```
import React, { useEffect, useState, useRef } from "react";
import { useDispatch, useSelector } from "react-redux";
import { getAllDocumentTypes, getAllSignPositions, getAllSignatureProcess, getSignatureProcessById } from "store/slices/signatureProcessSlice";
// import { getAllEmployee } from "store/slices/employeeSlice";
import { Spin } from "antd";
import { MediSelect } from "components/Custom";
import MeditechTablePage, { FilterComponent } from "components/table-layout/index";
import TableSignatureProcessList from "./TableSignatureProcessList";
import { SystemButtonSource } from "constants";
import ModalSignatureProcess from "./ModalSignatureProcess";
import { isEmpty } from "lodash";
import Utils from "utils";
import { PermitValue, permitKey } from "constants";

const initSearch = {
  documentTypeId: 1,
  branchId: null,
};

const SignatureProcessList = () => {
  const dispatch = useDispatch();
  const [visibleModal, setVisibleModal] = useState(false);
  const { loading, documentTypeList, signatureProcessList, signatureProcessDetail } =
    useSelector((state) => state.signatureProcess);

  const [signatureProcessId, setSignatureProcessId] = useState(null);
  const { branchId, permitList } = useSelector((state) => state.auth);
  const searchFormRef = useRef({ ...initSearch });
  const [searchFormData, setSearchFormData] = useState({ ...initSearch });

  const searchHandle = dataSearch => {
    searchFormRef.current = { ...searchFormRef.current, ...dataSearch, branchId: branchId };
    setSearchFormData({ ...searchFormRef.current });
    reloadData();
  };

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

  useEffect(() => {
    // dispatch(getAllEmployee({ ...initEmployee }));
    dispatch(getAllSignatureProcess({ ...searchFormRef.current, branchId: branchId }));
  }, [dispatch, branchId]);

  useEffect(() => {
    if (isEmpty(signatureProcessId)) {
      return;
    }
    dispatch(getSignatureProcessById(signatureProcessId));
  }, [dispatch, signatureProcessId]);

  const allowAddNew = Utils.checkPermitValue(
    PermitValue.chophep,
    permitList,
    permitKey.common_signatureProcess
  );

  const getRightButton = isLocked => {
    const arr = [];
    if (isLocked) {
      arr.push({
        keyName: SystemButtonSource.addBtn,
        title: 'Thêm',
        tooltipText: 'Thêm mới',
        showMore: false,
        disable: !allowAddNew,
      });
    }
    return arr;
  };

  const onRightTopBtnHandler = key => {
    if (SystemButtonSource.addBtn === key) {
      if (searchFormRef.current.documentTypeId) {
        setVisibleModal(!visibleModal);
      }
      else {
        alert("Vui lòng chọn loại chứng từ trước!");
      }
    }
  };

  const onRenderSearchView = () => {
    return (
      <FilterComponent
        datasource={searchFormData}
        onSearchImmediate={searchHandle}
      // renderInputSearch
      // renderDatePicker
      >
        <MediSelect
```

```
          key="documentTypeId"
          data-field="documentTypeId"
          style={{ width: "100%", minWidth: 280 }}
          placeholder="Loại chứng từ"
          allowClear
          options={documentTypeList?.map((i) => ({
            label: i.name,
            value: i.id,
          }))}
        />
      </FilterComponent>
    );
  }

  const reloadData = () => {
    dispatch(getAllSignatureProcess({ ...searchFormRef.current, branchId: branchId }));
  };

  return (
    <>
      <Spin tip="Đang tải..." spinning={loading}>
        <MeditechTablePage
          tableTitle='Quản lý quy trình ký'
          btnHeaderRightSource={getRightButton(true)}
          btnHeaderRightSourceHandle={onRightTopBtnHandler}
          onRenderSearchView={onRenderSearchView}
        >
          <TableSignatureProcessList
            setSignatureProcessId={setSignatureProcessId}
            signatureProcessList={signatureProcessList}
            reloadData={() => reloadData()}
            setVisibleModal={setVisibleModal}
            visibleModal={visibleModal}
            permitList={permitList}
          />
        </MeditechTablePage>
      </Spin>

      <ModalSignatureProcess
        visibleModal={visibleModal}
        loading={loading}
        setVisibleModal={setVisibleModal}
        signatureProcessId={signatureProcessId}
        setSignatureProcessId={setSignatureProcessId}
        signatureProcessDetail={signatureProcessDetail}
        documentTypeId={searchFormRef.current.documentTypeId}
        // hospitalList={hospitalList}
        branchId={branchId}
        reloadData={() => reloadData()}
        permitList={permitList}
      />
    </>
  );
};

export default SignatureProcessList;
```