```
import axios from "axios";
import { signOutApi } from "store/slices/authSlice";
import store from "../store";
import { AUTH_TOKEN, REFRESH_TOKEN } from "constants/AuthConstant";
import { notification } from "antd";
import { REFRESH_TOKEN_API } from "constants/ApiConstant";
import { setAuthenData } from "utils/helper";

const unauthorizedCode = [400, 401, 403];

const service = axios.create({
  baseURL: process.env.REACT_APP_HRM_ENDPOINT_URL,
  timeout: 120000,
});
// Config
const TOKEN_PAYLOAD_KEY = "authorization";

// API Request interceptor
service.interceptors.request.use(
  (config) => {
    const jwtToken = localStorage.getItem(AUTH_TOKEN) || null;
    if (jwtToken) {
      config.headers[TOKEN_PAYLOAD_KEY] = `Bearer ${jwtToken}`;
    }
    return config;
  },
  (error) => {
    // Do something with request error here
    notification.error({
      message: "Error",
    });
    Promise.reject(error);
  }
);
// API respone interceptor
service.interceptors.response.use(
  (response) => {
    return response.data;
  },
  async (error) => {
    let notificationParam = {
      message: "",
    };
    if (error.response) {
      if (unauthorizedCode.includes(error.response.status)) {
        notificationParam.message = "Bạn không có quyền thao tác";
      }
      if (error.response.status === 601) {
        // token expired
        try {
          const refreshToken = localStorage.getItem(REFRESH_TOKEN);
          if (refreshToken) {
            const response = await axios.post(
              REFRESH_TOKEN_API,
              {
                Token: refreshToken,
              },
              { baseURL: process.env.REACT_APP_IDENTITY_ENDPOINT_URL }
            );
            if (response.data && response.data.data) {
              setAuthenData(response.data.data);
              const originalConfig = error.config;
              originalConfig._retry = true;
              service.defaults.headers.common.Authorization = `Bearer ${response.data.data.tokenData.token}`;
              return await service(originalConfig);
            } else {
              store.dispatch(signOutApi());
            }
          }
        } catch (_error) {
          notificationParam.message =
            "Phiên đăng nhập hết hạn, xin vui lòng đăng nhập lại";
```

```
          store.dispatch(signOutApi());
        }
      } else if (error.response.status === 603) {
        notificationParam.message =
          "Mật khẩu đã bị thay đổi, xin vui lòng đăng nhập lại";
        store.dispatch(signOutApi());
      } else if (error.response.status === 602) {
        notificationParam.message =
          "Phiên đăng nhập hết hạn, xin vui lòng đăng nhập lại";
        store.dispatch(signOutApi());
      } else if (error.response.status === 604) {
        notificationParam.message = `Tài khoản đã bị khóa. Xin vui lòng liên hệ với quản trị hệ thống để xử lý`;
        store.dispatch(signOutApi());
      } else if (error.response.status === 600) {
        notificationParam.message = `Lỗi: ${error.response.data.description}`;
      } else if (error.response.status === 404) {
        notificationParam.message = `Not Found. ${error.response.data.description}`;
      } else if (error.response.status === 500) {
        notificationParam.message = "Internal Server Error";
      } else if (error.response.status === 508) {
        notificationParam.message = "Time Out";
      } else if (error.response.status === 400) {
        notificationParam.message = `Lỗi: ${error.response.data.description}`;
      }
    } else {
      notificationParam.message = "Network Error";
    }

    notification.error(notificationParam);

    return Promise.reject(error);
  }
);

export default service;
```