📌 Build an app with a user-management system

Our product team has imagined a new game, the “Cool Kids Network” ; and they need you to implement a proof of concept.
The Cool Kids Network is an online app that offers several features:

- Users can register and get a character automatically created through the front-end.
- A logged-in user can see the data of its character through the front-end.
- Users with the special role ”Cooler Kid” can see other character’s data, except the email and role through the front-end.
- Users with the special role ”Coolest Kid” can see other character’s data, including the email and role through the front-end.
- Assigning a special role to a user can be done through an API.

User Story 1

As an anonymous user, I can sign-up and generate a character.

The app homepage has a sign-up button displayed to anonymous users. When clicked, the user goes to a signup page with:

- a text field: email address;
- A button: Confirm.

When “Confirm” is clicked with a valid email that is not already registered:

- An account is created for this email on the website.
- A fake identity is generated and stored, representing the user’s character. This fake identity is composed of: First name, last name, country, email address, role. The email address is the one from the sign-up form; role is set to “Cool Kid” by default, and the other data is generated from randomuser.me API.

User Story 2

As a user with an account, I can log into the website and see my character’s data.

The website offers a login page, accessible from the homepage, where the user can enter its email address and log in. If the email address is associated with an existing user, the login is successful. Otherwise, it is declined.

Note: As this is a proof of concept, you should not care about passwords for now. Accepted solutions can be (not restricted to): Not checking a password or setting the same password for all users.

Once logged in, I can visualize the data of my character: First name, last name, country, email address, role.

User Story 3

As a logged in user with Cooler Kid or Coolest Kid role, I have access to the name and country of all users.

Once logged in and if the user has a Cooler Kid or Coolest Kid role, the user should be able to get a list of all the names and countries of all registered users on the website.
Users with a Cool Kid role don’t have access to this feature.

User Story 4

As a logged in user with Coolest Kid role, I have access to the email and role of all users.

Once logged in and if the user has a Coolest Kid role, the user should be able to get the role and email associated with each email of registered users.

User Story 5

As a maintainer, I can change a user’s role through a 3rd party integration.

The website must offer a protected endpoint that accepts authenticated requests and allows to set the role of a specific user. The endpoint must be able to identify a user with its email only, and with its first and last names only.
Only possible roles are “Cool Kid”, “Cooler Kid” and “Coolest Kid”.
If a valid request is received, the specified role is applied to the targeted user.

Technical Expectations

Here are the technical expectations and implementation guidelines your answer will be evaluated upon. It is okay not to match every single criterion, if it is mentioned and discussed (see Documentation 👇)

- The app is built with
    - Backend: Python, Node.js or PHP;
    - Frontend: React, Angular or Vue.js.
- You can freely use frameworks, libraries or packages.
- It's complete and works: It delivers the right expected outcome per the user stories.
- The implementation is documented (see the Documentation paragraph below)
- It lives in a GitHub repo and retains its history.
- It's built with modern OOP. It uses procedural where it makes sense.
- The system is observable, resilient and can easily be monitored: It produces relevant logs, handles errors and the database can be read easily. Those elements are clearly documented.

🎖 Bonus Points

Want extra credit? Cool, here are some bonus opportunities for you:

- Your codebase passes a linter inspection.
- Your codebase has meaningful unit and integration tests.
- Your GitHub repository has a CI automatically checking linter and tests on every commit to the main branch.
- The front-end design is user-friendly. You can get inspiration from our own websites!

Tips & Suggested solutions

The following points are suggestions to facilitate your implementation. They are not mandatory, and you can choose another approach if you feel more comfortable with it.

- In case you need a public URL for your app, you can temporarily use tools like ngrok to get a URL redirecting to your localhost.

Documentation

Add an `Explanation.md` file to your repo that explains at least the following:

- The problem to be solved in your own words.
- A technical specification of your design, explaining how it works.
- The technical decisions you made and why.

- How your solution achieves the admin’s desired outcome per the user story.

Feel free to add anything that seems relevant to understanding your work and your approach. Think out loud. We want to know:

- How you approach a problem
- How you think about it
- Why you chose this direction
- Why this direction is a better solution

Questions?

Email us. It's okay to ask. We're happy to help you.