PANORAMIC
SOFTWARE INC.

Panoramic Software was founded in 1982 with a mission to help government agencies leverage technology and better serve their communities. By modernizing workflows, Panoramic works to make agency staff more effective and productive thus increasing their impact.

Panoramic Software has evolved with technology and advocated for government modernization across the country. We provide state of the art applications, hosted in a secure and stable environment. We stand behind our products with outstanding customer support, which leads to long-term partnerships.

Please see the coding assignment below:

Data Modeling and Design

Please author the types for two representations of a Person along with the types of the function signatures converting one to the other.

- **Editing**: Various fields may be in an incomplete or invalid state while a person is in the process of being edited.
- **Validated / Fully Formed**: This model will be consumed by the rest of the application and should make impossible states unrepresentable, act as proof that the model has passed certain validations, and assist users of the model by preventing structural mistakes, enforcing semantics, and simplifying use.

For this exercise we will limit the fields we model to

- First Name
- Last Name
- Social Security Number
- Marital Status
- US Phone Number

Consider data integrity, usage ergonomics, transparency of the Api, and any other quality measures you generally value. You should design the validated Person based upon your own personal experience.

Artifacts of this exercise

- Type signatures for both the editable and the validated version of a Person.
- Key function signatures involved in translating an editable version of a Person to a validated Person. Do not worry about implementing these.
    - If there are any validations that cannot be represented in types then please do include a few code comments discussing those.
- Briefly describe the Module organization for these types and what should be exposed and what should be hidden. Be ready to discuss how this impacts total system maintainability.

Dog Breed Api

See https://dog.ceo/dog-api/documentation/ for general Api documentation. Write a small frontend application that

- Can be in one of two basic user interface states.
  - Dog Breed List
    - Loads the list of dog breeds/sub-breeds if not already loaded `dog.ceo/api/breeds/list/all`
    - Displays the list of breeds with sub-breed names displayed. The display styling need not be fancy but the list should be sorted alphabetically.
    - The user can transition to Dog Breed Details by clicking on a specific breed in the list.
  - Dog Breed Details
    - The user can return to the Dog Breed List state.
    - Loads image list from `dog.ceo/api/breed/{breed}/images`. A specific breed should be loaded at most once per session.
    - Displays the total number of images for the breed.
    - Displays 20 images at a time
    - Allows the user to page forward and backward with Previous and Back buttons. The buttons should only be enabled when appropriate.
- General Notes
  - Make a call to the underlying Api for a specific Url only once per application session / instance. The same data should not be fetched twice for the same run of the application so be certain to model that.
  - Always indicate to the user when data is loading and disallow interactions while loading.
  - Do not worry about fancy styling.
  - In a real application these states would be represented as routes but that complexity has been excluded here to reduce the burden of plumbing code.