Final Vue Capstone Project Seed - Vue-Starter-Java Frontend

This is the vue starter project for the final capstone. This document will walk you through how to setup and run the project. It will also explain some of the features of the starter such as vue router and authentication.

Project setup

The first thing you will need to do is to download any dependencies by running the command:

```
npm install
```

Next you will need to open up the file `.env` which is located in the root of the project. This configuration file is a great place to store any environment variables that you want to use throughout your application. When you open up the file it should look something like this:

```
# Java
VUE_APP_REMOTE_API=http://localhost:8080

# .NET
VUE_APP_REMOTE_API=https://localhost:44358
```

This is the API endpoint that your Vue frontend will talk to for things like authentication and registration. You will want to remove whatever language you aren't using and make sure the URL is correct. Before you can start your frontend application your backend needs to be running so this is a good time to make sure that it is.

If you have followed the previous steps you should be able to start your application using the following command:

```
npm run serve
```

Authentication

When you first run the project and visit the URL you will be taken to a login page. The reason for this is because the home route `/` is secured by default. If you look in `/src/router.js` you should see the following code:

```
router.beforeEach((to, from, next) => {
  // Determine if the route requires Authentication
  const requiresAuth = to.matched.some(x => x.meta.requiresAuth);
  const user = auth.getUser();

  // If it does and they are not logged in, send the user to "/login"
```

```
    if (requiresAuth && !user) {
      next("/login");
    } else {
      // Else let them go to their next destination
      next();
    }
  });
```

You might not have had a chance to cover this but this is a feature of Vue Router called Navigation Guards. If you haven't had a chance it might be worth reading through the documentation to learn what they are and how they work.

In short the code above will run before each route. It will first check to see if the route requires authentication which is defined per route using the meta object key `requiresAuth`. In the following configuration you must be authenticated to view the home route while anyone can visit the login & registration routes.

```
const router = new Router({
  mode: 'history',
  base: process.env.BASE_URL,
  routes: [
    {
      path: '/',
      name: 'home',
      component: Home,
      meta: {
        requiresAuth: true
      }
    },
    {
      path: "/login",
      name: "login",
      component: Login,
      meta: {
        requiresAuth: false
      }
    },
    {
      path: "/register",
      name: "register",
      component: Register,
      meta: {
        requiresAuth: false
      }
    },
  ]
})
```

Next it will try and get a user to see if you are logged in. The application stores users in `localStorage`. It will check to see if there is a key in local storage called `Authorization` which will contain the authenticated user token. If the route requires authentication and no user token was found it will forward you to the login page.

```
    // If it does and they are not logged in, send the user to "/login"
    if (requiresAuth && !user) {
      next("/login");
    } else {
      // Else let them go to their next destination
      next();
    }
```

Login

When you hit the `/login` route you will see a bar login page and this was intentional. It is up to you to style this page to fit within your application. When you fill in a username and password and click the Sign In button the method `login()` will be called. This will pass your login credentials to your backend application's REST API for authentication.

```
login() {
  fetch(`${process.env.VUE_APP_REMOTE_API}/login`, {
    method: 'POST',
    headers: {
      Accept: 'application/json',
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(this.user),
  })
    .then((response) => {
      if (response.ok) {
        return response.text();
      } else {
        this.invalidCredentials = true;
      }
    })
    .then((token) => {
      if (token != undefined) {
        if (token.includes('"')) {
          token = token.replace(/"/g, '');
        }
        auth.saveToken(token);
        this.$router.push('/');
      }
    })
    .catch((err) => console.error(err));
}
```

Register

When you hit the `/register` route you will see a bar register page and this was intentional. It is up to you to style this page to fit within your application. When you fill in a username, password, confirm password click the

Create Account button the method `register()` will be called. This will pass your user details to your backend application's REST API to create a new user.

```
methods: {
register() {
  fetch(`${process.env.VUE_APP_REMOTE_API}/register`, {
    method: 'POST',
    headers: {
      Accept: 'application/json',
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(this.user),
  })
    .then((response) => {
      if (response.ok) {
        this.$router.push({ path: '/login', query: { registration:
'success' } });
      } else {
        this.registrationErrors = true;
      }
    })

    .then((err) => console.error(err));
}
```