```
import React, { useState, useEffect, forwardRef, useImperativeHandle  } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { getSignedDocument } from "store/slices/signatureProcessSlice";
import Expandable from './Expandable';
import SignProcess from './SignProcess';
import { isEmpty } from 'lodash';

const SignFooter = ({ ...rest }) => {

    const dispatch = useDispatch();
    // const { ...rest } = props;
    const { documentTypeId, documentId } = rest;
    // const [signData, setSignData] = useState({});
    const [rejectConfirm, setRejectConfirm] = useState(false);
    const [forceUpdate, setForceUpdate] = useState(false);

    const [signProcess, setSignProcess] = useState();

    const { branchId } = useSelector((state) => state.auth);

     useEffect(() => {
        if (documentTypeId > 0) {
            reloadData();
        }
    }, [documentTypeId, forceUpdate]);

    const reloadData = () => {
        const payload = {
            documentTypeId: documentTypeId,
            documentId: !isEmpty(documentId) ? documentId : null,
            branchId: branchId,
            onSuccess: (result) => {
                // onSetSignData(result);
                setSignProcess(result);
                // let count = 0;
                // for (const item of result.signArrays) {
                //     if (item.DaKy) {
                //         if (onOneSigned) onOneSigned(item);
                //         count += 1;
                //     }
                // }
                // if (onAllSigned && count > 1)
                //     onAllSigned(count >= result.signArrays.length);
            },
        };

        dispatch(getSignedDocument(payload));
    };


    // const onSetSignData = (data = {}) => {
    //     setSignData({ ...data });
    // };

    // useImperativeHandle(ref, () => ({
    //     getChildState: () => {
    //         return signData;
    //     }
    // }));

    if (documentTypeId == null) return <>&nbsp;</>;

    return (
        <div className='sign-container'>
            <Expandable signData={signProcess}>
                <SignProcess
                    className="mb-2"
                    documentTypeId={documentTypeId}
                    documentId={documentId}
                    signData={signProcess}
                    // onSetSignData={onSetSignData}
                    forceUpdate={forceUpdate}
                    {...rest}
                />
            </Expandable>
        </div>
    );
}

export default SignFooter;
```