Project Challenge: Sign-Up Form Data Handling

In this challenge, we will complete the Sign-Up process by handling the data and adding a new user in the system.

WE'LL COVER THE FOLLOWING

Problem statement

In this challenge, we will build upon the solution of the previous challenge and complete the signup process by handling the data. We will also add a new user to the system by signing them up.

> 📌 **Note:** You are provided with a **list of dictionaries** called `users` in **line 17** in the application given below. It already contains one user, which is also the admin.

Your tasks in this challenge are:

1. **Validation:** only properly validated data should be processed.
2. **Error handling**: validation errors should be shown to the user on the `signup.html` template.
3. **Insertion of the new user**: the valid data after being submitted should then be used to create a new user that is appended to the `users` list.
4. **Success message:** after successful insertion of the new user, the sign-up page should show a **"successfully signed up"** message.

Sign-up form (errors) - expected output

Sign-Up Form (Errors) - Expected Output

Sign-up form (success message) - expected output

Sign-Up Form (Success Message) - Expected Output

Your implementation

Implement the features described above in the application provided below.

> ⚠ **Disclaimer:** If you edit the contents of `style.css`, you might notice that the application does not reflect the changes immediately. The reason for this is that the previous version is saved in your browser's cache. To solve this problem:
>
> 1. You can hard refresh your browser to clear the cache.

2. Alternatively, you can use inline `CSS` in the `HTML` templates using the `style` attribute. **(Recommended)**

```
"""Flask Application for Paws Rescue Center."""
from flask import Flask, render_template, abort
from forms import SignUpForm

app = Flask(__name__)
app.config['SECRET_KEY'] = 'dfewfew123213rwdsgert34tgfd1234trgf'

"""Information regarding the Pets in the System."""
pets = [
            {"id": 1, "name": "Nelly", "age": "5 weeks", "bio": "I am a tiny kitten rescued b
            {"id": 2, "name": "Yuki", "age": "8 months", "bio": "I am a handsome gentle-cat.
            {"id": 3, "name": "Basker", "age": "1 year", "bio": "I love barking. But, I love
            {"id": 4, "name": "Mr. Furrkins", "age": "5 years", "bio": "Probably napping."},
        ]

"""Information regarding the Users in the System."""
users = [
            {"id": 1, "full_name": "Pet Rescue Team", "email": "team@pawsrescue.co", "passwor
        ]


@app.route("/")
def homepage():
    """View function for Home Page."""
    return render_template("home.html", pets = pets)


@app.route("/about")
def about():
    """View function for About Page."""
    return render_template("about.html")


@app.route("/details/<int:pet_id>")
def pet_details(pet_id):
    """View function for Showing Details of Each Pet."""
    pet = next((pet for pet in pets if pet["id"] == pet_id), None)
    if pet is None:
        abort(404, description="No Pet was Found with the given ID")
    return render_template("details.html", pet = pet)


@app.route("/signup", methods=["POST", "GET"])
def signup():
    """View function for Showing Details of Each Pet."""
    form = SignUpForm()
    return render_template("signup.html", form = form)

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=3000)
```

In the next lesson, we will discuss the solution to this challenge.