```
import React, { useState, useEffect, useCallback } from "react";
import { useDispatch, useSelector } from "react-redux";
import { Spin, Checkbox, Form } from "antd";
import ActionButtons from "components/ActionButtons";
import EditableCell from "components/EditableCell";
import MediTable from "components/Custom";

import {
  signPositionSearchGrid,
  createSignPosition,
  updateSignPosition,
  deleteSignPosition,
} from "store/slices/signatureProcessSlice";

const SignPostion = () => {
  const dispatch = useDispatch();
  const [form] = Form.useForm();
  const [editingKey, setEditingKey] = useState("");

  const { loading, signPositionList } = useSelector(
    (state) => state.signatureProcess
  );

  useEffect(() => {
    dispatch(signPositionSearchGrid());
  }, [dispatch]);

  const isEditing = useCallback(
    (record) =>
      record.action === "initial"
        ? record.action === editingKey
        : record.id === editingKey,
    [editingKey]
  );

  const edit = useCallback(
    (record) => {
      form.setFieldsValue({
        name: "",
        description: "",
        displayOrder: 1,
        inActive: false,
        ...record,
      });
      setEditingKey(record.action === "initial" ? record.action : record.id);
    },
    [form]
  );

  const cancel = useCallback(() => {
    setEditingKey("");
  }, []);

  const save = useCallback(
    async (key) => {
      try {
        const row = await form.validateFields();
        const newData = [...signPositionList];
        const index = newData.findIndex((item) =>
          item.id ? item.id === key : item.action === key
        );
        const item = newData[index];

        const payload = {
          id: item?.id || null,
          ...row,
          onSuccess: () => {
            setEditingKey("");
            reloadData();
          },
        };
```

```
      if (key === "initial") {
        dispatch(createSignPosition(payload));
      } else {
        dispatch(updateSignPosition(payload));
      }
    } catch (error) {
      console.log(error);
    }
  },
  // eslint-disable-next-line react-hooks/exhaustive-deps
  [signPositionList, dispatch, form]
);

const reloadData = () => {
  dispatch(signPositionSearchGrid());
}

const handleDelete = useCallback(
  (id) => {
    const payload = {
      id,
      onSuccess: () => {
          reloadData();
      },
    };
    dispatch(deleteSignPosition(payload));
  },
  // eslint-disable-next-line react-hooks/exhaustive-deps
  [dispatch]
);

const tableColumns = [
  {
    title: "STT",
    dataIndex: "index",
    align: "center",
    render: (_, __, index) => index + 1,
  },
  {
    title: "Tên",
    dataIndex: "name",
    key: "name",
    isRequired: true,
    editable: true,
  },
  {
    title: "Mô tả",
    dataIndex: "description",
    key: "description",
    editable: true,
  },
  {
    title: "Thứ tự hiển thị",
    align: "center",
    dataIndex: "displayOrder",
    editable: true,
  },
  {
    title: "Ngừng sử dụng",
    align: "center",
    dataIndex: "inActive",
    editable: true,
    render: (value) => <Checkbox checked={value} />,
  },
  {
    fixed: "right",
    align: "center",
    render: (_, record) => {
      const editable = isEditing(record);
      return (
        <ActionButtons
          record={record}
          editable={editable}
```

```
            save={save}
            cancel={cancel}
            edit={edit}
            handleDelete={handleDelete}
          />
        );
      },
    },
  ];

  const mergedColumns = tableColumns.map((col) => {
    if (!col.editable) {
      return col;
    }

    let inputType = "text"; // Default to text
    if (col.dataIndex === "inActive") {
      inputType = "checkbox";
    } else if (col.dataIndex === "displayOrder") {
      inputType = "number";
    }

    return {
      ...col,
      onCell: (record) => ({
        record,
        inputType: inputType,
        dataIndex: col.dataIndex,
        title: col.title,
        isRequired: col?.isRequired,
        editing: isEditing(record),
        form: form,
      }),
    };
  });

  return (
    <Spin tip="Đang tải..." spinning={loading}>
      <Form form={form} component={false}>
        <MediTable
          dataSource={[{ action: "initial" }].concat(signPositionList)}
          tableColumns={mergedColumns}
          // onRenderFooter={onRenderFooter}
          components={{
            body: {
              cell: EditableCell,
            },
          }}
          rowKeyField={(item) => item.id}
        />
      </Form>
    </Spin>
  );
};

export default SignPostion;
```