//æøå

import \* as React from 'react';

import { RouteComponentProps } from 'react-router';

import { User } from "../utilities/user";

import { Http } from "../utilities/http";

import { TypeStyle } from "./TypeStyle";

//the React component is for the user to register his nfc chip

//currently this can only happen manually

export class InitNfc extends React.Component<RouteComponentProps<{}>, {}> {

manualRegisterInput: HTMLInputElement | null = null;

profileData: ProfileDataResult | null = null;

constructor() {

super();

if (User.current == null) {

window.location.href = "/";

return;

}

Http.post<ProfileDataResult>("api/UserGateway/ProfileData", {

username: User.current!.username,

key: User.current!.key

}).then(x => {

this.profileData = x;

this.setState({});

}).catch(x => console.log(x));

}

public render() {

if (this.profileData == null) {

return <div>loading</div>;

}

else {

if (this.profileData.nfcId != null) {

return <div>you already have an nfc initialized</div>

}

else {

return <div>

<TypeStyle css={

[

{

identifier: ".column",

style: {

float: "left",

width: "50%",

}

},

{

identifier: ".row:after",

style: {

display: "table",

clear: "both",

}

}

]

} />

<div className="row">

<div className="column">

<h2>

automatic register

</h2>

<p>

comming soon™

</p>

</div>

<div className="column">

<h2>

manual register

</h2>

{(() => {

let submit: (() => Promise<void>) = async (): Promise<void> => {

if (this.manualRegisterInput != null) {

let result: RegisterNfcResult = await Http.post<RegisterNfcResult>("api/UserGateway/RegisterNfc", {

key: User.current!.key,

nfcId: this.manualRegisterInput.value

})

if (result.success) {

alert("new nfc registered");

window.location.href = "/";

}

else {

alert("error");

console.log(result);

}

}

}

return <div><label>nfc id: </label> <input ref={(e) => this.manualRegisterInput = e} onKeyUp={(e) => {

if (e.keyCode == 13) {

submit();

}

}} /> <button onClick={submit}>submit</button></div>

})()}

</div>

</div>

</div>

}

}

}

}

interface ProfileDataResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

username: string | null;

communitiesName: string[] | null;

communitiesUniqueName: string[] | null;

myself: boolean | null;

nfcId: string | null;

}

interface RegisterNfcResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

}