```
import React, { useState, useEffect, useMemo, useRef } from "react";
import { useDispatch, useSelector } from "react-redux";
import { Select, Spin, Tag, Tooltip } from "antd";
import "react-circular-progressbar/dist/styles.css";

import MeditechTablePage, {
  FilterComponent,
} from "components/table-layout/index";
import Utils from "utils";
import MediTable from "components/Custom";
import {
  getAllDocumentTypes,
  getSignReport,
} from "store/slices/signatureProcessSlice";
import CircleStep from "components/CircleStep";
import ModalEmployeeGetAll from "views/app-views/hrm/employee/components/ModalEmployeeGetAll";
import { SIGN_OPTION_STATUS } from "constants";
import { DETAIL_PAGE_PATH } from "configs/AppConfig";
import { DOCUMENT_TYPE_COMP } from "constants";
import { SIGN_STATUS } from "constants";
const { Option } = Select;

const initSearch = {
  startDate: null,
  endDate: null,
  searchText: null,
  signStatus: 2,
  employeeId: null,
  documentType: null,
};

const BanKy = () => {
  const dispatch = useDispatch();
  const [localList, setLocalList] = useState([]);
  const [visibleEmployee, setVisibleEmployee] = useState(false);

  const [selectedRow, setSelectedRow] = useState([]);
  const searchFormRef = useRef(initSearch);
  const [searchFormData, setSearchFormData] = useState(searchFormRef.current);
  const { employeeAllList } = useSelector((state) => state.employee);
  const { documentTypeList } = useSelector((state) => state.signatureProcess);
  const {
    signReportList: { data, loading },
  } = useSelector((state) => state.signatureProcess);

  useEffect(() => {
    dispatch(getSignReport({ ...searchFormRef.current }));
    dispatch(getAllDocumentTypes());
  }, [dispatch]);

  useEffect(() => {
    setLocalList(data);
  }, [data]);

  const reloadData = () => {
    dispatch(getSignReport({ ...searchFormRef.current }));
  };

  const searchHandle = (dataSearch) => {
    const newDataSearch = {
      ...dataSearch,
      startDate: dataSearch.startDate === "" ? null : dataSearch.startDate,
      endDate: dataSearch.endDate === "" ? null : dataSearch.endDate,
    };
    searchFormRef.current = { ...searchFormRef.current, ...newDataSearch };
    setSearchFormData({ ...searchFormRef.current });
    reloadData();
  };
  const onCancelModal = () => {
    setVisibleEmployee(false);
  };
```

```
const onOkModal = () => {
  setVisibleEmployee(false);
};
const columns = useMemo(
  () => [
    {
      title: "STT",
      dataIndex: "index",
      align: "center",
      fixed: "left",
      render: (_, __, index) => index + 1,
    },
    {
      title: "Ngày chứng từ",
      align: "center",
      dataIndex: "createdAt",
      render: (date) => date && Utils.formatDate(date, true),
    },
    {
      title: "Số chứng từ",
      dataIndex: "documentCode",
      key: "documentCode",
      render: (item, record) => {
        const compDetail = DOCUMENT_TYPE_COMP.find(
          (item) => item.id === record.documentTypeId
        );
        console.log(DOCUMENT_TYPE_COMP);
        if (!compDetail) return <Tag color="green">{item}</Tag>;
        let url = `${DETAIL_PAGE_PATH}?page=${compDetail.comp}&${compDetail.comp}Id=${record.documentId}`;
        if (compDetail.comp === "VanBanNoiBoDi") {
          url += `&type=openByNoti`;
        }
        return (
          <>
            <a href={url} target="_blank" rel="noopener noreferrer">
              <Tag color="green">{item}</Tag>
            </a>
          </>
        );
      },
    },
    {
      title: "Loại phiếu",
      dataIndex: "documentName",
      key: "name",
    },
    {
      title: "Người lập",
      dataIndex: "createdName",
      key: "createdName",
    },
    {
      title: "Nội dung",
      dataIndex: "description",
      key: "description",
    },
    {
      title: "Vị trí ký",
      width: "200px",
      dataIndex: "mySignaturesList",
      key: "mySignaturesList",
      render: (signatures) => (
        <span>
          {signatures &&
            signatures.map((signature, index) => {
              const as = SIGN_STATUS.find(
                (item) => item.id === signature.signedStatus
              );
              const titleTooltip = `${as.name} ${
                signature.signedOnUtc
                  ? `lúc ${Utils.formatDate(signature.signedOnUtc)}`
                  : ""
              }`;
              return (
                <span key={index}>
                  <Tooltip title={titleTooltip}>
                    <span style={{ color: as.color }}>
                      {signature.signPositionName}
                    </span>
                  </Tooltip>

                  {index !== signatures.length - 1 && ", "}
                </span>
              );
            })}
          </span>
        ),
      },
      {
        title: "Chữ ký của bạn",
        dataIndex: "mySignaturesList",
        key: "mySignaturesList",
        align: "center",
        render: (signatures) => <CircleStep signatures={signatures} />,
        width: "120px",
      },
      {
        title: "Tổng số chữ ký",
        dataIndex: "allSignaturesList",
        align: "center",
        key: "allSignaturesList",
        render: (signatures) => <CircleStep signatures={signatures} />,
        width: "120px",
      },
    ],
    []
  );

  const onRenderSearchView = () => (
    <FilterComponent
      datasource={searchFormData}
      onSearchImmediate={searchHandle}
      renderInputSearch
      renderDatePicker
    >
      <Select
        allowClear
        key="documentType"
        data-field="documentType"
        placeholder="Loại phiếu"
      >
        {documentTypeList?.map((item) => (
          <Option key={item.id} value={item.id}>
            {item.name}
          </Option>
        ))}
      </Select>
      <Select
        allowClear
        placeholder="Chọn nhân viên"
        key="employeeId"
        data-field="employeeId"
        //onClick={() => setVisibleEmployee(true)}
        //open={false}

        //value={selectedEmployeeId}
      >
        {employeeAllList.map((item) => (
          <Option key={item.id} value={item.id}>
            {item.fullName}
          </Option>
        ))}
      </Select>
      <Select
        allowClear
        key="signStatus"
        data-field="signStatus"
        placeholder="Trạng thái"
      >
        {SIGN_OPTION_STATUS.map((item) => (
          <Option key={item.id} value={item.value}>
            {item.name}
          </Option>
        ))}
      </Select>
    </FilterComponent>
  );

  return (
    <>
      <MeditechTablePage onRenderSearchView={onRenderSearchView}>
        <Spin tip="Đang tải..." spinning={loading}>
          <MediTable
            dataSource={localList}
            tableColumns={columns}
            rowKeyField={(item) => item.id}
          />
        </Spin>
      </MeditechTablePage>
      <ModalEmployeeGetAll
        type={"radio"}
        visibleModal={visibleEmployee}
        selectedRowKeys={selectedRow} // Sử dụng state tạm thời
        setSelectedRowKeys={setSelectedRow} // Cập nhật state tạm thời
        onCancel={onCancelModal}
        onOk={onOkModal}
      />
    </>
  );
};

export default BanKy;
```