React - Technical Interview Exercise

Project Overview

Read carefully all the requirements and create a product that fits the expectations. You have to make a public GitHub repository to upload the exercise and share the link in the answer field.

Your development should be driven by an informal user story that you will create, and which should be included in your presentation.

Requirements

Frontend

Create a Pokémon app with the following features:

Login screen:

- The creds consist of a username/password form. You should validate the user credentials in a backend you have to build. (admin as username and admin as password, anything different should be considered as incorrect creds). Show all the validation that you think makes sense.
- The user should remain logged against a storage instance that fits your preference (Local db, Local storage, cookies). Routes should be protected, so if an already logged in user tries to log in, it should be redirected to the main page, and if it is not logged in and it tries to go to the main page, it should be redirected to the login page.

Main page:

- The home screen will have a search bar with a list of Pokémon. You'll have to use an API with the requirements described below. The API response will be paginated, so you should create a solution for it.
- Users should be able to sort results by name and number.
- Each Pokémon should be shown with its photo, name, and number.

React - Technical Interview Exercise

Detail view:

- If the user clicks on a Pokémon from the list, the user should be redirected to a detailed Pokémon page with detailed information about the Pokémon (Abilities, moves, and forms).

Final Design:

- In this link, you will find the design requirements the application should follow. Figma design. The design is done for a mobile screen, but feel free to adapt the design for bigger screens.

Notes:

- Feel free to use any library for local state management and UI.
- You should keep in mind SEO and responsiveness as part of your implementation.
- We will evaluate the architecture you think is better for the app. Please keep in mind that the app could have more features in the future.
- Be prepared to discuss your solution.

Backend

- You will **implement your own lightweight backend** that will rely on https://pokeapi.co/ as a source of truth to obtain Pokémon information. You may use **any backend technology** you're comfortable with (e.g., **Ruby on Rails, Node.js/Express, Python/Flask or FastAPI, PHP/Laravel**, etc.)
- This backend should provide three endpoints:
    - Login: this is to handle credentials authorization (admin/admin)
    - /pokemons: it should provide all the pokemons paginated as in the pokeapi
    - /pokemons/{id}: it should provide the detailed information of a pokémon.
- Feel free to add any other endpoint you consider necessary to implement the

application.

- Notes:
  - You can **use GenAI** to help write or optimize your code — we encourage you to!
  - If you do use GenAI tools (Cursor, Claude Code, etc.), please **share the prompts** you used and any **modifications** you made to the AI-generated code.
  - You do **not** need to implement full authentication or a production-ready DB unless you want to.
  - You may use **in-memory data**, SQLite, or any lightweight DB of your choice.

Generative AI tools

The following task should also be added in the same repo in a README file.

Imagine you're tasked with generating a **Table component** for a simple **task management system** using your preferred language. The system should support the following functionality:

- Create, read, update, and delete tasks (CRUD)
- Each task has a `title`, `description`, `status`, and `due_date`
- Tasks are associated with a user (assume basic User model exists)

Instructions:

- Using your preferred **GenAI coding tool** (e.g., Cursor, Claude Code, Windsurf, GitHub Copilot, etc.), write the **prompt you would use** to generate the API scaffold or full implementation.
- Show the **output code** (or a representative sample of it).
- Describe how you:

- Validated the AI's suggestions
- Corrected or improved the output, if necessary
- Handled edge cases, authentication, or validations
- Assessed the performance and idiomatic quality of the code

Presentation and Code Review

You will be required to present your project to the technical interview panel. During the presentation, you should explain your user story, design choices, the technical architecture, and demonstrate the functionality of the application. This will be done over Google Meets or Zoom, and you will screen share either your GitHub repository or your IDE.

After the presentation, the interview panel will conduct a code review of your project. You will be asked to explain your coding decisions and answer any questions related to the code. The interview panel will evaluate your project based on the following criteria:

- **Clean Architecture**: Your architecture should adhere to Clean Architecture principles, including separation of concerns and independence of components.
- **Application testing:** Your project should have sufficient test coverage. Use of TDD is preferable.
- **Code quality**: Your code should be well-organized, readable, and adhere to best practices.
- **Functionality**: Your application should perform as expected in the requirements without errors or bugs. Optional but desired: no warnings in the browser console.
- **Presentation**: Your presentation should be clear, concise, and demonstrate a good understanding of the project and of the main ***backend*** and ***frontend*** best practices.
- **GenAI tools**: Your answers and presentation must show fluency with GenAI tools and

prompt engineering, and critical thinking when evaluating AI-generated code.