```
import React, { memo, useState, useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import {
  getSignedDocument,
  updateSignedDocument,
} from "store/slices/signatureProcessSlice";
import { getAllEmployee } from "store/slices/employeeSlice";
import { Modal } from "antd";
import Sign from "./Sign";
import { isEmpty } from "lodash";
import { SIGNATURE_STATUS } from "constants";
import ModalEmployeeSignProcess from "views/app-views/hrm/employee/components/ModalEmployeeSignProcess";
import { updateSignatureProcess } from "store/slices/internalSlice";
const { confirm } = Modal;

const SignProcess = ({
  className,
  documentTypeId,
  documentId,
  onAllSigned,
  onOneSigned,
  onSetSignData,
  onValidBeforeSign, // return true/false before Sign
  forceUpdate,
  onSignReloadForm,
}) => {
  const dispatch = useDispatch();
  const [signProcess, setSignProcess] = useState();
  const [selectedSign, setSelectedSign] = useState();
  const [employeeId, setEmployeeId] = useState();
  const [employeeSignList, setEmployeeSignList] = useState();
  const [visibleEmployee, setVisibleEmployee] = useState(false);

  const { branchId } = useSelector((state) => state.auth);
  // console.log(documentId,"documentId")
  useEffect(() => {
    dispatch(getAllEmployee({ SearchText: null, branchId }));
  }, [dispatch, branchId]);

  useEffect(() => {
    if (documentTypeId > 0) {
      reloadData();
    }
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [documentTypeId, forceUpdate, documentId]);

  const reloadData = () => {
    const payload = {
      documentTypeId: documentTypeId,
      documentId: !isEmpty(documentId) ? documentId : null,
      branchId: branchId,
      onSuccess: (result) => {
        onSetSignData(result);
        setSignProcess(result);
      },
    };

    dispatch(getSignedDocument(payload));
  };

  const onSignClick = (item) => () => {
    if (!item.allowSign || documentTypeId == null || !item.isNextSign) return;

    // Phải ký theo thứ tự
    if (!item.isNextSign) return;

    // show confirmation dialog
    confirm({
      icon: <></>,
      centered: true,
      content: (
```

```
      <div style={{ fontWeight: 500 }}>
        Bạn có chắc chắn muốn ký tại vị trí này không? <br />
        Xác nhận ký
      </div>
    ),
    onOk() {
      const payload = {
        signedDocumentId: item.id,
        signed: true,
        onSuccess: ({ data }) => {
          const payloadTotal = {
            documentType: documentTypeId,
            objectId: documentId,
            totalSigns: data.totalSigns,
            totalSigned: data.totalSigned,
          };

          dispatch(updateSignatureProcess(payloadTotal));
          reloadData();
          if (onSignReloadForm) onSignReloadForm(data);
        },
      };

      dispatch(updateSignedDocument(payload));
    },
    // onCancel() {
    //     console.log('Cancel');
    // },
  });
};

const onChangeSigner = (item) => () => {
  if (documentTypeId == null || item.signed === SIGNATURE_STATUS.Signed)
    return;

  setSelectedSign(item);

  setEmployeeSignList(item.signerEmployees);

  if (item.signerId) setEmployeeId([item.signerId]);

  setVisibleEmployee(true);
};

const onSignerSelected = () => {
  if (signProcess == null || !employeeId) return;

  const payload = {
    signedDocumentId: selectedSign.id,
    authorizedSignatoryId: employeeId[0],
    onSuccess: () => {
      reloadData();
    },
  };

  dispatch(updateSignedDocument(payload));
};

const onShowMoreHandle = (reason) => {
  Modal.info({
    icon: <></>,
    content: (
      <>
        <div
          style={{
            fontSize: 16,
            lineHeight: "24px",
            fontWeight: 500,
            marginBottom: 12,
            textAlign: "center",
          }}
        >
          Lý do từ chối ký?
        </div>
```

```
          <div>{reason}</div>
        </>
      ),
      centered: true,
      onOk() {},
    });
  };

  if (signProcess == null) return null;

  return (
    <div className="d-flex">
      <div className="d-flex flex-fill">
        {signProcess &&
          signProcess?.signaturePositionGroupByOrder?.map((item, index) => {
            return (
              <div
                key={`${index}-group-step`}
                style={{
                  width: `${(item.totalSign / signProcess.totalSign) * 100}%`,
                }}
                className="sign-group-step"
              >
                {item.signatureProcesses &&
                  item.signatureProcesses.map((sign, index) => {
                    const enable = item.isNextSign;
                    return (
                      <div
                        key={`${index}-step`}
                        style={{
                          width: `${100 / item.totalSign}%`,
                        }}
                      >
                        <Sign
                          signInfo={sign}
                          onSignClick={onSignClick(sign, index)}
                          onChangeSigner={onChangeSigner(sign, index)}
                          onShowMore={onShowMoreHandle}
                          enable={enable}
                        />
                      </div>
                    );
                  })}
              </div>
            );
          })}
      </div>
      <div style={{ width: "280px" }}> </div>
      <ModalEmployeeSignProcess
        visibleModal={visibleEmployee}
        selectedRowKeys={employeeId}
        setSelectedRowKeys={setEmployeeId}
        onCancel={() => {
          setVisibleEmployee((prev) => !prev);
          setEmployeeId(null);
        }}
        onOk={() => {
          setVisibleEmployee((prev) => !prev);
          onSignerSelected();
        }}
        // branchId={branchId}
        isDeptByBranch
        type="radio"
        employeeIds={employeeSignList}
      />
    </div>
  );
};

export default memo(SignProcess);
```