Capstone project

It's the year 3542. Some people evolved into superheroes. The superheroes ended up saving the climate and the world is a happy place now. That's all great. However, emergencies still occur sometimes. We need to register all the superheroes. Somehow, you're still alive and your superpowers are Java, Angular, and SQL.

The president asked you to create a voluntary sign-up system for superheroes. This way they can be called for help when they're needed.

EXERCISE 1 CREATE THE BACKEND

1. Create a new project for the superhero sign-up backend. Add the following dependencies: web, data jpa, h2.
2. Configure the application.properties for h2.
3. Create an entity for your superhero. It should have:
    a. Name
    b. Nickname
    c. Superpower (String is fine to start with)
    d. Telephone number
    e. Date of birth
4. Create a repository for your superheroes.
    a. Get all
    b. Get by id
    c. Add a new hero
    d. BONUS: Modify a hero
    e. BONUS: Delete a hero
    f. Find a hero by superpower

Capstone project *cont'd*

5. Create a controller and service for your superheroes that contains methods for the methods under 2.
6. Test your backend using Postman.
7. Make sure to add automated tests for your endpoints and services as well.
8. BONUS: Make your backend connect to a MySQL database instead of having an H2 database.
9. BONUS: Also add an entity for a superpower, you can assume this is a many-to-many relationship.
10. BONUS BONUS: You can give superpower a column category. Category should be an entity with a list of Superpowers. And give every superhero a property weakness of type Category.

EXERCISE 2 CREATE THE FRONTEND (OPTION 1 FUN)

Our backend needs a frontend. We want to show the superheroes that have signed up and add a sign-up form that we can use to add superheroes. These are not ALL the steps you need to take, but some of them include:

1. Create a new React project called superheroes-sign-up using `create-react-app`.

```
npx create-react-app superheroes-sign-up
```

2. Create components for Header, Footer, SignUpForm, and HeroesOverview. These can be either functional or class components depending on your preference.

```
// example of functional component
function Header() {
```

Capstone project *cont'd*

```
return (
      <header>
        <h1>Superheroes Sign Up</h1>
      </header>
    );
  }
```

3. Add routing using react-router-dom package.

npm install react-router-dom

And then in your App.js:

```
  import { BrowserRouter as Router, Route, Switch } from
'react-router-dom';


  function App() {
    return (
      <Router>
        <Header />
        <Switch>
          <Route path="/signup" component={SignUpForm} />
          <Route path="/heroes" component={HeroesOverview}
/>
        </Switch>
        <Footer />
      </Router>
    );
```

```
}
```

5. Create a service to connect to the backend. This can be done with the fetch API or libraries like axios. E.g.:

```
export const getSuperheroes = async () => {
  const response = await fetch('your-backend-url');
  const data = await response.json();
  return data;
};
```

6. BONUS: Test your application. You can use Jest, which comes with create-react-app, or any other testing library you prefer.

7. BONUS: populate a dropdown with the superpowers that are available in the sign up form.

8. BONUS: add an edit option. This could involve creating an additional EditForm component and Route.

9. BONUS: add a delete option. This would involve adding a delete button to each Superhero item in the HeroesOverview, which would call a deleteSuperhero function in your backend service.

Remember to use `npm start` to run your application during development and check the functionality of each part as you create them.

Capstone project *cont'd*

EXERCISE 2 CREATE THE FRONTEND (OPTION 2 ALSO FUN)

If you don't want to use React, you can opt to create the frontend with HTML/CSS/JavaScript.

EXERCISE 2 CREATE THE FRONTEND (OPTION 3 FUN^2)

If you don't want to use React, or plain HTML/CSS/JavaScript… go for Angular? Vue?

EXERCISE 3 OPTIONAL

1. Add a login/authorization system to your app.

2. Maybe the new world isn't only peaceful… Create supervillains as well.

3. Give both superheroes and supervillains an HP and a Boolean aliveness.

4. Add an endpoint and UI for battling between superheroes and supervillains when one of them reaches HP 0 their status changes from alive to dead.

5. BONUS: make superheroes and supervillains both extend from the class SuperPerson.