```
import React, { useEffect, useState } from "react";
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 TablePagination from "@mui/material/TablePagination";
import TableRow from "@mui/material/TableRow";
import TextField from "@mui/material/TextField";
import Button from "@mui/material/Button";
import Grid from "@mui/material/Grid";
import Typography from "@mui/material/Typography";
import { MenuItem } from "@mui/material";
import { useTheme } from "@emotion/react";
import { tokens } from "../theme";
import initializeFirebase from "../data/firebase/firebase";
import { ref, get, update } from "firebase/database";

const columns = [
  {
    id: "id",
    label: "ID",
    align: "center",
    minWidth: 100,
  },
  {
    id: "name",
    label: "Name",
    align: "center",
    minWidth: 100,
  },
  {
    id: "address",
    label: "Address",
    minWidth: 100,
    align: "center",
  },
  {
    id: "latitude",
    label: "Latitude",
    minWidth: 100,
    align: "center",
  },
  {
    id: "longitude",
```

```
    label: "Longitude",
    minWidth: 100,
    align: "center",
  },
];

function createData(id, name, address, latitude, longitude) {
  return { id, name, address, latitude, longitude };
}

export default function DeviceManagement() {
  const theme = useTheme();
  const colors = tokens(theme.palette.mode);
  const [page, setPage] = useState(0);
  const [rowsPerPage, setRowsPerPage] = useState(8);
  const [selectedDeviceId, setSelectedDeviceId] = useState("");
  const [nameInput, setNameInput] = useState("");
  const [addressInput, setAddressInput] = useState("");
  const [latitudeInput, setLatitudeInput] = useState("");
  const [longitudeInput, setLongitudeInput] = useState("");
  const [rows, setRows] = useState([]);
  const [idOptions, setIdOptions] = useState([]);
  const [isButtonClicked, setIsButtonClicked] = useState(false);
  const database = initializeFirebase();

  useEffect(() => {
    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(
            ([id, { name, address, latitude, longitude }]) =>
              createData(id, name, address, latitude, longitude),
          );
          setRows(gutterLocations);
          setIdOptions(gutterLocations.map((location) => location.id));
        } else {
          console.log("No data available under GutterLocations.");
        }
      } catch (error) {
        console.error("Error fetching data from Firebase:", error);
      }
```

```
  };

  fetchDataFromFirebase();

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

const handleChangePage = (event, newPage) => {
  setPage(newPage);
};

const handleChangeRowsPerPage = (event) => {
  setRowsPerPage(+event.target.value);
  setPage(0);
};

const handleAddDevice = async () => {
  setIsButtonClicked(true);
  if (
    selectedDeviceId &&
    nameInput &&
    addressInput &&
    latitudeInput &&
    longitudeInput
  ) {
    const rowIndex = rows.findIndex((row) => row.id === selectedDeviceId);
    if (rowIndex !== -1) {
      const updatedRow = rows[rowIndex];
      updatedRow.name = nameInput;
      updatedRow.address = addressInput;
      updatedRow.latitude = parseFloat(latitudeInput);
      updatedRow.longitude = parseFloat(longitudeInput);
      try {
        const snapshot = await get(
          ref(database, `/GutterLocations/${selectedDeviceId}`),
        );
        const existingData = snapshot.val();
        if (existingData) {
          const updatedData = {
            ...existingData,
            name: nameInput,
            address: addressInput,
            latitude: parseFloat(latitudeInput),
            longitude: parseFloat(longitudeInput),
          };
          await update(
```

```
            ref(database, `/GutterLocations/${selectedDeviceId}`),
            updatedData,
          );
          const newRows = [...rows];
          newRows[rowIndex] = updatedRow;
          setRows(newRows);
          setNameInput("");
          setAddressInput("");
          setLatitudeInput("");
          setLongitudeInput("");
          setSelectedDeviceId("");
          setIsButtonClicked(false);
          console.log("Device updated successfully!");
        } else {
          console.log("No data found for the selected device ID.");
        }
      } catch (error) {
        console.error("Error updating device:", error);
      }
    } else {
      console.log("Row index not found.");
    }
  } else {
    console.log("Please fill all required fields.");
  }
};

return (
  <Grid container spacing={3}>
    <Grid item xs={8}>
      <Paper sx={{ width: "100%", overflow: "hidden" }}>
        <TableContainer sx={{ maxHeight: 400 }}>
          <Table stickyHeader aria-label="sticky table">
            <TableHead>
              <TableRow>
                {columns.map((column) => (
                  <TableCell
                    key={column.id}
                    align={column.align}
                    style={{ minWidth: column.minWidth }}
                  >
                    {column.label}
                  </TableCell>
                ))}
              </TableRow>
            </TableHead>
```

```
        <TableBody>
          {rows
            .slice(page * rowsPerPage, page * rowsPerPage + rowsPerPage)
            .map((row) => (
              <TableRow hover role="checkbox" tabIndex={-1} key={row.id}>
                {columns.map((column) => (
                  <TableCell key={column.id} align={column.align}>
                    {row[column.id]}
                  </TableCell>
                ))}
              </TableRow>
            ))}
        </TableBody>
      </Table>
    </TableContainer>
    <TablePagination
      rowsPerPageOptions={[8]}
      component="div"
      count={rows.length}
      rowsPerPage={rowsPerPage}
      page={page}
      onPageChange={handleChangePage}
      onRowsPerPageChange={handleChangeRowsPerPage}
    />
  </Paper>
</Grid>
<Grid item xs={4}>
  <Paper sx={{ padding: 2 }}>
    <Typography variant="h4" sx={{ fontWeight: "bold", marginBottom: 2 }}>
      Configure Device
    </Typography>
    <TextField
      select
      label="ID"
      variant="filled"
      fullWidth
      InputLabelProps={{
        style: {
          color: colors.primary[200],
        },
      }}
      SelectProps={{
        MenuProps: {
          PaperProps: {
            style: {
              backgroundColor: colors.primary[900],
```

```
        },
      },
    },
  }}
  sx={{ marginBottom: 2 }}
  value={selectedDeviceId}
  onChange={(e) => setSelectedDeviceId(e.target.value)}
  required
  error={isButtonClicked && !selectedDeviceId}
  helperText={
    isButtonClicked && !selectedDeviceId && "ID is required"
  }
>
  {idOptions.map((option) => (
    <MenuItem
      key={option}
      value={option}
      style={{ color: colors.primary[100] }}
    >
      {option}
    </MenuItem>
  ))}
</TextField>
<TextField
  label="Name"
  variant="filled"
  fullWidth
  InputLabelProps={{
    style: {
      color: colors.primary[200],
    },
  }}
  sx={{ marginBottom: 2 }}
  value={nameInput}
  onChange={(e) => setNameInput(e.target.value)}
  required
  error={isButtonClicked && !nameInput}
  helperText={isButtonClicked && !nameInput && "Name is required"}
/>
<TextField
  label="Address"
  variant="filled"
  fullWidth
  InputLabelProps={{
    style: {
      color: colors.primary[200],
```

```
    },
  }}
  sx={{ marginBottom: 2 }}
  value={addressInput}
  onChange={(e) => setAddressInput(e.target.value)}
  required
  error={isButtonClicked && !addressInput}
  helperText={
    isButtonClicked && !addressInput && "Address is required"
  }
/>
<TextField
  label="Latitude"
  variant="filled"
  fullWidth
  InputLabelProps={{
    style: {
      color: colors.primary[200],
    },
  }}
  sx={{ marginBottom: 2 }}
  value={latitudeInput}
  onChange={(e) => setLatitudeInput(e.target.value)}
  required
  error={isButtonClicked && !latitudeInput}
  helperText={
    isButtonClicked && !latitudeInput && "Latitude is required"
  }
/>
<TextField
  label="Longitude"
  variant="filled"
  fullWidth
  InputLabelProps={{
    style: {
      color: colors.primary[200],
    },
  }}
  sx={{ marginBottom: 2 }}
  value={longitudeInput}
  onChange={(e) => setLongitudeInput(e.target.value)}
  required
  error={isButtonClicked && !longitudeInput}
  helperText={
    isButtonClicked && !longitudeInput && "Longitude is required"
  }
```

```
          />
          <Button
            variant="contained"
            sx={{
              backgroundColor: colors.orangeAccent[400],
              color: colors.primary[900],
              "&:hover": {
                backgroundColor: colors.orangeAccent[300],
              },
            }}
            onClick={handleAddDevice}
            fullWidth
          >
            Update Device
          </Button>
        </Paper>
      </Grid>
    </Grid>
  );
}
```