```
import React, { useCallback, useEffect, useState } from "react";
import { Badge, Avatar, List, Button, Popover, Spin } from "antd";
import {
  MailOutlined,
  WarningOutlined,
  CheckCircleOutlined,
} from "@ant-design/icons";
import Flex from "components/shared-components/Flex";
import BellNoti from "components/Icon/BellNoti";
import { useDispatch, useSelector } from "react-redux";
import {
  getNoti,
  readAllNoti,
  readNoti,
} from "store/slices/common/erpNotificationSlice";

import { Link, useNavigate } from "react-router-dom";
import { DETAIL_PAGE_PATH, NOTIFICATION_REPORT_PATH } from "configs/AppConfig";
import moment from "moment";

const getIcon = (icon) => {
  switch (icon) {
    case "mail":
      return <MailOutlined />;
    case "alert":
      return <WarningOutlined />;
    case "check":
      return <CheckCircleOutlined />;
    default:
      return <MailOutlined />;
  }
};
export const NavNotification = () => {
  const dispatch = useDispatch();
  const {
    dataNav: { data, loading },
  } = useSelector((state) => state.erpNotification);
  const [displayedNotifications, setDisplayedNotifications] = useState([]);
  const [visibleCount, setVisibleCount] = useState(10);
  const [totalNotifications, setTotalNotifications] = useState([]);
  const navigate = useNavigate();

  const handleNavigate = () => {
    navigate(NOTIFICATION_REPORT_PATH);
  };
  const handleMarkRead = (record) => {
    const payload = {
      id: record?.notiId,
      onSuccess: () => {
        dispatch(getNoti({}));
        let url = `${DETAIL_PAGE_PATH}?page=${record.objectType}&${record.objectType}Id=${record.objectId}`;
        if (record.objectType === "VanBanNoiBoDi") {
          url += `&type=openByNoti`;
        }
        window.open(url, "_blank", "noopener,noreferrer");
      },
    };
    dispatch(readNoti(payload));
  };

  useEffect(() => {
    setDisplayedNotifications(totalNotifications.slice(0, visibleCount));
  }, [totalNotifications, visibleCount]);

  const loadMore = useCallback(() => {
    setVisibleCount((prevCount) => {
      const newCount = prevCount + 5;
      setDisplayedNotifications(totalNotifications.slice(0, newCount));
      return newCount;
    });
  }, [totalNotifications]);
```

```
const handleScroll = (event) => {
  const bottom =
    event.target.scrollHeight ===
    event.target.scrollTop + event.target.clientHeight;
  if (bottom) {
    loadMore();
  }
};
useEffect(() => {
  dispatch(getNoti({}));
}, [dispatch]);

useEffect(() => {
  if (data) {
    setTotalNotifications(data.notificationList);
  }
}, [data]);

const handleReadAll = () => {
  const payload = {
    onSuccess: (data) => {
      dispatch(getNoti({}));
    },
  };
  dispatch(readAllNoti(payload));
};
const NotificationItem = React.memo(({ item }) => (
  <List.Item
    className="list-clickable"
    style={{
      backgroundColor: !item.isRead ? "#f0f0f0" : "transparent",
    }}
  >
    <Link
      className="d-block"
      onClick={(e) => {
        e.preventDefault();
        handleMarkRead(item);
      }}
    >
      <Flex alignItems="center">
        <div className="pr-3">
          {item.senderAvatarUrl ? (
            <Avatar src={`${item.senderAvatarUrl}`} />
          ) : (
            <Avatar
              className={`ant-avatar-${item.type}`}
              icon={getIcon(item.icon)}
            />
          )}
        </div>
        <div>
          <span className="font-weight-bold text-dark">{item.subject} </span>
          <span className="text-gray-light">{item.content}</span>
          <div>
            <small className="ml-auto">
              {moment(item.sendTime).fromNow()}
            </small>
          </div>
        </div>
      </Flex>
    </Link>
  </List.Item>
));
const getNotificationBody = useCallback(
  (list) =>
    list.length > 0 ? (
      <List
        size="small"
        itemLayout="horizontal"
        dataSource={list}
        renderItem={(item) => (
          <NotificationItem key={item.notiId} item={item} />
```

```
          )}
        />
      ) : (
        <div className="empty-notification">
          <img
            src="https://gw.alipayobjects.com/zos/rmsportal/sAuJeJzSKbUmHfBQRzmZ.svg"
            alt="empty"
          />
          <p className="mt-3">You have viewed all notifications</p>
        </div>
      ),
    []
  );

  const notificationList = (
    <div>
      <div className="nav-notification-header d-flex justify-content-between align-items-center">
        <h4 className="mb-0">Thông báo</h4>
        {data?.countNotRead > 0 && (
          <Button
            className="text-primary"
            type="text"
            onClick={() => {
              handleReadAll();
            }}
            size="small"
          >
            Đánh giấu đã đọc{" "}
          </Button>
        )}
      </div>
      <div className="nav-notification-body" onScroll={handleScroll}>
        <Spin tip="Đang tải..." spinning={loading}>
          {getNotificationBody(displayedNotifications)}
        </Spin>
      </div>
      {displayedNotifications.length > 0 ? (
        <div className="nav-notification-footer">
          <Link
            className="d-block"
            onClick={(e) => {
              e.preventDefault();
              handleNavigate();
            }}
          >
            Xem tất cả{" "}
          </Link>
        </div>
      ) : null}
    </div>
  );

  return (
    <Popover
      placement="bottomRight"
      title={null}
      content={notificationList}
      trigger="click"
      overlayClassName="nav-notification"
    >
      <div className="nav-item">
        <Badge count={data?.countNotRead} overflowCount={9}>
          <BellNoti />
        </Badge>
      </div>
    </Popover>
  );
};

export default NavNotification;
```