```
import React, { useState, useEffect, useCallback } from "react";
import { useDispatch, useSelector } from "react-redux";
import { Input, Result, Spin, Form, Typography } from "antd";
import SlipButton from "components/SlipButton";
import { PlusOutlined } from "@ant-design/icons";
import {
  getByObjectId,
  updateAttachment,
  deleteAttachment,
} from "store/slices/common/attachmentSlide";
import MediTable from "components/Custom";
import EditableCell from "components/EditableCell";
import ActionButtons from "components/ActionButtons";
import ModalUpload from "./ModalUpload.js";
import { MediActionDownload } from "components/Custom";
import Utils from "utils";
import { isMobile } from "mobile-device-detect";
import { isImageURL } from "utils/helper";
const { Search } = Input;

const CommonAttachment = ({ record, objectId, objectType, selectedRow }) => {
  console.log(selectedRow, "selectedRow");
  const dispatch = useDispatch();
  const [form] = Form.useForm();
  const [visibleUploadModel, setVisibleUploadModal] = useState(false);
  const [listAttachment, setListAttachment] = useState([]);
  const [originalAttachmentList, setOriginalAttachmentList] = useState([]); // Thêm state này
  const { loading, attachmentList } = useSelector((state) => state.attachment);
  const [editingKey, setEditingKey] = useState("");
  // console.log(objectId, "objectIdobjectId");
  useEffect(() => {
    reload();
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [objectId]);
  useEffect(() => {
    setOriginalAttachmentList([...attachmentList]);
    setListAttachment([...attachmentList]);
  }, [attachmentList]);
  const reload = () => {
    if (objectId) {
      const payload = {
        objectId: objectId,
        objectType: objectType,
      };
      dispatch(getByObjectId(payload));
    }
  };

  const handleClickAdd = () => {
    setVisibleUploadModal(true);
  };

  const onhandleSearch = (value) => {
    const filteredList = originalAttachmentList.filter(
      (item) =>
        item.description.toLowerCase().includes(value.toLowerCase()) ||
        item.fileName.toLowerCase().includes(value.toLowerCase())
    );
    setListAttachment(filteredList);
  };

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

  const edit = useCallback(
    (record) => {
```

```
    form.setFieldsValue({
      description: "aaaa",
      ...record,
    });
    setEditingKey(record.action === "initial" ? record.action : record.id);
    console.log("edit", record);
  },
  [form]
);

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

const save = useCallback(
  async (key) => {
    try {
      const row = await form.validateFields();
      const newData = [...listAttachment];
      const index = newData.findIndex((item) =>
        item.id ? item.id === key : item.action === key
      );
      const item = newData[index];
      const payload = {
        id: item.id || null,
        description: row.description,
      };
      const data = {
        payload: payload,
        onSuccess: () => {
          reload();
          setEditingKey("");
        },
      };
      dispatch(updateAttachment(data));
    } catch (error) {}
  },
  // eslint-disable-next-line react-hooks/exhaustive-deps
  [listAttachment, dispatch, form]
);
const handleDelete = useCallback(
  (id) => {
    const data = {
      id: id,
      onSuccess: () => {
        reload();
      },
    };
    dispatch(deleteAttachment(data));
  },
  // eslint-disable-next-line react-hooks/exhaustive-deps
  [dispatch]
);
const getHrefFromLink = (link) => {
  if (isImageURL(link)) return link;
  return `https://docs.google.com/viewerng/viewer?url=${link}`;
};
const columns = [
  {
    title: "STT",
    dataIndex: "index",
    align: "center",
    fixed: !isMobile ? "left" : false,
    render: (_, __, index) => index + 1,
    width: 50,
  },
  {
    title: "Nội dung",
    dataIndex: "description",
    align: "left",
    editable: true,
    key: "description",
    // width: 00,
  },
```

```
{
  title: "Tên file",
  dataIndex: "fileName",
  align: "left",
  width: 220,
  render: (text, record) => (
    <a
      href={getHrefFromLink(record.fileUrl)}
      target="_blank"
      style={{ display: "flex", alignContent: "center" }}
      rel="noreferrer"
    >
      {
        <div
          style={{
            display: "flex",
            alignItems: "center",
            marginRight: "8px",
          }}
        >
          {Utils.GetFileIcon(record.fileType)}
        </div>
      }{" "}
      <div
        style={{
          width: "200px",
          overflow: "hidden",
          textOverflow: "ellipsis",
          whiteSpace: "nowrap",
        }}
      >
        {text}
      </div>
    </a>
  ),
},
{
  title: "Người tải lên",
  dataIndex: "createdByName",
  width: 150,
  align: "left",
},
{
  title: "Ngày tải lên",
  dataIndex: "createdTime",
  width: 150,
  align: "center",
  render: (value) => Utils.formatDate(value),
},
{
  title: "Loại file",
  dataIndex: "fileType",
  align: "center",
  width: 75,
},
{
  title: "Dung lượng",
  dataIndex: "fileSizeKb",
  align: "center",
  width: 100,
  render: (value) => `${value} KB`,
},
{
  fixed: !isMobile ? "right" : false,
  align: "center",
  render: (_, record) => {
    const editable = isEditing(record);
    return (
      <ActionButtons
        record={record}
        editable={editable}
        save={save}
        cancel={cancel}
        edit={edit}
```

```
          handleDelete={handleDelete}
          disabled={handleDisableButton()}
          extraButtons={() => {
            return (
              <Typography.Link
                href={record.fileUrl?.replace("http:", "https:")}
                target="_blank"
              >
                <MediActionDownload
                  className="mr-2"
                  // shape=""
                />
              </Typography.Link>
            );
          }}
        />
      );
    },
  },
];

const mergedColumns = columns.map((col) => {
  if (!col.editable) {
    return col;
  }
  return {
    ...col,
    onCell: (record) => ({
      record,
      title: col.title,
      dataIndex: col.dataIndex,
      editing: isEditing(record),
    }),
  };
});

const onSubmitSuccessHandler = () => {
  // reload data
  reload();
  setVisibleUploadModal(false);
};

const handleDisableButton = useCallback(() => {
  if (selectedRow) {
    return (
      selectedRow?.totalSigned > 1 ||
      (selectedRow?.totalSigned !== null &&
        selectedRow?.totalSigns !== null &&
        selectedRow?.totalSigned === selectedRow?.totalSigns)
    );
  } else {
    return false;
  }
}, [selectedRow]);
return (
  <>
    {objectId ? (
      <Spin tip="Đang tải..." spinning={loading}>
        <div className="custom-formdetail">
          <div className="flex-space">
            <div
              style={{ display: "flex", alignItems: "center", width: 200 }}
            >
              <Search
                placeholder="Tìm tên file, nội dung"
                style={{ width: "100%" }}
                onSearch={onhandleSearch}
              />
            </div>
            <SlipButton
              type="primary"
              size="middle"
              text="Thêm"
              icon={PlusOutlined}
```

```
                onClick={handleClickAdd}
                disabled={handleDisableButton()}
              ></SlipButton>
            </div>
            <ModalUpload
              objectId={objectId}
              objectType={objectType}
              isVisibleModal={visibleUploadModel}
              onCancelModal={() => {
                setVisibleUploadModal(false);
              }}
              onSubmitSuccess={onSubmitSuccessHandler}
            />
            <Form form={form} component={false}>
              <MediTable
                dataSource={listAttachment}
                tableColumns={mergedColumns}
                // onRenderFooter={onRenderFooter}
                rowKeyField={(item) => item.id}
                components={{
                  body: {
                    cell: EditableCell,
                  },
                }}
              />
            </Form>
          </div>
        </Spin>
      ) : (
        <Result
          status="warning"
          title="Xin vui lòng tạo mới phiếu trước khi thao tác."
        />
      )}
    </>
  );
};

export default CommonAttachment;
```