Lab 12

Many To Many Mapping & Image Thumbnails

Initial Set Up

1. Go to www.github.com and log in to your GitHub account.

2. Go to Moodle and click on the link for Lab 12.

3. Once you have accepted the assignment you are asked to refresh the page, and you are then presented with a link to your repository for lab 12.

4. When you click on the repository link, you are taken to the repository where you see a starter project has already been added:

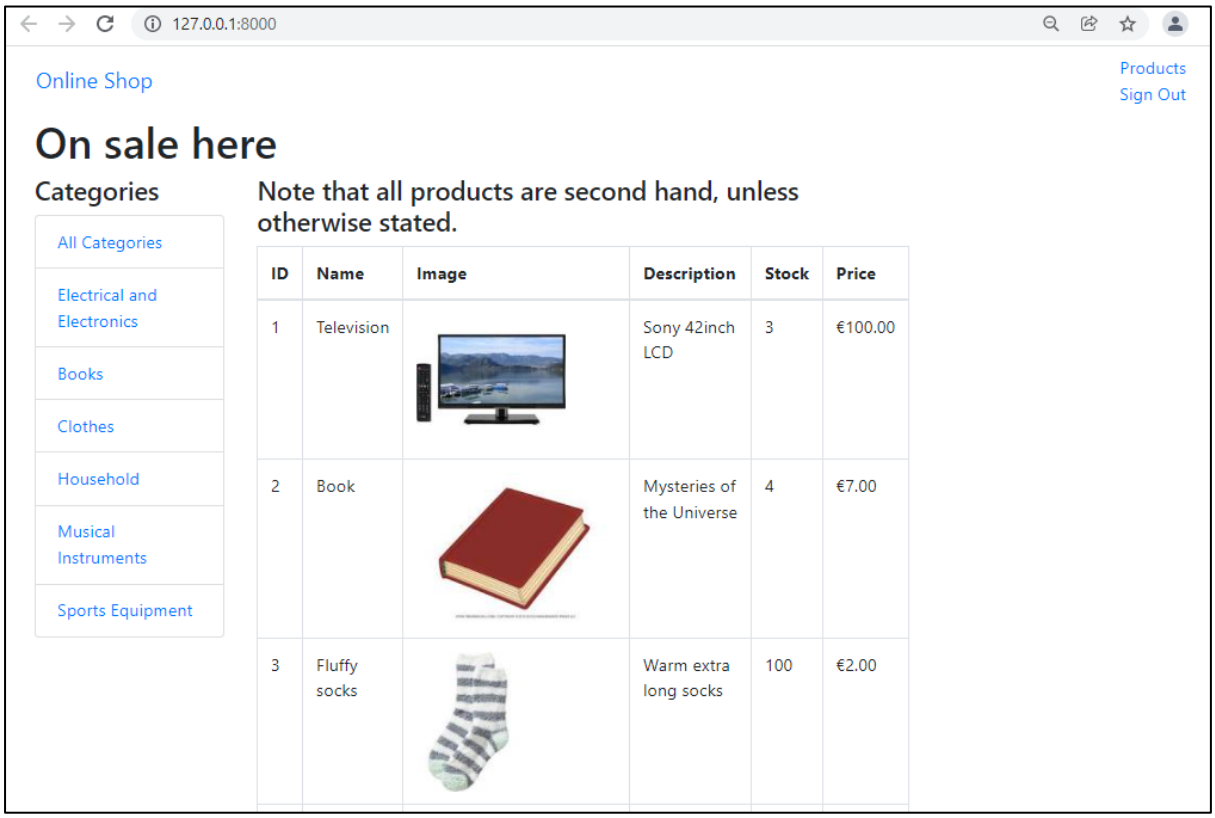
5. In Windows Command line make sure that you are in the djangoprojects folder and type the suitable command to activate the virtual environment:

6. Use the git clone command to clone the repo to your local computer:

7. Move into this lab-12-username folder using the cd command.

8. Open VS Code and open the project folder & create a **static** folder at the project root level

In Windows Command Line, run the server and open the home page using http://127.0.0.1:8080 and you should see the following page:



This project displays a list of products and the category to which they belong. This is a One to Many relationship. The category links on the left-hand side can be used to

view the products in each category. In this part of the exercise, we will change the One to Many mapping (Foreign key) between Category and Product to a Many to Many mapping. This will allow us to add the same product to more than one category.

Open **shop/models.py** and change line 18 in the Product class to the code shown below

```
15  class Product(models.Model):
16      name = models.TextField()
17      image = models.ImageField(upload_to='product_images', blank=True)
18      category = models.ManyToManyField(Category)
19      description = models.TextField()
20      stock = models.IntegerField()
21      price = models.DecimalField(max_digits=10,decimal_places=2)
```

Database Migrations

Run migrations using the following commands:

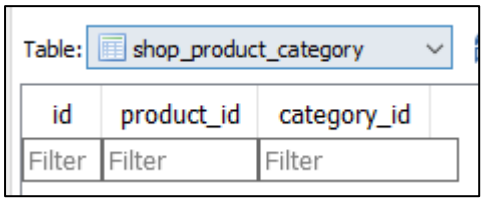
---

python manage.py makemigrations shop

---

python manage.py migrate

If you open the database file in DB Browser you will see a new associative table called shop_product_category. The structure of the table is shown below:



We will populate this with data using the SQL INSERT statement.

To run sql commands in Django we create an empty **migrations** file and add the SQL statement into it.

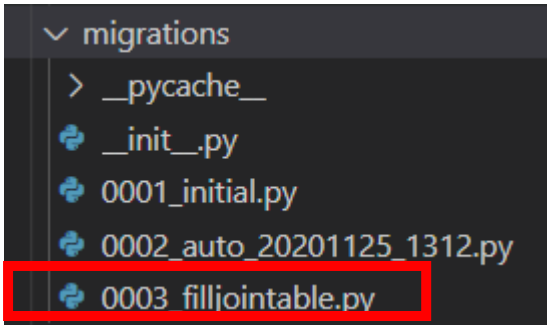
Create an empty migration file using the following command:

---

python manage.py makemigrations shop --empty -n filljointable

---

This command creates a migrations file which you can see in the migrations folder:



Open this migrations file and you will see the following content:

shop > migrations > 0003_filljointable.py

```
1   # Generated by Django 3.1.3 on 2020-11-25 13:15
2
3   from django.db import migrations
4
5
6   class Migration(migrations.Migration):
7
8       dependencies = [
9           ('shop', '0002_auto_20201125_1312'),
10      ]
11
12      operations = [
13      ]
```

We can place our SQL code inside the **operations**

Open the file **operations.txt** on Moodle and copy the contents and paste them into this file – make sure to overwrite lines 12 and 13

Run the migrate command as shown below:

---

python manage.py migrate

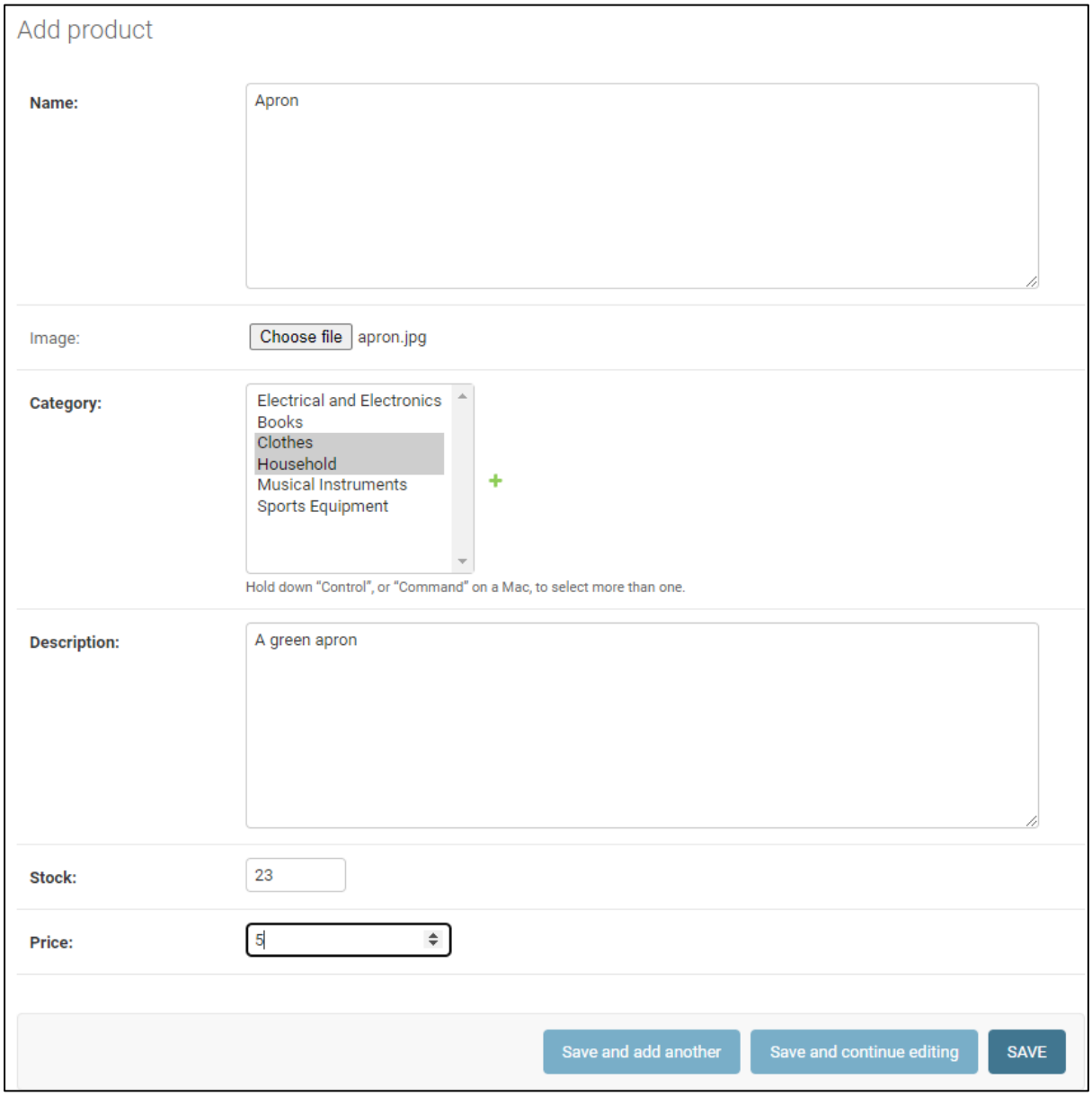
Register the Product & Category models in admin.py

Create a superuser account

Run the server

Log into admin

Add a new product called apron and upload the image apron.jpeg. This image is available on Moodle. In the Category section select both Clothes & Household using the Ctrl key to select both. Click **Save** to save the new object.



Refresh the browser and view the products for the Household & Clothes categories and you should see the apron in both categories:

Clothes

Household

Image Thumbnails

The product images are a bit too big for the table so we will look at creating thumbnail images instead. We can create thumbnail images from our existing images using a Django app called **ImageKit**.

ImageKit is a Django app for processing images. It comes with a set of image processors for common tasks like resizing and cropping. We will use it to create thumbnails for our images.

Use the following command to install Imagekit:

---

pip install django-imagekit

Register the new pp in **settings.py**:

```
41        'imagekit',
```

Open **shop/models.py** and add the following imports:

```
3   from imagekit.models import ImageSpecField # import this
4   from imagekit.processors import ResizeToFill # import this
```

Line 4: The imagekit.processors module contains processors for many common image manipulations, like resizing, rotating, and color adjustments.

Add a new field to the Product model class using the code below:

```
24      image_thumbnail = ImageSpecField(source='image',
25                                       processors=[ResizeToFill(70,70)],
26                                       format='JPEG',
27                                       options={'quality':60})
```

Template

Change the link in the products.html page to use the image_thumbnail field:

```
37                  <td><img src="{{product.image_thumbnail.url}}" alt =
```

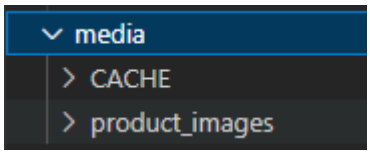
Run the server & you should see the smaller thumbnail images on the home page.

Note that all products are second hand, unless otherwise stated.

ID	Name	Image	Description	Stock	Price
1	Television		Sony 42inch LCD	3	€100.00
2	Book		Mysteries of the Universe	4	€7.00
3	Fluffy socks		Warm extra long socks	100	€2.00

By default, ImageKit uses the approach of generating thumbnails from source images the first time they're requested and storing them. This way, the expensive generation only must happen once, and all subsequent requests can then simply serve the previously generated file. Additionally, it caches a reference to the thumbnail, so that it doesn't even have to access your storage unless the cache is cleared.

If you look in the media folder, you will see a new folder structure as shown below where a cached folder has been created with subfolders for each image.



Stop the server and run the following git commands to update the local and remote repositories:

git add -A

git commit -m "lab 12 commit"

git push -u origin main