Solution: Create a Login and Logout Mechanism

In this lesson, we will go over the solution of this challenge.

WE'LL COVER THE FOLLOWING

Solution

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

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()
    if form.validate_on_submit():
        new_user = {"id": len(users)+1, "full_name": form.full_name.data, "email": form.email
        users.append(new_user)
        return render_template("signup.html", message = "Successfully signed up")
    return render_template("signup.html", form = form)


@app.route("/login", methods=["POST", "GET"])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = next((user for user in users if user["email"] == form.email.data and user["pas
        if user is None:
            return render_template("login.html", form = form, message = "Wrong Credentials. P
        else:
            session['user'] = user
            return render_template("login.html", message = "Successfully Logged In!")
    return render_template("login.html", form = form)

@app.route("/logout")
def logout():
    if 'user' in session:
        session.pop('user')
    return redirect(url_for('homepage', _scheme='https', _external=True))

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

Explanation

Let's break down the solution of this challenge to figure out how we solved it.

Modifications in `app.py`

1. The `login` view

In the `login` view, we first took care of authenticating the user. In **line 58**, if the `form.validate_on_submit()` returns `true`, we searched for that user in the

`users` list. If such a `user` was **not found**, then we returned the `form` with the `message` saying, *"Wrong credentials. Please try again."*

If the `user` is found, then we authenticate the user by adding it to the session object with the key `'user'`. Then, we return the template with only the `message` saying, *"Successfully logged in!"*

2. The `logout` view

The `logout` view is really straightforward. This view has the `URL` route `/logout`. In this view, we check if the `'user'` is present in the `session`, in **line 69**. If `'user'` is found, we remove it by using:

```
session.pop('user')
```

Afterward, we redirect the user to the `homepage` view by using the `redirect` and `url_for()` function that we imported in **line 4**.

```
return redirect(url_for('homepage', _scheme='https', _external=True))
```

Modifications in `base.html`

We also had the task of modifying the navigation bar in the case of a logged-in user. For this purpose, we used an `if` statement in Jinja to check if `'user'` is present in the `session`. If the `'user'` was indeed present, we added a link to the `logout` route.

```
{% if 'user' in session %}
    <li style="display:inline"><a href="{{url_for('logout')}}">Log Out</a>
</li>
```

Otherwise, we displayed the default links in the `else` block.

```
{% else %}
    <li style="display:inline"><a href="{{url_for('signup')}}">Sign Up</a>
</li>
    <li style="display:inline"><a href="{{url_for('login')}}">Login</a></l
i>
```

Well done! Now you have mastered the form handling and **sessions** in `Flask`.

Now, let's move on and learn about how to interact with a database using the

`Flask` application. Stay tuned!