Assignment 7
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)
- AngularCLI
- Visual Studio Code Text Editor
- Browser Developer Tools for testing and debugging
- Git
- Heroku or Gitlab Pages (You may choose)

2) Recommended: Design a wireframe

Use Figma to create a wireframe of the layout. The layout should only show placeholders and positioning of your design (i.e., no colours or images required).

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/.

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 wireframe 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 Gitlab Pages and submit a link to your app and the source code on your Git repository. Use https://dev.to/gaurangdhorda/deploy-angular-project-to-gitlab-pages-using-gitlab-ci-aik to learn how to deploy Angular apps to Gitlab Pages. You may also use Heroku, if you prefer.

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 name. Once the name has been stored in local storage, the app must display the main page, the Pokémon Catalogue page. The users must NOT be able to see the Pokémon Catalogue without have a username stored in local storage. Users that are already have already entered their username, may automatically be redirected to the Pokémon Catalogue page.

NB!
Local storage can ONLY store strings. You will have to stringify the data using JSON.stringify. 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 they have a username stored in local storage. Please redirect a user back to the Landing page if they do not have a username stored in local storage.

The trainer page should list the Pokémon that the user has collected.

Note:
You may store an array of Pokémon in local storage. Remember, local storage can ONLY store strings.

3) Pokémon Catalogue Page

The catalogue page may NOT be viewed if there is no username stored in local storage. The Catalogue page must list the Pokémon name and avatar*.

** Note*:
You may use the `npm` package to obtain high quality images for the Pokémon.
https://github.com/PokeAPI/sprites

Clicking on a Pokémon should add the Pokémon to the trainer's collection. And 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.

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 local storage.
- Use Angular Services to manage the state of your application

Optional features

None.