PROBLEM

As a WRCC website visitor, I need to create a new account so that I can access certain web applications.

SOLUTION

Using either plain JavaScript, HTML, and CSS **or** Vue.js, write a registration form UI which connects to **https://wrcc.dri.edu/pass** for registering accounts.

The form must contain fields for Account Name, Email, First Name, Last Name, Phone Number, Password, Confirm Password, and a Submit button.

Each field must apply validation on user input which adheres to the values allowed by the back end for that field. (See the "Allowed values" column in Table 1 for what is expected for each field). Attempts to submit the form when fields are invalid or empty must be rejected by the UI and the UI should provide feedback to the user to indicate which fields are invalid and why. Only when the form is valid should clicks on the Submit button fire off a request to the back-end to register the user.

subsvc	key	Value type	required/optional	Allowed values
acct_create	acct_name	<string>	required	case-insensitive, must be unique, minimum 5 characters
	email	<string>	required	case-insensitive, must be unique, valid e-mail address format
	first	<string>	required	first name
	last	<string>	required	last name
	phone	<alphanum string>	required	valid phone number using dashes
	passwd	<alphanum string>	required	minimum 12 characters alphanum/special characters

Table 1: Specification for the acct_create sub-service

Making the Registration Request

- Requests to the back end should come in the form of a **POST** request to **https://wrcc.dri.edu/pass.**
- The **Content-Type** header should be set to **application/x-www-form-urlencoded**
- The request body must include the keys from Table 1 along with the values from the corresponding form field, as well as the following key/value pairs:
    - `"session_id": ""` (empty string)
    - `"svc": "acct"`
    - `"subsvc": "acct_create"`

Example Request Body

```
{
    "session_id": "",
    "svc": "acct",
    "subsvc": "acct_create",
    "acct_name": "fsociety",
    "email": "elliot@fsociety.com",
    "first": "Elliot",
    "last": "Alderson",
    "phone": "775-123-4576",
    "passwd": "whiterose59!"
}
```

Receiving the Registration Response

Responses from the back end will be JSON-formatted. If there is an error in processing the response, the `"status"` prop will be `"ERR"` and the `"msg"` prop will contain the error message. In this case, the UI must notify the user of the error and display the error message provided in the response.

If the request is successful, the `"status"` prop will be `"OK"`. In this case, the UI must notify the user of successful registration and display `"session_id"`, `"timeout"`, and `"msg"` values in the response to the user. (Note: the `"timeout"` value represents the number of seconds left until the session expires).

Example Response Body on Error

```
{
    "api_ver": "1.201903",
    "session_id": "",
    "status": "ERR",
    "msg": "account name already used; e-mail address already used",
    "sent": "2020-10-22T14:06:50",
    "received": "2020-10-22T14:06:50",
```

```
}
```

Example Response Body on Success

```
{
    "api_ver": "1.201903",
    "session_id": "<some hexadecimal string>",
    "status": "OK",
    "acct_name": "fsociety",
    "first": "Elliot",
    "last": "Alderson",
    "timeout": "600",
    "msg": " Welcome to the WRCC Web site and its data services!",
    "sent": "2020-10-22T11:36:50",
    "received": "2020-10-22T11:36:52"
}
```

Your solution should, at the very least, print the registration response to the UI, so that the user can verify that either an error or a success response has been returned. Any additional UI is considered optional.

You may send your solution in an email reply or post to a public repository like GitHub. If you send your solution over email, please compress all files into a single tarball or zip file. If you post to a repository, please email a link to the project so that we can look at your implementation before meeting. In either case, make sure to include a “README” file that explains how to setup and run your solution in a browser.

Be prepared to discuss your solution in detail as part of the interview.