```
import {
  Col,
  Form,
  Input,
  InputNumber,
  Modal,
  Row,
  Spin,
  notification,
  Checkbox,
} from "antd";
import { isEmpty } from "lodash";
import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { EnvironmentTwoTone } from "@ant-design/icons";
import Utils from "utils";
import { useState } from "react";
import { PermitValue } from "constants";
import { permitKey } from "constants";
import {
  createSignatureProcess,
  updateSignatureProcess,
} from "store/slices/signatureProcessSlice";
import { MediSelect } from "components/Custom";
import { getAllEmployee } from "store/slices/employeeSlice";
import ModalEmployeeGetAll from "views/app-views/hrm/employee/components/ModalEmployeeGetAll";

export const arrUnits = [
  {
    id: "minute",
    label: "phút",
  },
  {
    id: "hour",
    label: "giờ",
  },
  {
    id: "day",
    label: "ngày",
  },
  {
    id: "week",
    label: "tuần",
  },
];

const initEmployee = {
  searchText: null,
  branchId: null,
  workStatus: null,
  gender: null,
  departmentById: null,
  isCertificate: null,
  personnelForm: [],
};

export default function ModalSignatureProcess({
  visibleModal,
  setVisibleModal,
  signatureProcessDetail = null,
  loading,
  reloadData,
  signatureProcessId,
  setSignatureProcessId,
  documentTypeId,
  branchId,
  permitList = [],
}) {
  const dispatch = useDispatch();
  const { signPositionList } = useSelector((state) => state.signatureProcess);
  const { employeeAllList } = useSelector((state) => state.employee);
```

```
const [form] = Form.useForm();
const [allowSubmit, setAllowSubmit] = useState(false);
const [requiredChecked, setRequiredChecked] = useState(true);
const [specificSignerChecked, setSpecificSignerChecked] = useState(true);
const [hidePrintChecked, setHidePrintChecked] = useState(false);
const [unit, setUnit] = useState("minute");
const [isDisabled, setIsDisabled] = useState(false);
const [visibleEmployee, setVisibleEmployee] = useState(false);
const [employeeIds, setEmployeeIds] = useState();

// const allowDelete = Utils.checkPermitValue(
//   PermitValue.xoa,
//   permitList,
//   permitKey.crm_doctor
// );

const onRequiredChange = (e) => {
  setRequiredChecked(e.target.checked);
};

const onSpecificSignerChange = (e) => {
  setSpecificSignerChecked(e.target.checked);
};

const onHidePrintChange = (e) => {
  setHidePrintChecked(e.target.checked);
};

useEffect(() => {
  if (visibleModal) {
    setIsDisabled(false);
  }
}, [visibleModal]);

useEffect(() => {
  dispatch(getAllEmployee({ ...initEmployee }));
}, [dispatch, branchId]);

useEffect(() => {
  if (isEmpty(signatureProcessId)) return;

  setIsDisabled(signatureProcessDetail.signPositionId === 1);

  form.setFieldsValue(signatureProcessDetail);

  setRequiredChecked(signatureProcessDetail.required);
  setSpecificSignerChecked(signatureProcessDetail.specificSigner);
  setHidePrintChecked(signatureProcessDetail.hidePrint);

  // eslint-disable-next-line react-hooks/exhaustive-deps
}, [signatureProcessDetail, form, signatureProcessId]);

const onFinish = (values) => {
  if (
    !Utils.checkPermitValue(
      !isEmpty(signatureProcessId) ? PermitValue.sua : PermitValue.them,
      permitList,
      permitKey.crm_doctor
    )
  ) {
    notification.warning({
      message: !isEmpty(signatureProcessId)
        ? "Sửa quy trình ký"
        : "Thêm quy trình ký",
      description: "Bạn không có quyền thao tác!",
    });
    return;
  }

  let numberOfMinutes = null;

  if (values.signDeadline) {
    if (unit === "minute") numberOfMinutes = values.signDeadline;
```

```
    else if (unit === "hour") numberOfMinutes = values.signDeadline * 60;
    else if (unit === "day") numberOfMinutes = values.signDeadline * 60 * 24;
    else if (unit === "week")
      numberOfMinutes = values.signDeadline * 60 * 24 * 7;
  }

  const payload = {
    documentTypeId: documentTypeId,
    signPositionId: values.signPositionId,
    displayName: values.displayName,
    signOrder: values.signOrder,
    signDeadline: numberOfMinutes,
    description: values.description,
    required: requiredChecked,
    specificSigner: specificSignerChecked,
    hidePrint: hidePrintChecked,
    signerEmployee: values.signerEmployee,
    branchId: branchId,
    onSuccess: () => {
      setVisibleModal(!visibleModal);
      setSignatureProcessId(null);
      form.resetFields();
      reloadData();
    },
  };

  if (!isEmpty(signatureProcessId)) {
    payload.id = signatureProcessId;
    dispatch(updateSignatureProcess(payload));
  } else {
    dispatch(createSignatureProcess(payload));
  }
};

const handleUnitChange = (value) => {
  setUnit(value);
};

const handleSelectChange = (value) => {
  const selectedOption = signPositionList.find(
    (option) => option.id === value
  );
  if (selectedOption) {
    form.setFieldsValue({ displayName: selectedOption.name });
  } else {
    form.setFieldsValue({ displayName: "" });
  }
};

const selectAfter = (
  <MediSelect
    defaultValue={unit}
    style={{
      width: 80,
    }}
    options={arrUnits?.map((i) => ({
      label: i.label,
      value: i.id,
    }))}
    onChange={handleUnitChange}
  />
);

return (
  <>
    <Modal
      maskClosable={false}
      bodyStyle={{ overflowY: "auto", maxHeight: "calc(100vh - 300px)" }}
      width={680}
      title={
        !isEmpty(signatureProcessId)
          ? "Sửa quy trình ký"
          : "Thêm quy trình ký"
      }
```

```
    open={visibleModal}
    onCancel={() => {
      setVisibleModal(!visibleModal);
      form.resetFields();
      setSignatureProcessId(null);
    }}
    okButtonProps={{
      // disabled: !allowSubmit,
      title: allowSubmit ? "Thao tác" : "Bạn không có quyền thao tác",
    }}
    onOk={() => form.submit()}
    okText={!isEmpty(signatureProcessId) ? "Lưu" : "Thêm"}
  >
    <Spin tip="Đang tải..." spinning={loading}>
      <Form form={form} layout="vertical" onFinish={onFinish}>
        <Row
          gutter={24}
          style={{ display: "flex", justifyContent: "center" }}
        >
          <Col sm={24} md={24} lg={24}>
            <Row gutter={16}>
              <Col span={24}>
                <Form.Item
                  name="signPositionId"
                  label="Vị trí ký"
                  rules={[
                    {
                      required: true,
                      message: "Vị trí là bắt buộc",
                    },
                  ]}
                >
                  <MediSelect
                    disabled={isDisabled}
                    placeholder="Chọn vị trí ký"
                    allowClear
                    style={{ width: "100%", minWidth: 280 }}
                    options={signPositionList?.map((i) => ({
                      label: i.name,
                      value: i.id,
                    }))}
                    onChange={handleSelectChange}
                  />
                </Form.Item>
              </Col>

              <Col span={24}>
                <Form.Item
                  name="displayName"
                  label="Tên hiển thị"
                  rules={[
                    {
                      required: true,
                      message: "Tên hiển thị là bắt buộc",
                    },
                  ]}
                >
                  <Input placeholder="Nhập tên hiển thị" />
                </Form.Item>
              </Col>

              <Col span={24}>
                <Form.Item
                  name="signOrder"
                  label="Thứ tự ký"
                  rules={[
                    {
                      required: true,
                      message: "Thứ tự ký là bắt buộc",
                    },
                  ]}
                >
                  <InputNumber
                    style={{
```

```
        width: "100%",
      }}
      placeholder="Nhập thứ tự ký"
    />
  </Form.Item>
</Col>
<Col span={12}>
  <Form.Item name="required" span={12}>
    <Checkbox
      checked={requiredChecked}
      onChange={onRequiredChange}
    >
      <b>Bắt buộc</b>
    </Checkbox>
  </Form.Item>
</Col>

<Col span={12}>
  <Form.Item name="specificSigner" span={12}>
    <Checkbox
      disabled={isDisabled}
      checked={specificSignerChecked}
      onChange={onSpecificSignerChange}
    >
      <b>Chỉ định người ký</b>
    </Checkbox>
  </Form.Item>
</Col>
<Col span={24}>
  <Form.Item
    name="signerEmployee"
    label="Người được ký"
    rules={[
      {
        required:
          !isDisabled && (!requiredChecked || !specificSignerChecked) ,
        message: "Chỉ định người ký là bắt buộc",
      },
    ]}
  >
    <MediSelect
      mode="multiple"
      disabled={isDisabled}
      // options={employeeAllList}
      options={employeeAllList.map((item) => ({
        value: item.id,
        label: item.fullName,
      }))}
      onClick={() => {
        if(isDisabled) return false;
        setVisibleEmployee((prev) => !prev);
        setEmployeeIds(form.getFieldValue("signerEmployee"));
      }}
      optionLabelProp="label"
      open={false}
      placeholder="Chọn người ký"
    />
  </Form.Item>
</Col>

<Col span={24}>
  <Form.Item name="signDeadline" label="Hạn ký">
    <InputNumber
      style={{
        width: "100%",
      }}
      addonAfter={selectAfter}
      placeholder="Không giới hạn"
    />
  </Form.Item>
</Col>

<Col span={24}>
  <Form.Item name="description" label="Mô tả">
```

```
                    <Input.TextArea
                      prefix={<EnvironmentTwoTone />}
                      placeholder="Nhập mô tả"
                    />
                  </Form.Item>
                </Col>

                <Col span={12}>
                  <Form.Item name="hidePrint" span={12}>
                    <Checkbox
                      checked={hidePrintChecked}
                      onChange={onHidePrintChange}
                    >
                      <b>Ẩn in</b>
                    </Checkbox>
                  </Form.Item>
                </Col>
              </Row>
            </Col>
          </Row>
        </Form>
      </Spin>
    </Modal>
    <ModalEmployeeGetAll
      visibleModal={visibleEmployee}
      selectedRowKeys={employeeIds}
      setSelectedRowKeys={setEmployeeIds}
      onCancel={() => {
        setVisibleEmployee((prev) => !prev);
      }}
      onOk={() => {
        setVisibleEmployee((prev) => !prev);
        form.setFieldValue("signerEmployee", employeeIds);
      }}
      branchId={branchId}
    />
  </>
 );
}
```