Assignment
Create a Pokémon Trainer using Angular

Pokémon Trainer

Build a Pokémon Trainer web app using the Angular Framework. You have freedom to be as creative as you wish, if it meets the minimum requirements described in Appendix A.

1) Set up the development environment

Make sure you have the following tools available:

- Figma
- NPM/Node.js (LTS – Long Term Support version)
- Angular CLI
- Visual Studio Code Text Editor/ IntelliJ
- Browser Developer Tools for testing and debugging
    - Angular Dev Tools
- Git
- Trainer API: https://github.com/dewald-els/noroff-assignment-api
- Heroku

2) Design a Component Tree

Use Figma to create a component tree of the application. The component tree should show the pages and feature components you plan to create in your application. This will count towards the overall grade for the application. It should be done BEFORE a single line of code is written.

3) Test API in Postman

Test the endpoints to get an understanding of the data structures that will be used in your application. Use the Pokémon API to display Pokémon names: https://pokeapi.co/.

4) Write HTML & CSS as needed

a) **Colours:** If you have trouble choosing colours, use a free resource like https://coolors.co to browse and experiment with colour combinations.
b) **Animations:** If you want to use animations to bring your design to life, use https://animate.style/.
c) **Free graphics for your web applications:** https://www.justinmind.com/blog/35-places-to-get-free-vector-images-for-your-designs/
d) See Pokémon Catalogue page section for more information on Pokémon images.

5) Use the Angular framework to build the following screens into your Pokémon Trainer app. (See Appendix A for detailed specs):

a) The *Landing* Page
b) The *Trainer* Page
c) The *Pokémon Catalogue* Page

6) Submit

a) Export the component tree to PDF, upload the file to the project's Git repository and submit a link to your file.
b) Publish your Single Page Application on Heroku and submit a link to your app and the source code on your Git repository. Use https://gitlab.com/javascript-project-examples/heroku-deployment-guides/-/blob/main/guides/heroku-angular-deployment.md to learn how to deploy Angular apps to Heroku.

Appendix A: Requirements for the Pokémon Trainer app

The application allows a user to collect Pokémon received from the PokeAPI. Users must enter username before being able to collect any Pokémon. Users must also be able to view the Pokémon that have been collected.

1) Landing Page

The first thing a user should see is the "Login page" where the user must be able to enter their "Trainer" name.

There should be a button that saves the Trainer name to the Trainer API and in localStorage. The app must then be redirected to the main page, the Pokémon Catalogue page.

The users may *NOT* be able to see the Pokémon Catalogue without have a Trainer name stored in localStorage. Use a Guard service to achieve this functionality.

If username exists in localStorage, you may automatically redirect to the Pokémon Catalogue page.

You can first check if the username exists in the Trainer API before redirecting to the Catalogue page.

NB! Local storage can ONLY store strings. You will have to stringify the data using JSON.stringify when storing objects. Remember, when reading the data, you will have to parse it back to a JavaScript object using JSON.parse.

2) Trainer Page

A user may only view this page if there is a Trainer name that exists in localStorage. Please redirect a user back to the Landing page if they do not have a Trainer name stored in localStorage. Use a Guard service to achieve this functionality.

The Trainer page should list the Pokémon that the trainer has collected. For each collected Pokémon, display the Pokémon name and image.

A user must also be able to remove a Pokémon from their collection from the Trainer page.

3) Pokémon Catalogue Page

The Catalogue page may *NOT* be viewed if there is no Trainer name stored in localStorage. Use a Guard service to achieve this functionality.

The Catalogue page must list the Pokémon name and avatar*.

It is recommended to get all the Pokémon and store it in sessionStorage. (Use the limit query parameter for the PokeAPI) When the page is reloaded, read from sessionStorage rather than the API. This way, the PokeAPI is not constantly hit with requests every time you save your files.

You may optionally create a page that uses Pagination to load *n* number of Pokémon at a time.

*** Note:**
You may use the Github repository to obtain images for the Pokémon. You can use the URL of the image with the id of the Pokémon.

The ID of the Pokemon is in the initial response as part of the URL property. You will need to "extract" the id from this URL.

```
{
  name:"bulbasaur"
  url:"https://pokeapi.co/api/v2/pokemon/1/"
}
```

Example for Pokémon with id of 1:
https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/1.png

See all sprites:
https://github.com/PokeAPI/sprites

Add a button on each Pokémon that, when clicked, adds the Pokémon to the trainer's collection. This must update the Trainer API with the collected Pokémon.

You must also visually indicate that this Pokémon has been collected. You may choose how to indicate this, perhaps display a small Poke ball in the corner of the collected Pokémon, but you have freedom to be creative.

You may *optionally* add a details section to each Pokémon that is *ONLY* shown when a "Show more info" button is clicked. Here you can add information like the Pokémon base states and abilities.

Do *NOT* download all the Pokémon details on the Catalogue page. Only download additional information for the Pokémon if you add the "Show more" feature.

Required features

The following features/tools must be present in the application:

- **Angular framework**
- **Angular Router** to navigate between pages
- Store the username and collected Pokémon in the Trainer API (Noroff API deployed to Heroku).
- Use Angular Services to manage the state of your application

Optional features

None.