Project Challenge: Create User and Pet Models

In this challenge, we will be initializing a database connection and creating the User and Pet models in the project application.

WE'LL COVER THE FOLLOWING ^

- Problem statement
- Your implementation

Problem statement

Up until now, we have been using lists of dictionaries to store the information related to **users** and **pets** in our project. However, now that we have learned how to use the database, we can create **models** for these entities instead.

Therefore, in this challenge, you will complete the following tasks:

- 1. Initiate a database connection with an **SQLite** database.
- 2. Create **models** called **Pet** and **User** to store the data that we previously stored in the lists.
- 3. The email column of the User model and the name column of the Pet model should contain the unique constraint on them.
- 4. Create the tables in the database associated with the **models**.

Your implementation

Implement the features described above in the application provided below.

<u>M</u> Disclaimer: Please do not remove the users and pets list from the application yet, as it may break the existing features of the application.

```
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
            {"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. F
        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.