import { NgRedux } from '@angular-redux/store';

import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { ActivatedRoute, Router } from '@angular/router';

import { Observable, throwError } from 'rxjs';

import { map, catchError } from 'rxjs/operators';

import { FETCHED\_COMPANY\_DETAILS } from 'src/app/store/app.action';

import { AppStoreInterFace } from 'src/app/store/app.store';

import { environment } from 'src/environments/environment';

import { CONST\_URL\_SEGEMENTS } from '../common/data.url';

import { AppErrors } from '../errors/app.errors';

import { NotFoundError } from '../errors/not-found.errors';

@Injectable()

export class AppInitService{

[x: string]: any;

getSubDomain: any

getStoredDomain:any

isResponse:boolean=false

constructor(private http: HttpClient, private router: Router, private ngRedux: NgRedux<AppStoreInterFace>) {}

httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json'

})

};

getCompanyInformation (CID: string) {

// this.ngRedux.dispatch({type: REQ\_COMPANY\_DETAILS}) /\* store.isFetchin = true\*/

const cid = new HttpParams().set('CID', CID);

return this.http.get(environment.APP\_URL + CONST\_URL\_SEGEMENTS.INIT.GET\_COMPANY\_INFO , {headers: this.httpOptions.headers , params: cid})

// .subscribe((dt: any)=>{

// console.log('COmdsad', dt)

// if(dt.status == 200) {

// console.log('companyName', )

// this.ngRedux.dispatch({type: FETCHED\_COMPANY\_DETAILS, companydetails: dt.data})

// localStorage.setItem('uservalidity', btoa(atob(btoa(JSON.stringify(dt.data)))));

// } else {

// this.router.navigate(['page-not-found']);

// }

// },(error: AppErrors) => {

// if (error instanceof NotFoundError) {

// alert('this is expected');

// } else {

// alert('something went wrong');

// }

// })

.pipe(

map((res: any) => {

if (res.status==200) {

// localStorage.setItem('token', JSON.stringify(res.token));

this.ngRedux.dispatch({type: FETCHED\_COMPANY\_DETAILS, companydetails: res.data})

localStorage.setItem('uservalidity', btoa(atob(btoa(JSON.stringify(res.data)))));

return true ;

} else {

this.router.navigate(['page-not-found'])

return false;

}

}),

catchError((error: Response) => {

if (error.status === 400) {

return throwError(new NotFoundError(error));

}

return throwError(new AppErrors(error));

})

);

}

get allLanguages () {

let languageList:any = []

const languageListstate = this.ngRedux.select('compDetails')

languageListstate.subscribe((comp:any) => {

languageList = comp.languages

})

console.log('LANGUAGE######',languageList)

return languageList

}

get getCompanyInfo () {

let compInfoTemp:any;

const languageListstate = this.ngRedux.select('compDetails')

languageListstate.subscribe((comp:any) => {

compInfoTemp = comp

})

console.log('cinfo', compInfoTemp)

return compInfoTemp

}

get defaultLanguage () {

let findDefault:any;

const defaultLang = this.ngRedux.select('selectedLang')

defaultLang.subscribe((lang:any) => {

findDefault = lang

})

console.log('\*\*\*\*\*\*\*UUUUU',findDefault)

return findDefault

}

get validateDomain () {

this.getSubDomain = new URL(window.location.href).port;

let getuservalidity:any = localStorage.getItem('uservalidity')

if(getuservalidity) {

let decodedUserValidity= atob(btoa((atob(getuservalidity))))

if(this.getSubDomain == JSON.parse(decodedUserValidity).DomainAlias) {

return true

} else{

return false

}

} else {

this.getCompanyInformation(this.getSubDomain).subscribe((dt) =>{

this.isResponse =dt

}, (error:AppErrors)=> {

if(error instanceof NotFoundError){

alert('this is expected');

this.isResponse =false

}

else{

alert('something went wrong');

this.isResponse =false

}

})

console.log('dddddd' ,this.isResponse)

return this.isResponse ;

}

}

}

import { NgRedux } from '@angular-redux/store';

import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { ActivatedRoute, Router } from '@angular/router';

import { Observable, throwError } from 'rxjs';

import { map, catchError } from 'rxjs/operators';

import {  FETCHED\_COMPANY\_DETAILS } from 'src/app/store/app.action';

import { AppStoreInterFace } from 'src/app/store/app.store';

import { environment } from 'src/environments/environment';

import { CONST\_URL\_SEGEMENTS } from '../common/data.url';

import { AppErrors } from '../errors/app.errors';

import { NotFoundError } from '../errors/not-found.errors';

@Injectable()

export class AppInitService{

[x: string]: any;

getSubDomain: any

getStoredDomain:any

isResponse:boolean=false

//companyinf:string=''

constructor(private http: HttpClient, private router: Router, private ngRedux: NgRedux<AppStoreInterFace>) {}

httpOptions = {

    headers: new HttpHeaders({

      'Content-Type':  'application/json'

    })

};

setcode(code:any){

 return  btoa(atob(btoa(JSON.stringify(code))));

}

getcode(decode:any){

  return  atob(btoa((atob(decode))));

 // atob(btoa((atob(getuservalidity))))

 }

getCompanyInformation (CID: string) {

  // this.ngRedux.dispatch({type: REQ\_COMPANY\_DETAILS}) /\* store.isFetchin = true\*/

  const cid = new HttpParams().set('CID', CID);

    return this.http.get(environment.APP\_URL + CONST\_URL\_SEGEMENTS.INIT.GET\_COMPANY\_INFO , {headers: this.httpOptions.headers , params: cid})

    // .subscribe((dt: any)=>{

    //   console.log('COmdsad', dt)

    //   if(dt.status == 200) {

    //     console.log('companyName', )

    //     this.ngRedux.dispatch({type: FETCHED\_COMPANY\_DETAILS, companydetails: dt.data})

    //     localStorage.setItem('uservalidity', btoa(atob(btoa(JSON.stringify(dt.data)))));

    //   } else {

    //     this.router.navigate(['page-not-found']);

    //   }

    // },(error: AppErrors) => {

    //   if (error instanceof NotFoundError) {

    //     alert('this is expected');

    //   } else {

    //     alert('something went wrong');

    //   }

    // })

    .pipe(

      map((res: any) => {

        if (res.status==200) {

          // localStorage.setItem('token', JSON.stringify(res.token));

          this.ngRedux.dispatch({type: FETCHED\_COMPANY\_DETAILS, companydetails: res.data})

      // localStorage.setItem('uservalidity', btoa(atob(btoa(JSON.stringify(res.data)))));

   localStorage.setItem('companyInfo', this.setcode(res.data));

       // localStorage.setItem('companyInfo', (JSON.stringify(res.data)));

        return true ;

        } else {

          this.router.navigate(['page-not-found'])

          return false;

        }

      }),

      catchError((error: Response) => {

        if (error.status === 400) {

          return throwError(new NotFoundError(error));

        }

        return throwError(new AppErrors(error));

      })

    );

}

get allLanguages () {

  let languageList:any = []

  const languageListstate = this.ngRedux.select('compDetails')

  languageListstate.subscribe((comp:any) => {

    languageList = comp.languages

  })

  console.log('LANGUAGE######',languageList)

  return languageList

}

get getCompanyInfo () {

  let compInfoTemp:any;

  const languageListstate = this.ngRedux.select('compDetails')

  languageListstate.subscribe((comp:any) => {

    compInfoTemp = comp

  })

  console.log('cinfo', compInfoTemp)

  return compInfoTemp

}

get defaultLanguage () {

  let findDefault:any;

  const defaultLang = this.ngRedux.select('selectedLang')

  defaultLang.subscribe((lang:any) => {

    findDefault = lang

  })

  console.log('\*\*\*\*\*\*\*UUUUU',findDefault)

  return findDefault

}

get validateDomain () {

  this.getSubDomain = new URL(window.location.href).port;

  let getuservalidity:any =  localStorage.getItem('companyInfo')

  if(getuservalidity) {

    //let companyinf=localStorage.getItem('companyInfo')

   // let decodedUserValidity= atob(btoa((atob(getuservalidity))))

   let decodedUserValidity=this.getcode(getuservalidity);

    if(this.getSubDomain == JSON.parse(decodedUserValidity).DomainAlias) {

      return true

    } else{

      return false

    }

  } else {

    this.getCompanyInformation(this.getSubDomain).subscribe((dt) =>{

      this.isResponse =dt

    }, (error:AppErrors)=> {

      if(error instanceof NotFoundError){

        alert('this is expected');

        this.isResponse =false

      }

      else{

        alert('something went wrong');

        this.isResponse =false

      }

    })

    console.log('dddddd' ,this.isResponse)

    return this.isResponse ;

  }

}

}