dist/leaflet.css";
import L from "leaflet";
import initializeFirebase from "./firebase/firebase";
import { ref, get } from "firebase/database";
import Paper from "@mui/material/Paper";
import Table from "@mui/material/Table";
import TableBody from "@mui/material/TableBody";
import TableCell from "@mui/material/TableCell";
import TableContainer from "@mui/material/TableContainer";
import TableHead from "@mui/material/TableHead";
import TableRow from "@mui/material/TableRow";
import Grid from "@mui/material/Grid";
import Typography from "@mui/material/Typography";
import progressIcon from "../icons/progress-icon.svg";
import pendingIcon from "../icons/pending-icon.svg";
import norequestIcon from "../icons/noreq-icon.svg";

export default function DeviceLocation() {
  const [rows, setRows] = useState([]);
  const [center, setCenter] = useState([14.5798232, 120.98568789]);
  const mapRef = useRef();

  useEffect(() => {
    const database = initializeFirebase();
    const paramPath = "/GutterLocations";
    const paramRef = ref(database, paramPath);

    const fetchDataFromFirebase = async () => {
      try {
        const snapshot = await get(paramRef);
        const data = snapshot.val();
        if (data) {
          const gutterLocations = Object.entries(data)
            .map(([deviceId, deviceData]) => {
              const {
                name,
                address,
                latitude,
                longitude,
                maintenanceStatus,
                isClogged,
              } = deviceData;

              let clogStatus = "Cleared";
```

```
      if (isClogged) {
        const timestamps = Object.keys(isClogged);
        const latestTimestamp =
          timestamps.length > 0
            ? timestamps[timestamps.length - 1]
            : null;
        const latestStatus = isClogged[latestTimestamp];

        if (latestStatus !== undefined) {
          clogStatus = latestStatus ? "Clogged" : "Cleared";
        }
      }

      const lat = parseFloat(latitude);
      const lng = parseFloat(longitude);
      if (isNaN(lat) || isNaN(lng)) {
        console.error(
          `Invalid latitude or longitude for ${name}:`,
          latitude,
          longitude,
        );
        return null;
      }

      return {
        name,
        address,
        latitude: lat,
        longitude: lng,
        maintenanceStatus,
        clogStatus,
      };
    })
    .filter((entry) => entry !== null);

  setRows(gutterLocations);

  if (gutterLocations.length > 0) {
    const centerLocation = gutterLocations[0];
    setCenter([
      parseFloat(centerLocation.latitude),
      parseFloat(centerLocation.longitude),
    ]);
  }
} else {
```

```
        console.log("No data available under GutterLocations.");
       }
     } catch (error) {
       console.error("Error fetching data from Firebase:", error);
     }
   };

   fetchDataFromFirebase();

   return () => {};
  }, []);

  const columns = [
   { id: "name", label: "Name", minWidth: 170 },
   { id: "maintenanceStatus", label: "Maintenance Status", minWidth: 170 },
  ];

  const maintenanceStatusMapping = {
   pending: "Pending",
   nomaintenancereq: "No maintenance required",
   inprogress: "In progress",
  };

  return (
   <Grid container spacing={3}>
    <Grid item xs={8}>
     <MapContainer
      center={center}
      zoom={16}
      ref={mapRef}
      style={{ height: "525px", width: "100%" }}
      maxZoom={18}
      minZoom={14}
     >
      <TileLayer
       url="https://api.maptiler.com/maps/dataviz/256/{z}/{x}/{y}.png?
key=qKtzXYmOKKYYAxMzX6D4"
       attribution='&copy; <a href="https://www.maptiler.com/copyright/">MapTiler</a>
contributors'
      />
      {rows.map((row, index) => (
       <Marker
        key={index}
        position={[parseFloat(row.latitude), parseFloat(row.longitude)]}
        icon={
         row.maintenanceStatus === "inprogress"
```

```
            ? L.icon({
                iconUrl: progressIcon,
                iconSize: [30, 30],
              })
            : row.maintenanceStatus === "pending"
              ? L.icon({
                  iconUrl: pendingIcon,
                  iconSize: [30, 30],
                })
              : L.icon({
                  iconUrl: norequestIcon,
                  iconSize: [30, 30],
                })
        }
      >
        <Popup>
          <div>
            <h2>{row.name}</h2>
            <p>Address: {row.address}</p>
            <p>Clog Status: {row.clogStatus}</p>
            Maintenance Status:{" "}
            {maintenanceStatusMapping[row.maintenanceStatus]}{" "}
          </div>
        </Popup>
      </Marker>
    ))}
  </MapContainer>
</Grid>
<Grid item xs={4}>
  <Paper
    elevation={3}
    sx={{ maxHeight: "525px", overflow: "auto", padding: 1 }}
  >
    <Typography
      variant="h5"
      align="center"
      sx={{ fontWeight: "bold", marginBottom: 2, marginTop: 2 }}
    >
      Clogged Gutters
    </Typography>
    <TableContainer>
      <Table stickyHeader aria-label="sticky table">
        <TableHead>
          <TableRow>
            {columns.map((column) => (
              <TableCell
```

```
                key={column.id}
                align="center"
                style={{ minWidth: column.minWidth }}
              >
                {column.label}
              </TableCell>
            ))}
          </TableRow>
        </TableHead>
        <TableBody>
          {rows.map((row, index) => (
            <TableRow key={index} hover role="checkbox" tabIndex={-1}>
              {columns.map((column) => (
                <TableCell
                  key={column.id}
                  align="center"
                  style={{ minWidth: column.minWidth }}
                >
                  {column.id === "maintenanceStatus"
                    ? maintenanceStatusMapping[row[column.id]]
                    : row[column.id]}
                </TableCell>
              ))}
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </TableContainer>
  </Paper>
 </Grid>
</Grid>
);
}
```