```
import React, {
  useEffect,
  useMemo,
  useCallback,
  useState,
  memo,
  forwardRef,
  useImperativeHandle,
} from "react";
import { useDispatch, useSelector } from "react-redux";
import { useSearchParams } from "react-router-dom";
import { Modal, Row, Col, Card, Table, Tabs, Form, Input, Button } from "antd";
import CommonAttachment from "../CommonAttachment";
import Expandable from "components/SignProcess/Expandable";
import SignProcess from "components/SignProcess/SignProcess";
import {
  createSignedDocument,
  signReject,
  signRemind,
} from "store/slices/signatureProcessSlice";
import { updateSignatureProcess } from "store/slices/internalSlice";
import { notification } from "antd";
import { LinkOutlined } from "@ant-design/icons";
import CopyToClipboard from "react-copy-to-clipboard";
import { DETAIL_PAGE_PATH } from "configs/AppConfig";
import { DOCUMENT_TYPE_COMP } from "constants/index";

const { confirm } = Modal;
const SlipModalComponent = forwardRef(
  (
    {
      title,
      isVisibleModal,
      onCancelModal,
      onRenderInfo,
      widthInfo,
      onRenderTab,
      objectId = "",
      objectType = "",
      titleTLink = "",
      descriptionTLink = "",
      documentTypeId = null,
      activeKey,
      onSignReloadForm,
      selectedRow,
    },
    ref
  ) => {
    const [activeTabKey, setActiveTabKey] = useState("998");
    const items = useMemo(() => {
      const tabs = [];
      let key = 3;
      if (onRenderTab) {
        onRenderTab.forEach((item) => {
          tabs.push({
            key: ++key,
            label: item?.label,
            children: item?.children,
          });
        });
      }
      tabs.push(
        {
          key: "1",
          label: "File đính kèm",
          children: (
            <CommonAttachment
              objectId={objectId}
              objectType={objectType}
              selectedRow={selectedRow}
            />
          ),
        },
        {
          key: "2",
          label: "Lịch sử thay đổi",
          children: <Table />,
        }
      );
      return tabs;
    }, [objectId, objectType, onRenderTab]);

    const dispatch = useDispatch();
    const [signData, setSignData] = useState({});
    const [forceUpdate, setForceUpdate] = useState(false);
    const { branchId } = useSelector((state) => state.auth);
    const [searchParams] = useSearchParams();
    const onSetSignData = (data = {}) => {
      setSignData({ ...data });
    };

    const [visibleModal, setVisibleModal] = useState(true);
    const [form] = Form.useForm();

    useImperativeHandle(ref, () => ({
      signedDocumentCreate: (id, code, description = "") => {
        const payload = {
          documentId: id,
          documentCode: code,
          documentTypeId: documentTypeId,
          branchId: branchId,
          description: description,
          onSuccess: ({ data }) => {
            setForceUpdate((prev) => !prev);

            const payloadTotal = {
              documentType: documentTypeId,
              objectId: id,
```

```
          totalSigns: data.totalSigns,
          totalSigned: data.totalSigned,
        };

        dispatch(updateSignatureProcess(payloadTotal));
      },
    };
    console.log("signedDocumentCreate", payload);
    dispatch(createSignedDocument(payload));
  },
}));

useEffect(() => {
  if (items.length > 0) {
    setActiveTabKey(items[0].key);
  }
}, [items]);

const renderContent = useCallback(() => {
  return (
    <Row gutter={[16, 16]}>
      <Col
        xs={24} // Chiếm toàn bộ chiều rộng khi màn hình dưới md
        lg={8}
        md={12}
      >
        <Card className="custom-card">
          {onRenderInfo && onRenderInfo()}
        </Card>
      </Col>
      <Col
        xs={24} // Chiếm toàn bộ chiều rộng khi màn hình dưới md
        lg={16}
        md={12}
      >
        <Card className="custom-card">
          <Tabs
            activeKey={activeTabKey}
            items={items}
            onChange={setActiveTabKey}
          />
        </Card>
      </Col>
    </Row>
  );
}, [onRenderInfo, items, activeTabKey]);

const onRejectHandle = (signName) => {
  confirm({
    icon: <></>,
    open: visibleModal,
    content: (
      <>
        <div className="text-center">
          <div style={{ fontSize: 16, fontWeight: 500, marginBottom: 12 }}>
            Bạn muốn từ chối ký?
          </div>
          <div style={{ marginBottom: 12 }}>
            Vui lòng xác nhận bạn muốn từ chối ký vị trí{" "}
            <span style={{ fontWeight: 500 }}>{signName}</span>. Sau khi từ
            chối ký bạn{" "}
            <span style={{ fontWeight: 500 }}>không thể ký lại</span> vị trí
            này nữa
          </div>
        </div>
        <Form form={form} layout="vertical" onFinish={onFinishHandle}>
          <Form.Item
            name="reason"
            rules={[
              {
                required: true,
                message: "Lý do từ chối ký là bắt buộc",
              },
            ]}
          >
            <Input.TextArea placeholder="Lý do từ chối ký" />
          </Form.Item>
        </Form>
      </>
    ),
    onOk() {
      return form
        .validateFields()
        .then((values) => {
          onFinishHandle(values);
        })
        .catch((errorInfo) => {
          return Promise.reject();
        });
    },
    // onCancel() {
    //     console.log('Cancel');
    // },
  });
};

const renderLink = () => {
  if (documentTypeId === null) {
    return;
  }

  const compDetail = DOCUMENT_TYPE_COMP.find(
    (item) => item.id === documentTypeId
  );

  if (!compDetail) {
    notification.error({
      message: "Không tìm thấy loại phiếu!",
      duration: 3,
    });
    return "";
  }
```

```
    const link = `[MEDIPHA ERP - ${titleTLink}] - ${descriptionTLink}
${window.location.protocol}//${window.location.host}${DETAIL_PAGE_PATH}?page=${compDetail.comp}&${compDetail.comp}Id=${objectId}`;

    return link;
  };

  const handleCodeCopied = () => {
    notification.success({
      message: "Sao chép thành công!",
      duration: 1,
    });
  };

  const onFinishHandle = (values) => {
    const payload = {
      documentId: objectId,
      documentTypeId: documentTypeId,
      reason: values.reason,

      onSuccess: ({ data }) => {
        console.log(visibleModal, "prev");
        setVisibleModal((prev) => !prev);
        console.log(visibleModal, "form");
        setForceUpdate((prev) => !prev);
        form.resetFields();

        const payloadTotal = {
          documentType: documentTypeId,
          objectId: objectId,
          totalSigns: data.totalSigns,
          totalSigned: data.totalSigned,
          onSuccess: () => {
            if (onSignReloadForm) onSignReloadForm(data);
          },
        };

        dispatch(updateSignatureProcess(payloadTotal));
      },
    };

    dispatch(signReject(payload));
  };

  const onRemindHandle = () => {
    const payload = {
      documentId: objectId,
      documentTypeId: documentTypeId,
      // onSuccess: () => {},
    };

    dispatch(signRemind(payload));
  };
  const onBackToParentPage = (pageName) => {
    let homePage = `${window.location.protocol}//${window.location.host}`;
    // todo
    if (
      pageName === "VanBanNoiBoDi" ||
      pageName === "QuanLyCongViec" ||
      pageName === "QuanLyCongVanBenNgoaiDen" ||
      pageName === "QuanLyCongVanBenNgoaiDi" ||
      pageName === "HoSoTaiLieu" ||
      pageName === "PhieuQuyChe" ||
      pageName === "PhieuVanBan"
    ) {
      homePage = `${homePage}/app/common/tong-hop-ky`;
    }
    window.location.href = homePage;
  };
  const renderSignProcess = () => {
    // return (<SignFooter documentTypeId={documentTypeId} documentId={objectId} />);
    return (
      <div style={{ overflow: "scroll" }}>
        {" "}
        <div className="sign-container">
          <Expandable
            signData={signData}
            onRejectHandle={onRejectHandle}
            onRemindHandle={onRemindHandle}
          >
            {documentTypeId && (
              <SignProcess
                className="mb-2"
                documentTypeId={documentTypeId}
                documentId={objectId}
                onSetSignData={onSetSignData}
                forceUpdate={forceUpdate}
                onSignReloadForm={onSignReloadForm}
              />
            )}
          </Expandable>
          {/* <Modal
      closable={false}
      width={383}
      open={visibleModal}
      onCancel={() => {
        setVisibleModal(!visibleModal);
        form.resetFields();
      }}
      onOk={() => form.submit()}
      okText="Xác nhận"
      cancelText="Hủy"
    >
      <div className="text-center">
        <div style={{ fontSize: 16, fontWeight: 500, marginBottom: 12 }}>Bạn muốn từ chối ký?</div>
        <div style={{ marginBottom: 12 }}>Vui lòng xác nhận bạn muốn từ chối ký vị trí <span style={{ fontWeight: 500 }}>Thủ kho</span>. Sau khi từ chối ký bạn <spa
      </div>
      <Form form={form} layout="vertical" onFinish={onFinishHandle}>
        <Form.Item
          name="reason"
          rules={[
            {
```

```
                required: true,
                message: "Lý do từ chối ký là bắt buộc",
              },
            ]}
          >
            <Input.TextArea
              placeholder="Lý do từ chối ký"
            />
          </Form.Item>
        </Form>
      </Modal> */}
        </div>
      </div>
    );
  };

  return (
    <Modal
      title={
        <div className="d-flex align-items-center">
          {title || "Chi tiết"}{" "}
          {titleTLink && (
            <CopyToClipboard
              className="ml-auto mr-5"
              text={renderLink()}
              onCopy={handleCodeCopied}
            >
              <Button size="small" type="link" icon={<LinkOutlined />}>
                Sao chép link
              </Button>
            </CopyToClipboard>
          )}
        </div>
      }
      open={isVisibleModal}
      onCancel={() => {
        setForceUpdate((prev) => !prev);
        const detailPage = searchParams.get("page");
        if (detailPage) {
          // Back to parent page
          onBackToParentPage(detailPage);
        } else {
          onCancelModal();
        }
      }}
      footer={renderSignProcess()}
      width={1660}
      height={"80vh"}
      className="sign-modal"
      // bodyStyle={{ overflowY: 'scroll' }}
      maskClosable={false}
    >
      {renderContent()}
    </Modal>
  );
 }
);

export default memo(SlipModalComponent);
```