## Project Challenge: Updating and Deleting Pets

In this challenge, you will be given the task of updating and deleting pets.

#### WE'LL COVER THE FOLLOWING

- Problem statement
  - Pet Details page expected output
  - Pet Details page (after editing the pet) expected output
  - Home page (after deleting the pet) expected output
- Your implementation

### Problem statement #

In this challenge you are required to implement the following features:

- Add and edit pet form: you will create a new form that contains all the fields to edit a pet.
- Modify pet\_details to edit the pet: make necessary changes in the
   pet\_details view so that it can show the new form and handle form data
   to edit the pet.
- Add a delete pet button: you will create a button in the Pet Details page that can be used to delete the pet.
- Add a delete\_pet view: for the purpose of deleting the pet, you will also have to create a new view called delete\_pet. This view should redirect to the Home page after deletion.

Pet Details page - expected output #



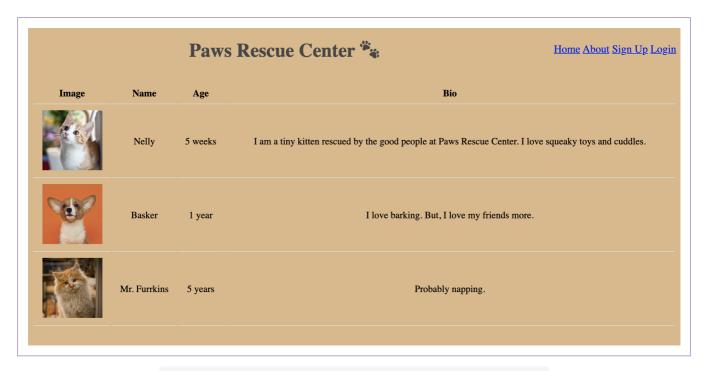
Pet Details Page - Expected Output

# Pet Details page (after editing the pet) - expected output #



Pet Details Page (After Editing the Pet) - Expected Output

Home page (after deleting the pet) - expected output #



Home Page (After Deleting the Pet) - Expected Output

# Your implementation #

```
"""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)
# Create all pets
nelly = Pet(name = "Nelly", age = "5 weeks", bio = "I am a tiny kitten rescued by the good pe
yuki = Pet(name = "Yuki", age = "8 months", bio = "I am a handsome gentle-cat. I like to dres
basker = Pet(name = "Basker", age = "1 year", bio = "I love barking. But, I love my friends m
mrfurrkins = Pet(name = "Mr. Furrkins", age = "5 years", bio = "Probably napping.")
# Add all pets to the session
db.session.add(nelly)
db.session.add(yuki)
db.session.add(basker)
db.session.add(mrfurrkins)
# Commit changes in the session
    db.session.commit()
except Exception as e:
   db.session.rollback()
finally:
   db.session.close()
@app.route("/")
def homepage():
    """View function for Home Page."""
    pets = Pet.query.all()
    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)
    pet = Pet.query.get(pet_id)
    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 = User.query.filter_by(email = form.email.data, password = form.password.data).f
        if user is None:
            return render_template("login.html", form = form, message = "Wrong Credentials. F
        else:
            session['user'] = user.id
            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)
```

Let's take a look at the solution to this challenge in the next lesson.