Project Challenge: Retrieving a User at Login

In this challenge, we will be modifying the login function to retrieve an inserted user from the database.

WE'LL COVER THE FOLLOWING

Problem statement

In this challenge, we will modify the `login` method so that it retrieves the required `user` from the database instead of searching in the list.

> 📌 **Note:** we have already inserted one `user` in the database for you. This is the **admin** user. You can see the insertion on **lines 31 to 41**. You can use this `user` to test out the modified `login` method.

You will perform the following tasks:

1. Modify the `login` method to check for the `user` in the database and then authenticate it.
2. Next, instead of storing the `user` object in the `session`, only store the `id` of the user in the `session`.

Your implementation

Implement the features described above in the application provided below.

> ⚠ **Disclaimer:** Please do not remove the `users` and `pets` list from the application yet, as it may break the existing features of the application.

```
"""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
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SECRET_KEY'] = 'dfewfew123213rwdsgert34tgfd1234trgf'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///paws.db'
db = SQLAlchemy(app)

"""Model for Pets."""
class Pet(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String, unique=True)
    age = db.Column(db.String)
    bio = db.Column(db.String)
    posted_by =  db.Column(db.String, db.ForeignKey('user.id'))


"""Model for Users."""
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    full_name = db.Column(db.String)
    email = db.Column(db.String, unique=True)
    password = db.Column(db.String)
    pets = db.relationship('Pet', backref = 'user')

db.create_all()

# Create "team" user and add it to session
team = User(full_name = "Pet Rescue Team", email = "team@petrescue.co", password = "adminpass
db.session.add(team)

# Commit changes in the session
try:
    db.session.commit()
except Exception as e:
    db.session.rollback()
finally:
    db.session.close()

"""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 = User(full_name = form.full_name.data, email = form.email.data, password =
        db.session.add(new_user)
        try:
            db.session.commit()
        except Exception as e:
            print(e)
            db.session.rollback()
            return render_template("signup.html", form = form, message = "This Email already
        finally:
            db.session.close()
        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)
```

In the next lesson, we'll take a look at the solution to this challenge.