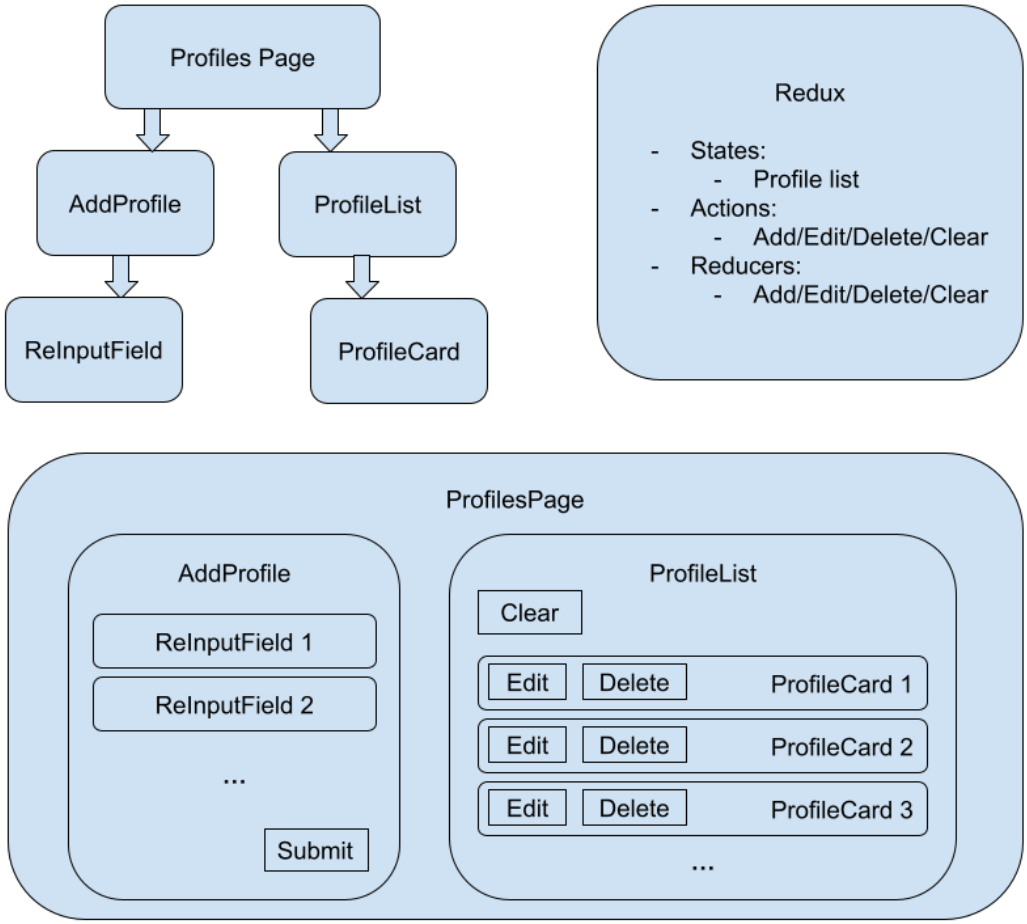
Goal

We want to create a single page that displays a list of user profiles and a form that allows you to add user profiles to the list. The frontend must be implemented according to this diagram.



Requirements

1. Coding
    a. Backend: Node.js Express server in **TypeScript**
    b. Frontend: React, **Redux**, in **TypeScript**
        i. You must follow the component hierarchy tree in the top left of the diagram. The bottom of the diagram is an example of how the components *could* be rendered on the page, but the design is up to you.
        ii. ReInputField and ProfileCard are **reusable components**.
            1. https://blog.logrocket.com/building-reusable-ui-components-with-react-hooks/
    c. Database: MongoDB
    d. Cloud: **AWS S3**
2. Set up your Github repository and add the Training Managers as collaborators.
    a. eric.lee@beaconfireinc.com
    b. rtam@beaconfireinc.com

c. ehuang@beaconfireinc.com
d. rlin@beaconfireinc.com

Features

1. React
    a. **ProfilesPage:** the root component that renders the two child sibling components below.
        i. **ProfileList:** renders a list of reusable profile card components (passing in props) and a button to clear the list. This component needs to read from redux state and send each profile as props to the child profile card.
            1. **ProfileCard:** renders all of the properties (picture, name, email, phone) and two clickable buttons (edit and delete).
                a. **Edit:** each of the properties should become form input fields that allow users to modify its contents. The form fields should be prefilled with the existing values of that profile.
                    i. Hint: conditionally render AddProfile, but you'll need a way to customize the action type for dispatch
                b. **Delete:** dispatch an action to delete this target profile
            2. Clear button: this should dispatch an action to the store
        ii. **AddProfile:** renders a form with a list of 4 reusable input field components (passing in props) and a button for submission. You should define an object state that contains properties for each input field and its relevant attributes and a reusable onChange handler. The onChange handler should also do basic input validation and set the validity of the state.
            1. **ReInputField:**
                a. Renders a label and an input field using the following props:
                    i. id (string)
                    ii. input type (string)
                    iii. field name (string)
                    iv. value (string)
                    v. valid (boolean)
                    vi. onChange (function)
                b. The input fields should be highlighted green if valid else red if not valid. They are valid if they meet these conditions.
                    i. **Profile picture:** valid if type jpg/jpeg/png
                    ii. **Name:** valid if string with only alphabetical characters
                    iii. **Email:** valid if string with the format *prefix@domain*
                    iv. **Phone number:** valid if string that contains 10 digit characters in the format *xxx-xxx-xxxx*

        2. Submit button: this should generate the payload based on the local state and dispatch an action to the store
2. Redux
    a. Set it up in two different ways: the traditional way (covered during training) **and** with Redux Toolkit.
    b. Actions (type: payload) with THUNK middleware
        i. **ADD:** the new user profile
        ii. **EDIT:** the target user profile
        iii. **DELETE:** the target user profile
        iv. **CLEAR:** { }
    c. Store (what slices of state?)
        i. Profile list: this represents the list of all user profiles and their properties
    d. Reducers
        i. **ADD:** add a user profile to the list
        ii. **EDIT:** modify an existing user profile
        iii. **DELETE:** remove an existing user profile
        iv. **CLEAR:** reset the list
3. Node.js
    a. 4 routes, one for each request
        i. **GET:** for requesting all user profiles
        ii. **POST:** for adding a new user profile
        iii. **PUT:** for editing user profile information
        iv. **DELETE:** for deleting a user profile
    b. When adding or editing a user profile with an uploaded picture, the server should communicate with S3 to reflect the changes.
4. MongoDB
    a. User model has 4 properties: profile picture, name, email, phone number
    b. Seed the database with 5 sample users.

NOTE:

1. This mini-project is meant for practice with TypeScript, React reusable components, Redux, and cloud technologies. Any features that are not listed here (login/register, styling, routing, etc.), are not required.
2. **Context API is not allowed**.

Submission

1. You do not need to submit anything anywhere or record a video demo.
2. There is a *soft* deadline, on **7/20 Wednesday.**
3. **You will not be able to enter the marketing phase** until you complete this project and get approved.