



## Overview

The Python Imaging Library adds image processing capabilities to your Python interpreter.

This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

1. Go to [Pillow documentation](#) and install the Django package in the Django server terminal and in Celery terminal

```
$ pip install pillow
```

2. In the Django server, we create a new app IMAGE\_COMPRESSION

```
$ python manage.py startapp image_compression
```

3. Register the new app in SETTINGS.PY INSTALLED\_APPS

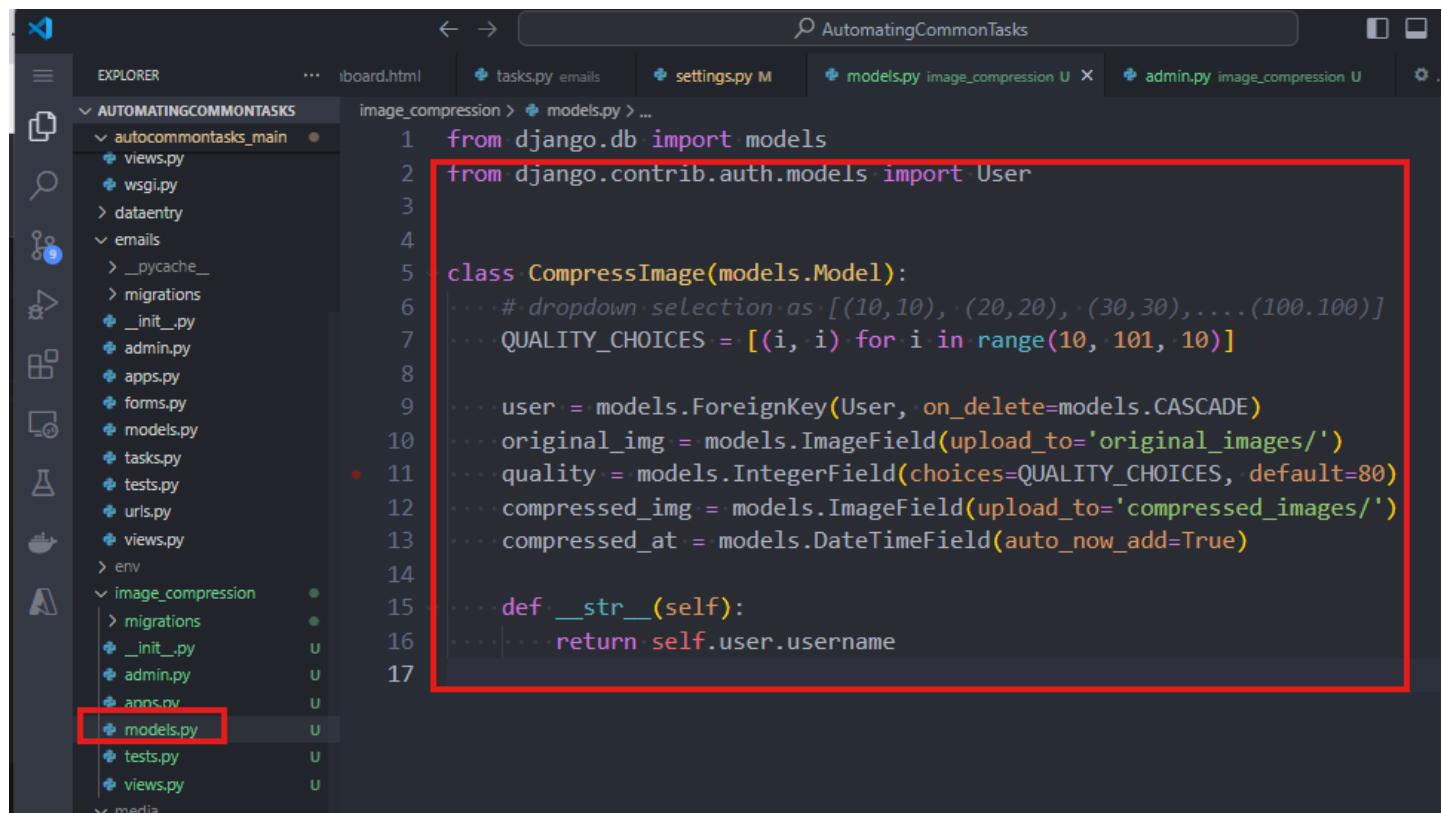
```

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'dataentry',
    'uploads',
    'crispy_forms',
    'crispy_bootstrap5',
    'emails',
    'ckeditor',
    'anymail',
    'image_compression',
]

```

4. Create the model in MODELS.PY



```

from django.db import models
from django.contrib.auth.models import User

class CompressImage(models.Model):
    # dropdown selection as [(10,10), (20,20), (30,30), ... (100,100)]
    QUALITY_CHOICES = [(i, i) for i in range(10, 101, 10)]

    user = models.ForeignKey(User, on_delete=models.CASCADE)
    original_img = models.ImageField(upload_to='original_images/')
    quality = models.IntegerField(choices=QUALITY_CHOICES, default=80)
    compressed_img = models.ImageField(upload_to='compressed_images/')
    compressed_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.user.username

```

To make the QUALITY field a drop-down option, we use the SELECT TAG. Where we see the VALUE AND THE LABEL (which is displayed as a user option in the dropdown)

```

<!DOCTYPE html>
<html>
<body>

<h1>The select element</h1>

<p>The select element is used to create a drop-down list.</p>

<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select name="cars" id="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="opel">Opel</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit" value="Submit">
</form>

<p>Click the "Submit" button and the form-data will be sent to a page on the server called "action_page.php".</p>

</body>
</html>

```

## The select element

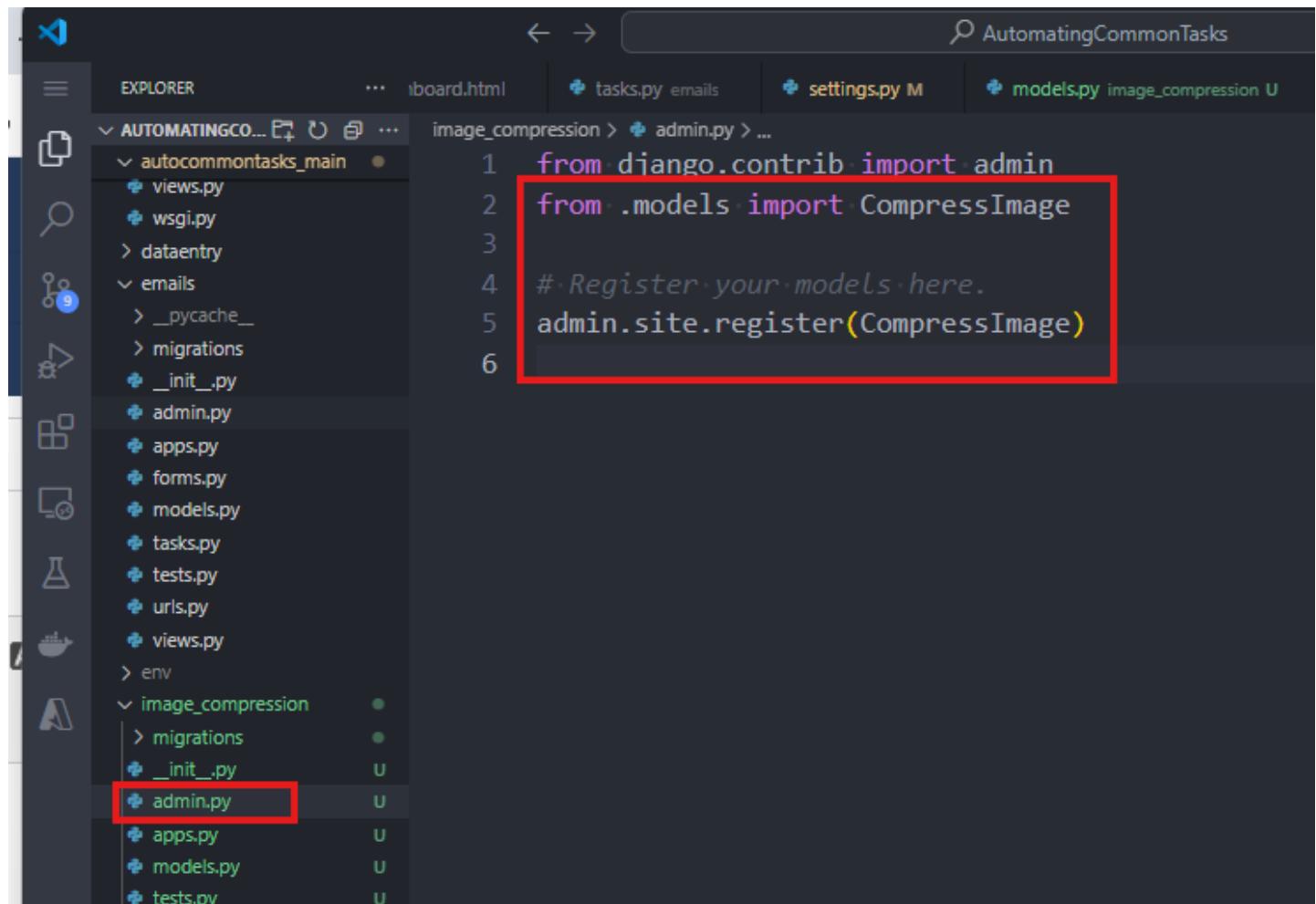
The select element is used to create a drop-down list.

Choose a car:

Submit

Click the "Submit" button and the form-data will be sent to a page on the server.

5. Register the model for our ADMIN panel. Update ADMIN.PY:



```

1  from django.contrib import admin
2  from .models import CompressImage
3
4  # Register your models here.
5  admin.site.register(CompressImage)
6

```

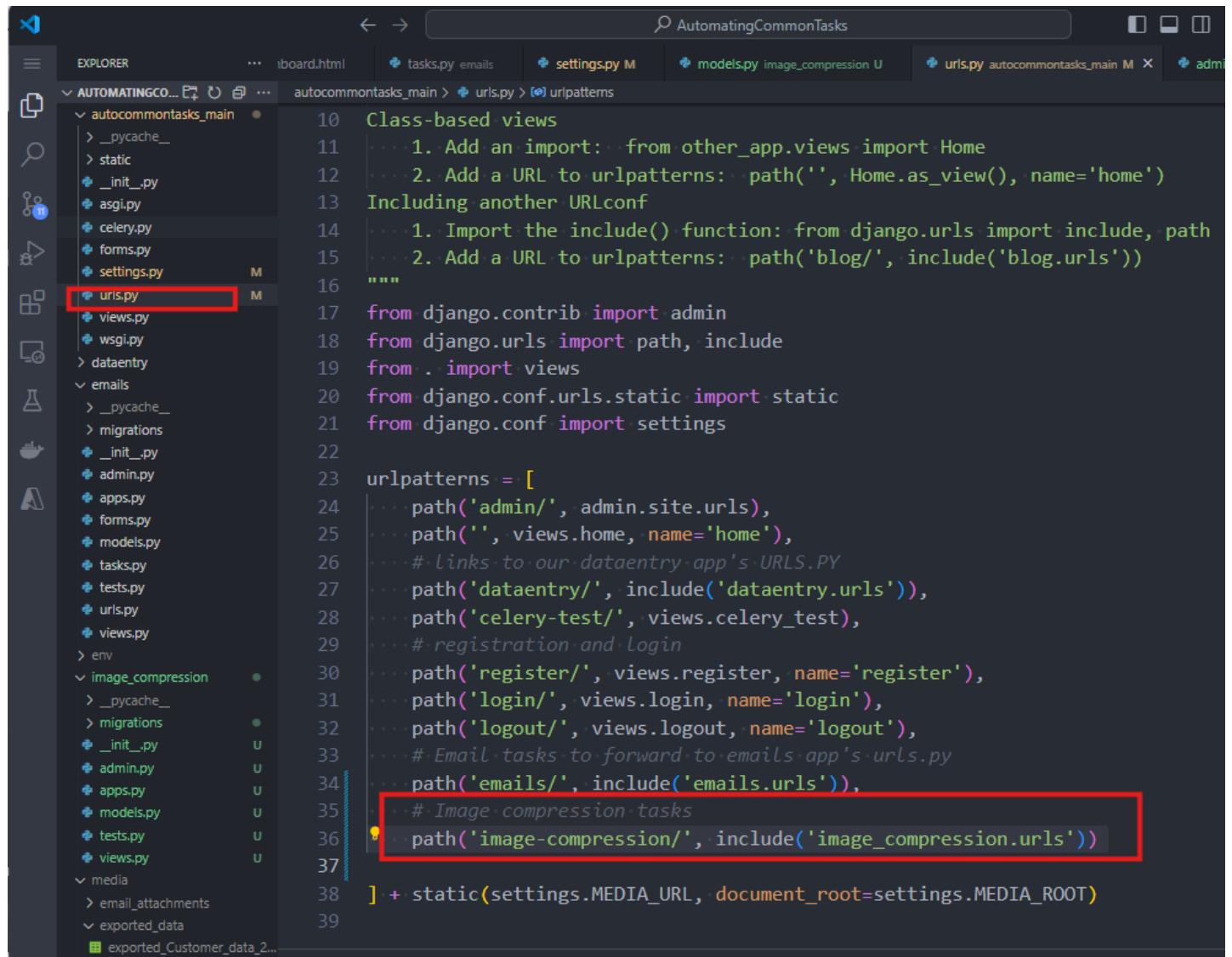
6. Make the necessary model migrations

```

$ python manage.py makemigrations
$ python manage.py migrate

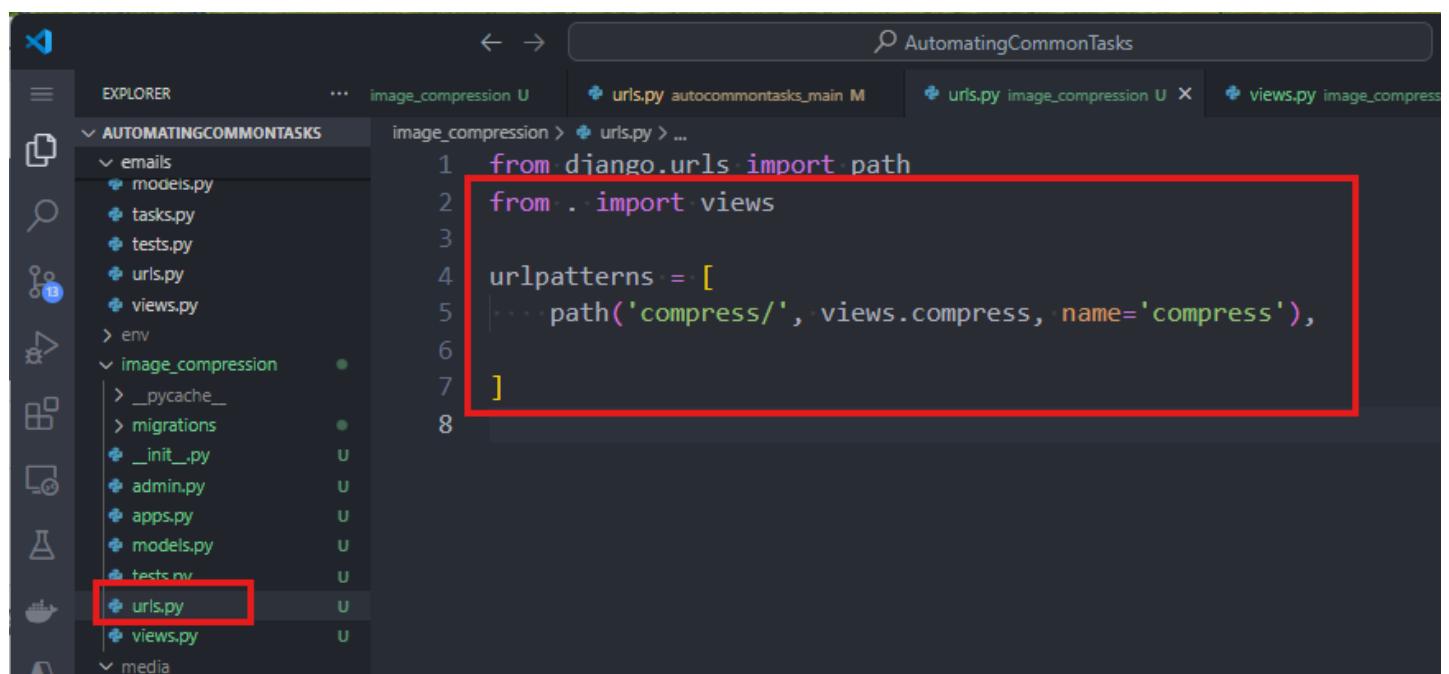
```

7. Create the new URL pattern. Since we have a new app, we need to create a new pattern in our main project's URLs.PY.



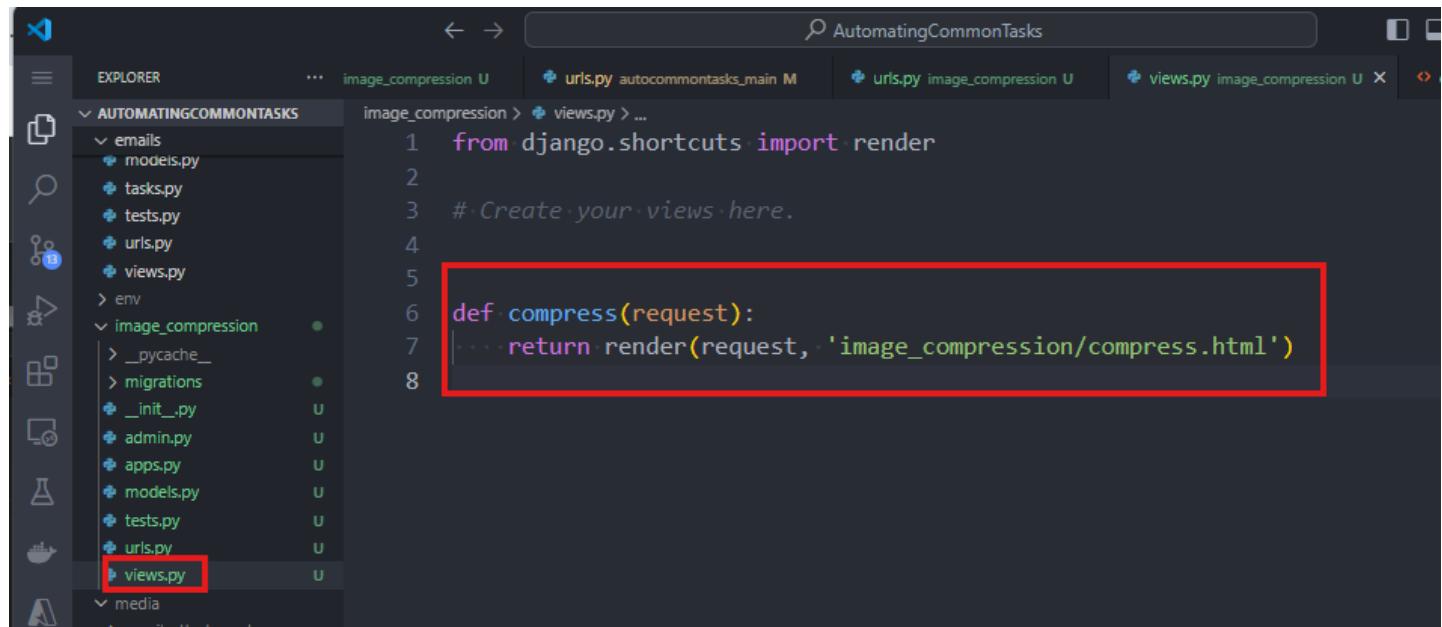
```
10 Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19 from . import views
20 from django.conf.urls.static import static
21 from django.conf import settings
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25     path('', views.home, name='home'),
26     # Links to our dataentry app's URLs
27     path('dataentry/', include('dataentry.urls')),
28     path('celery-test/', views.celery_test),
29     # registration and login
30     path('register/', views.register, name='register'),
31     path('login/', views.login, name='login'),
32     path('logout/', views.logout, name='logout'),
33     # Email tasks to forward to emails app's urls.py
34     path('emails/', include('emails.urls')),
35     # Image compression tasks
36     path('image-compression/', include('image_compression.urls'))
37 ]
38 + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
39
```

Then, create a new URLs.PY file in our new app for image-compression-related URL paths.



```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('compress/', views.compress, name='compress'),
6 ]
7
8
```

8. Create the function in the VIEWS.PY



EXPLORER

AUTOMATINGCOMMONTASKS

- emails
- models.py
- tasks.py
- tests.py
- urls.py
- views.py

env

image\_compression

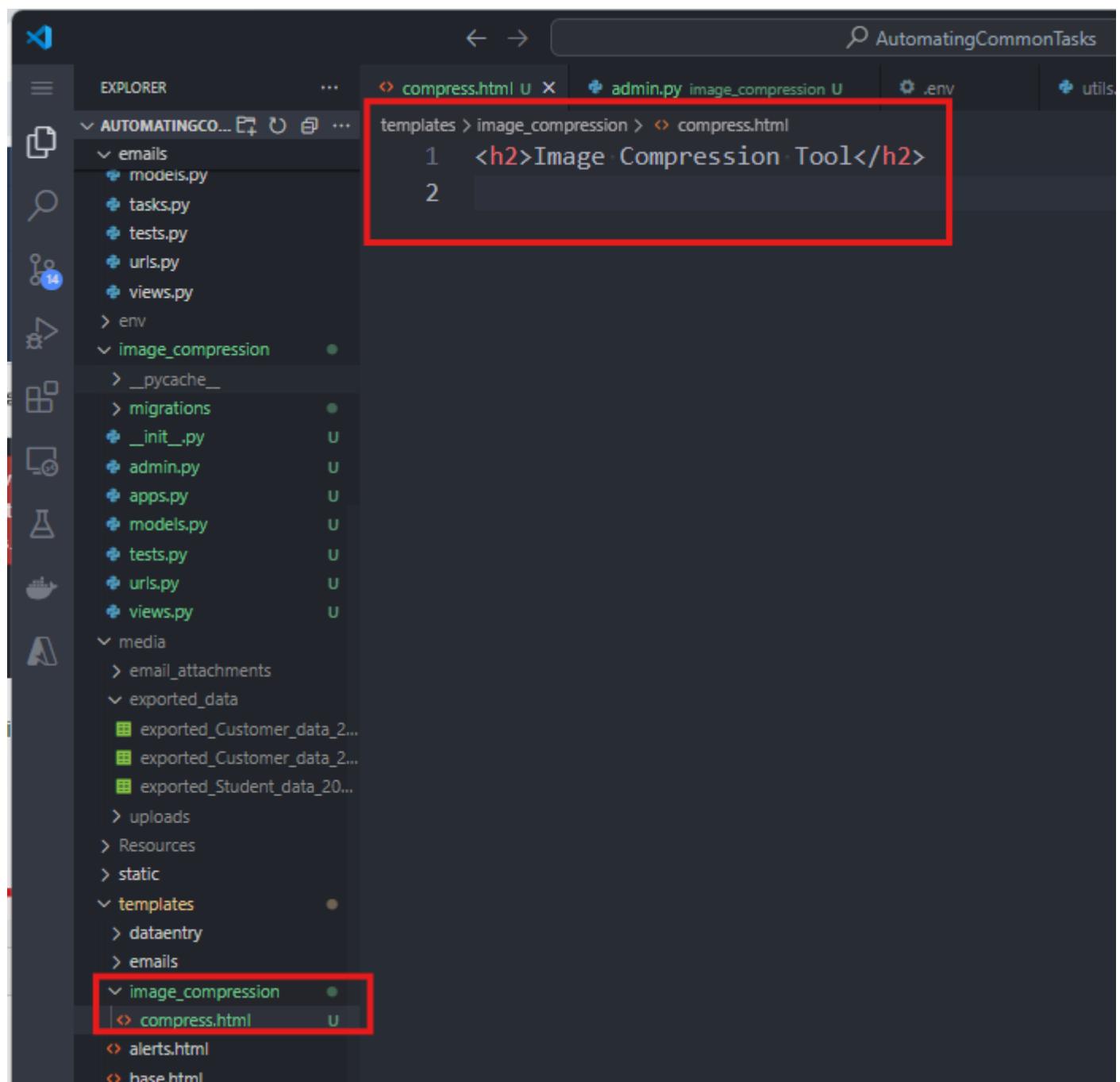
- \_\_pycache\_\_
- migrations
- \_\_init\_\_.py
- admin.py
- apps.py
- models.py
- tests.py
- urls.py
- views.py

media

views.py

```
1 from django.shortcuts import render
2
3 # Create your views here.
4
5
6 def compress(request):
7     return render(request, 'image_compression/compress.html')
8
```

9. Create a new FOLDER called IMAGE\_COMPRESSION, and in this folder, create a new file, COMPRESS.HTML



EXPLORER

AUTOMATINGCOMMONTASKS

compress.html

admin.py

.env

utils

templates > image\_compression > compress.html

```
1 <h2>Image Compression Tool</h2>
2
```

emails

models.py

tasks.py

tests.py

urls.py

views.py

env

image\_compression

\_\_pycache\_\_

migrations

\_\_init\_\_.py

admin.py

apps.py

models.py

tests.py

urls.py

views.py

media

email\_attachments

exported\_data

- exported\_Customer\_data\_2...
- exported\_Customer\_data\_2...
- exported\_Customer\_data\_2...

uploads

Resources

static

templates

dataentry

emails

image\_compression

compress.html

alerts.html

base.html

10. Update the HOME.HTML to call this new web page.

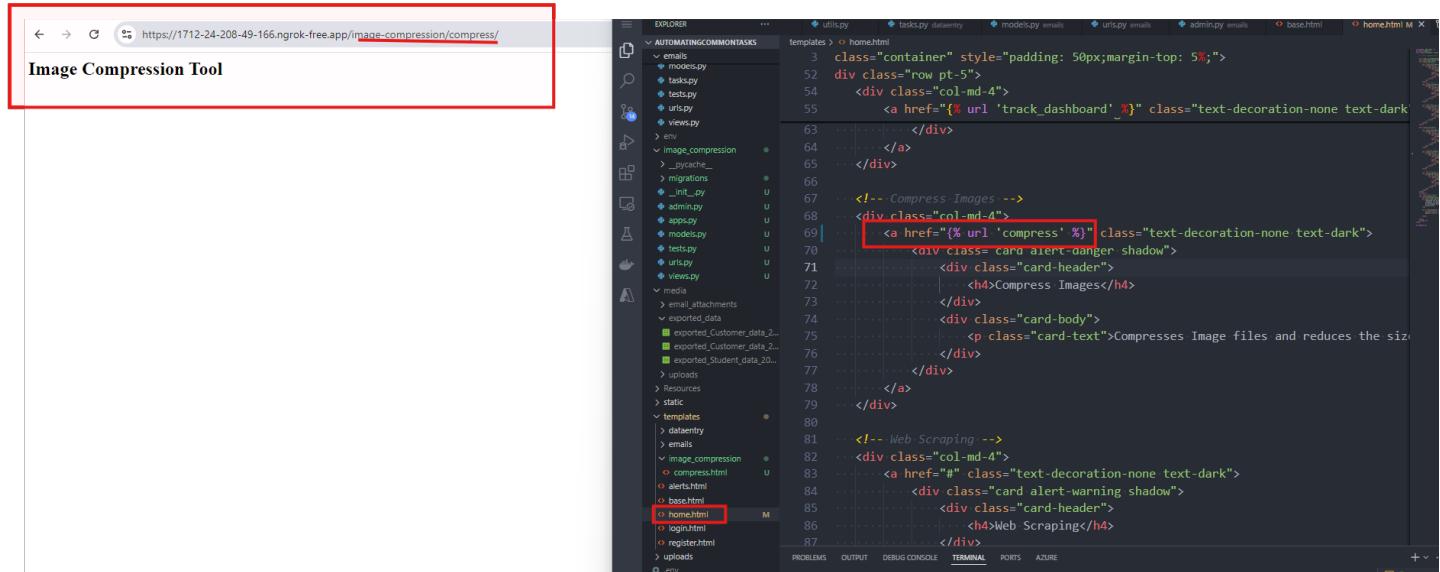
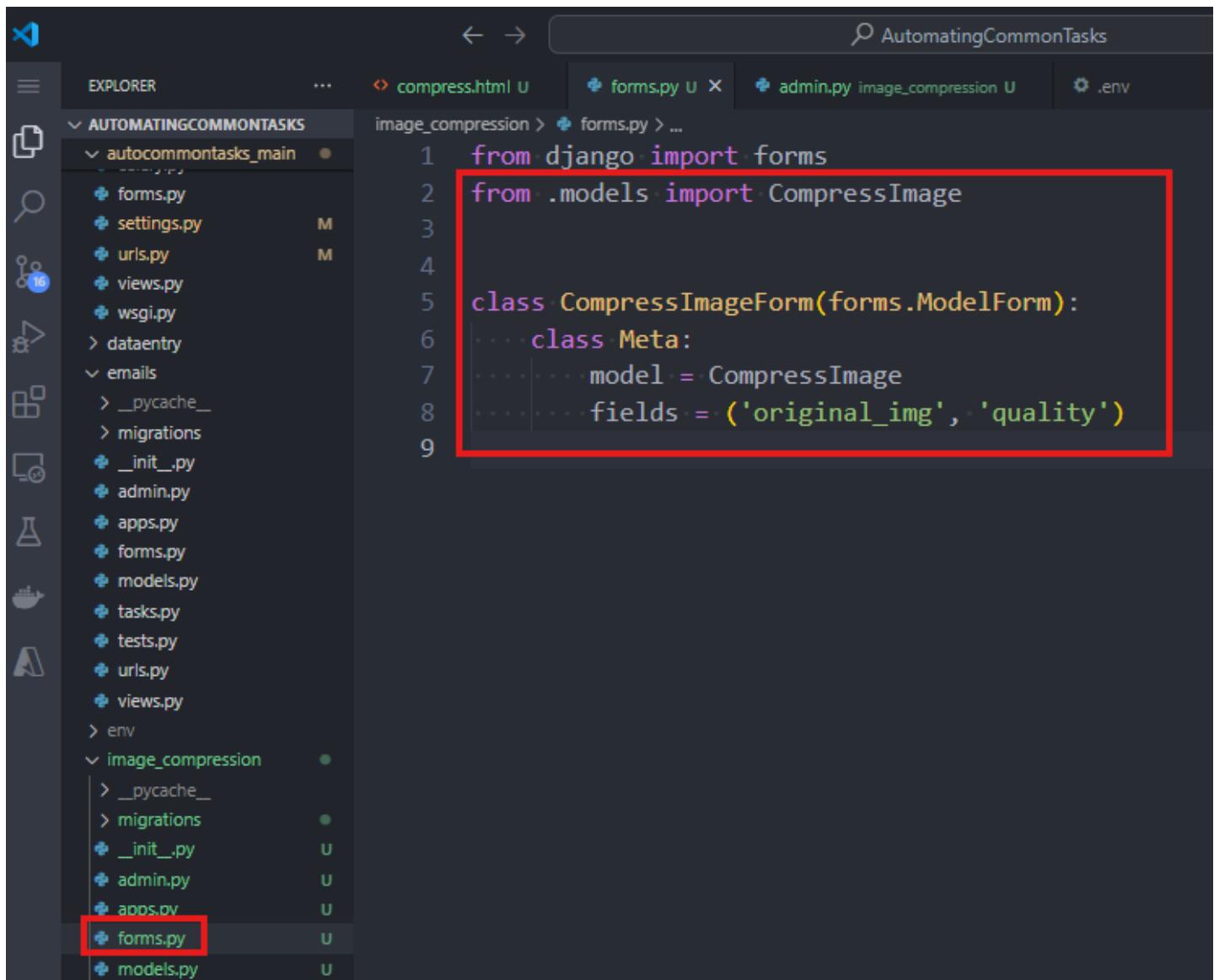


Image Compression Tool

```
EXPLORER          HOME.HTML
AUTOMATINGCOMMONTASKS  templates > home.html
  emails
  models.py
  migrations
  tasks.py
  urls.py
  views.py
  > image_compression
    > __pycache__
    > migrations
    > __init__.py
    > admin.py
    > apps.py
    > models.py
    > tasks.py
    > urls.py
    > views.py
    > uploads
    > resources
    > static
    > templates
      > dataentry
      > emails
      > image_compression
        compress.html
        alerts.html
        base.html
        home.html
        register.html
        uploads
        env
  > env
```

```
home.html
1  <div class="container" style="padding: 50px; margin-top: 5%;>
2   <div class="row pt-5">
3     <div class="col-md-4">
4       <a href="{% url 'track_dashboard' %}" class="text-decoration-none text-dark">
5         <div class="card alert-danger shadow">
6           <div class="card-header">
7             <h4>Compress Images</h4>
8           <div class="card-body">
9             <p>Compresses Image files and reduces the size</p>
10            </div>
11          </div>
12        </a>
13      </div>
14    </div>
15    <!-- Compress Images -->
16    <div class="col-md-4">
17      <a href="{% url 'compress' %}" class="text-decoration-none text-dark">
18        <div class="card alert-warning shadow">
19          <div class="card-header">
20            <h4>Web Scraping</h4>
21          <div class="card-body">
22            <p>Scrapes data from various websites</p>
23          </div>
24        </div>
25      </a>
26    </div>
27  </div>
28  </div>
29  </div>
30  </div>
31  </div>
32  </div>
33  </div>
34  </div>
35  </div>
36  </div>
37  </div>
38  </div>
39  </div>
40  </div>
41  </div>
42  </div>
43  </div>
44  </div>
45  </div>
46  </div>
47  </div>
48  </div>
49  </div>
50  </div>
51  </div>
52  </div>
53  </div>
54  </div>
55  </div>
56  </div>
57  </div>
58  </div>
59  </div>
60  </div>
61  </div>
62  </div>
63  </div>
64  </div>
65  </div>
66  </div>
67  </div>
68  </div>
69  </div>
70  </div>
71  </div>
72  </div>
73  </div>
74  </div>
75  </div>
76  </div>
77  </div>
78  </div>
79  </div>
80  </div>
81  </div>
82  </div>
83  </div>
84  </div>
85  </div>
86  </div>
87  </div>
```

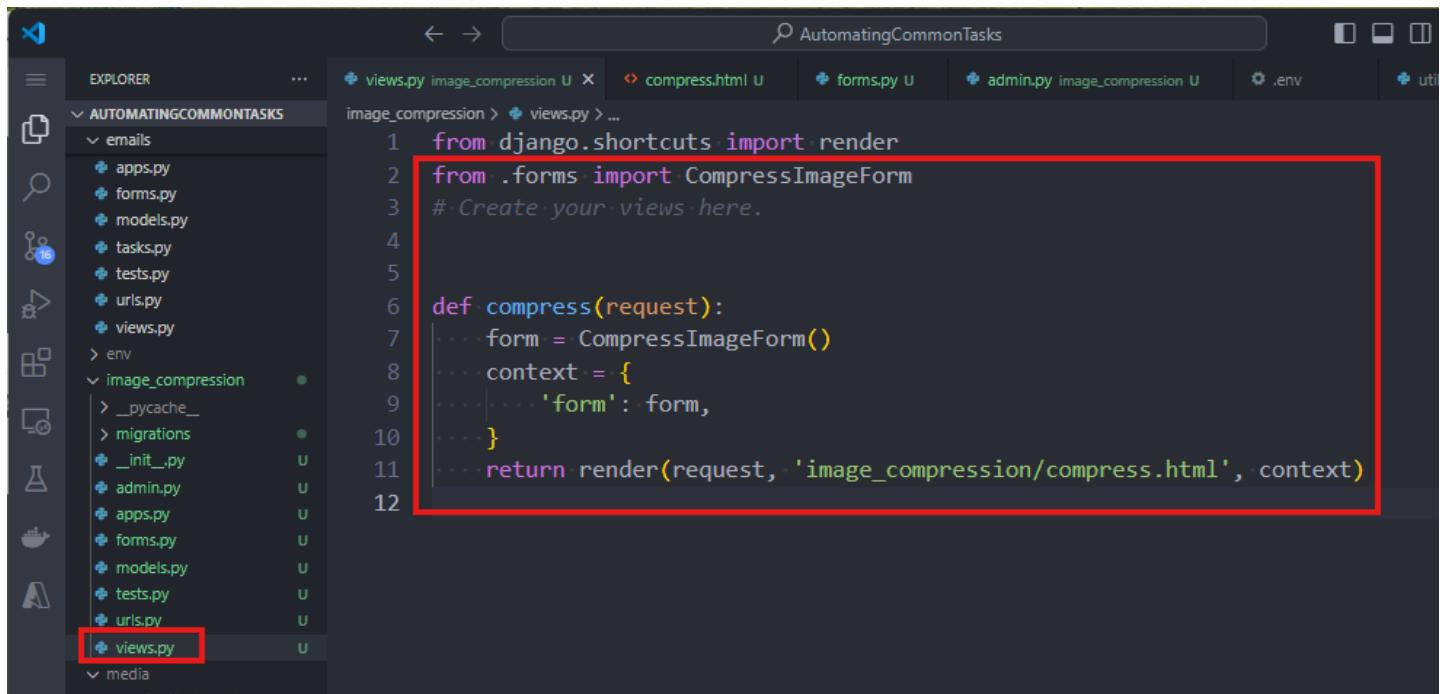
11. Create a FORMS.PY



```
EXPLORER          compress.html U  forms.py U X  admin.py image_compression U  .env
AUTOMATINGCOMMONTASKS  image_compression > forms.py > ...
  > __pycache__
  > migrations
  > __init__.py
  > admin.py
  > apps.py
  > forms.py
  > models.py
  > tasks.py
  > tests.py
  > urls.py
  > views.py
  > uploads
  > resources
  > static
  > templates
    > dataentry
    > emails
    > image_compression
      forms.py
      models.py
      compress.html
      alerts.html
      base.html
      register.html
      uploads
      env
```

```
forms.py
1  from django import forms
2  from .models import CompressImage
3
4
5  class CompressImageForm(forms.ModelForm):
6      class Meta:
7          model = CompressImage
8          fields = ('original_img', 'quality')
```

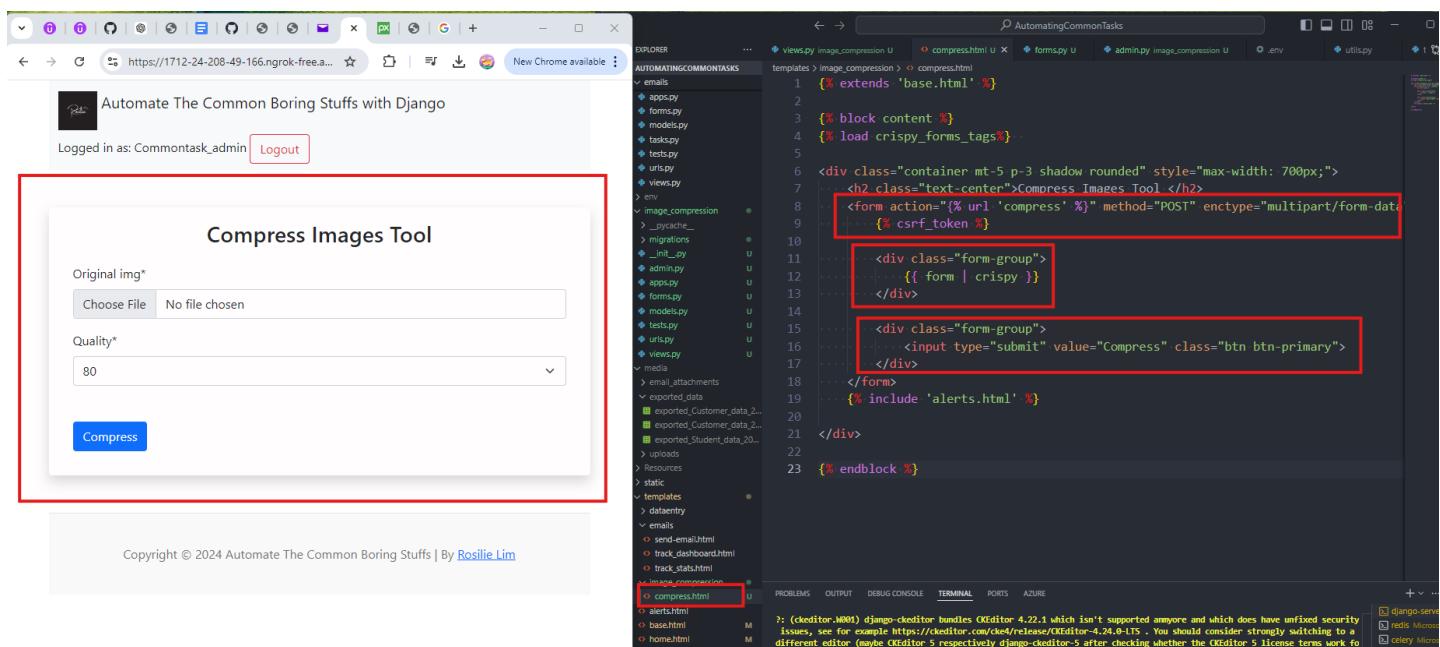
12. Call our form using our VIEWS.PY:



```
from django.shortcuts import render
from .forms import CompressImageForm
# Create your views here.

def compress(request):
    form = CompressImageForm()
    context = {
        'form': form,
    }
    return render(request, 'image_compression/compress.html', context)
```

13. In the COMPRESS.HTML, update as:



Automate The Common Boring Stuffs with Django

Logged in as: Commontask\_admin [Logout](#)

## Compress Images Tool

Original img\*

Choose File  No file chosen

Quality\*

80

Compress

```
{% extends 'base.html' %}

{% block content %}
    {% load crispy_forms_tags %}

## Compress Images Tool


            {% csrf_token %}

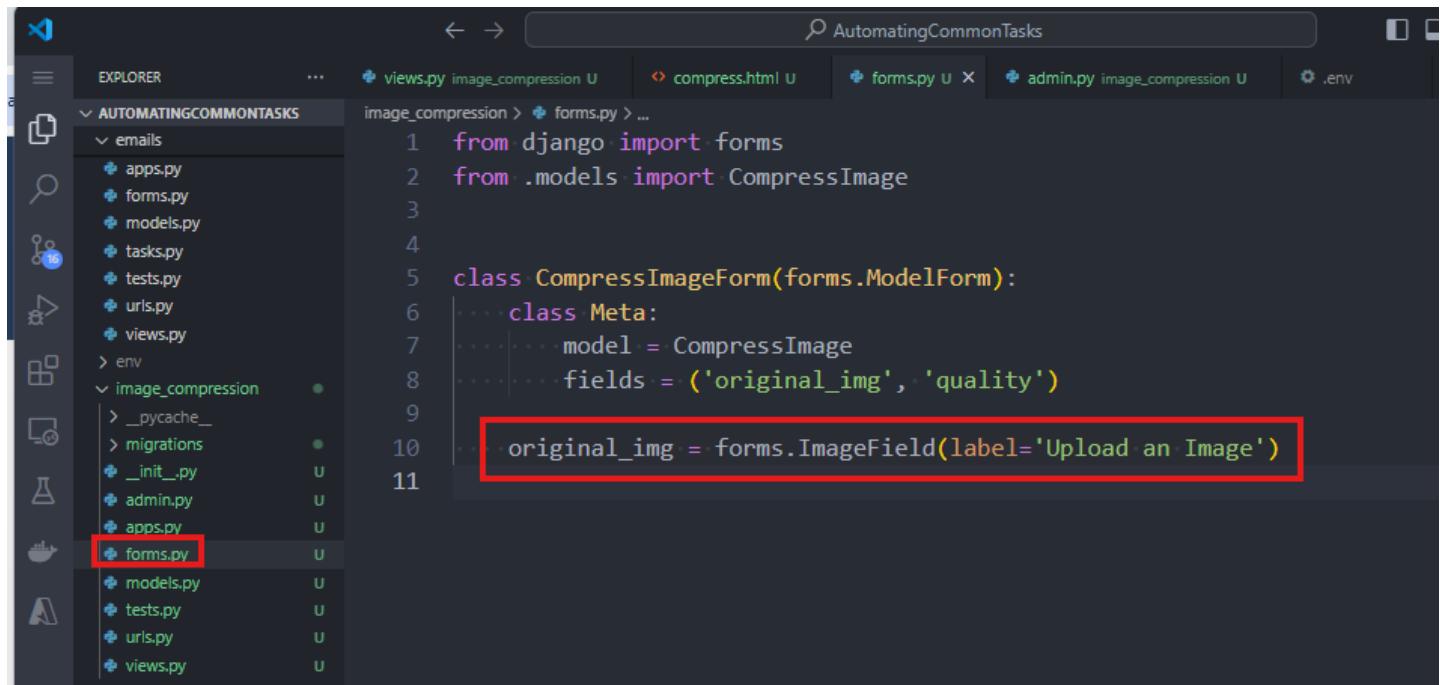
            

{{ form | crispy }}


        {% include 'alerts.html' %}


{% endblock %}
```

14. We can change the label of the field on the form, so update FORMS.PY and add the line:



```
 1  from django import forms
 2  from .models import CompressImage
 3
 4
 5  class CompressImageForm(forms.ModelForm):
 6      class Meta:
 7          model = CompressImage
 8          fields = ('original_img', 'quality')
 9
10      original_img = forms.ImageField(label='Upload an Image')
11
```

FROM:

Original img\*

Choose File No file chosen

Quality\*

80

TO:

Upload an Image\*

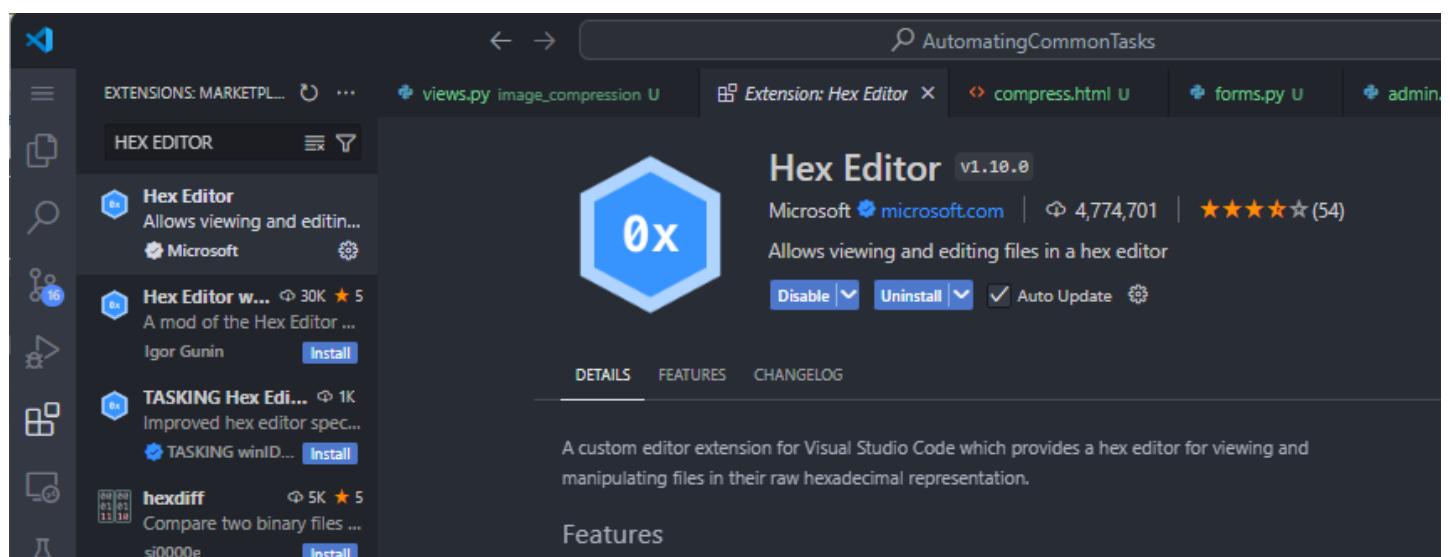
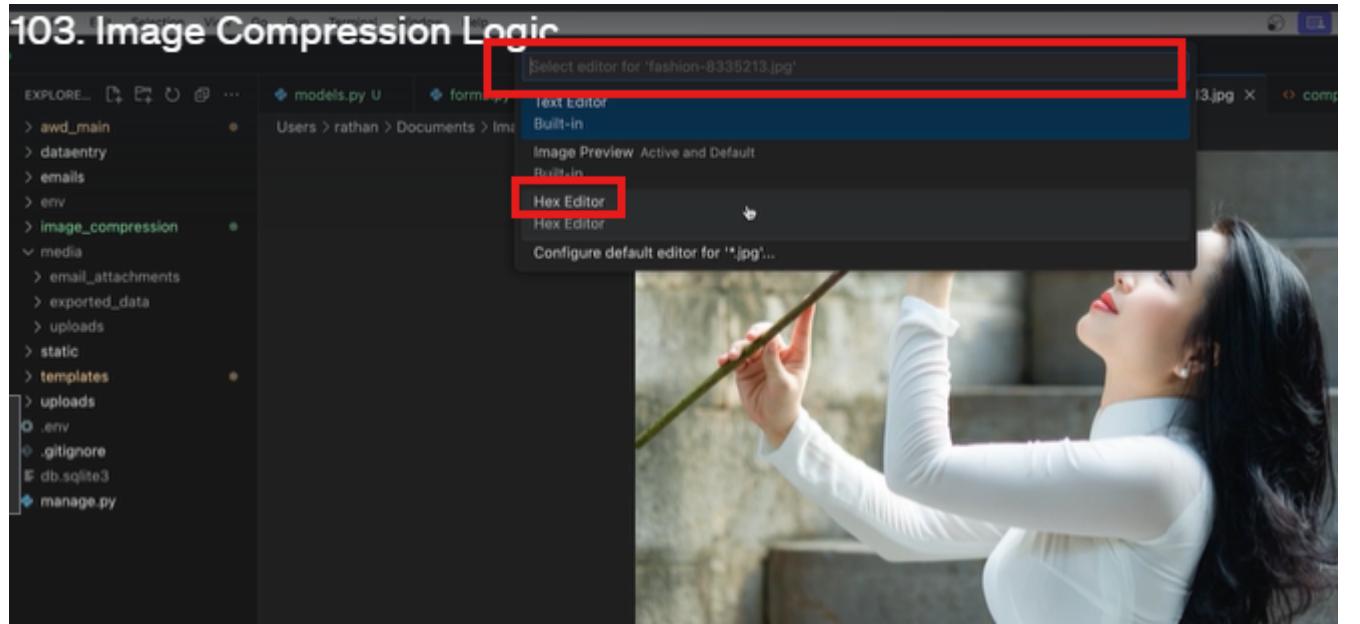
Choose File No file chosen

Quality\*

80

15. When we use `io.BytesIO`, we get bytes value of the image. To see what is the visual representation of these bytes, you can add the EXTENSION 'HEX EDITOR'.

Now open an image file. On the tab of this image, right-click, select 'REOPEN EDITOR WITH', and select HEXEDITOR and you will see the HEX value of the file.





Resources > Datasets > Image+Compression > Image Compression > butterflies-1127666.jpg

Hex	Decoded Text
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	J F I F
00000000 FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01	
00000010 00 01 00 00 FF DB 00 43 00 05 03 04 04 04 03 05	C
00000020 04 04 04 05 05 05 06 07 0C 08 07 07 07 07 0F 0B	
00000030 0B 09 0C 11 0F 12 12 11 0F 11 11 13 16 1C 17 13	
00000040 14 1A 15 11 11 18 21 18 1A 1D 1D 1F 1F 1F 13 17	!
00000050 22 24 22 1E 24 1C 1E 1F 1E FF DB 00 43 01 05 05	" \$ " . \$
00000060 05 07 06 07 0E 08 08 0E 0E 14 11 14 1E 1E 1E	C
00000070 1E	
00000080 1E	
00000090 1E	
000000A0 00 11 08 0D 80 14 40 03 01 22 00 02 11 01 03 11	@ "
000000B0 01 FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00	
000000C0 00 00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09	
000000D0 0A 0B FF C4 00 B5 10 00 02 01 03 03 02 04 03 05	
000000E0 05 04 04 00 00 01 7D 01 02 03 00 04 11 05 12 21	}
000000F0 31 41 06 13 51 61 07 22 71 14 32 81 91 A1 08 23	! A . Q a . " q . 2 . . #
00000100 42 B1 C1 15 52 D1 F0 24 33 62 72 82 09 0A 16 17	B . . R . . \$ 3 b r . . .
00000110 18 19 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A	% & ' ( ) * 4 5 6 7 8 9 :
00000120 43 44 45 46 47 48 49 4A 53 54 55 56 57 58 59 5A	C D E F G H I J S T U V W X Y Z
00000130 63 64 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A	c d e f g h i j s t u v w x y z
00000140 83 84 85 86 87 88 89 8A 92 93 94 95 96 97 98 99	
00000150 9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7	
00000160 B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5	
00000170 D6 D7 D8 D9 DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1	
00000180 F2 F3 F4 F5 F6 F7 F8 F9 FA FF C4 00 1F 01 00 03	
00000190 01 01 01 01 01 01 01 01 00 00 00 00 00 00 00 01	

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

WARNINGS:

AutomatingCommonTasks: (extension W201) dinero-skedoton handles CKEditor 4.22.1 which isn't supported anymore and which does have undefined security

16. We use `BUFFERSEEK(0)` to make sure that after we save, we set our cursor position back to 0.

AutomatingCommonTasks

```

17     def compress(request):
18         # perform the compression
19         img = Image.open(original_img)
20         buffer = io.BytesIO()
21         print('buffer cursor position or pointer at the beginning = >', buffer.tell())
22
23         img.save(buffer, format='JPEG', quality=quality)
24         # print('buffer =', buffer.getvalue())
25         print('buffer cursor position or pointer after image compression = >', buffer.tell())
26         # pointer goes back to 0 location after saving in the buffer
27         buffer.seek(0)
28         print('buffer cursor position or pointer after return to zero = >', buffer.tell())
29
30         # save the compressed image inside the model with filename format
31         compressed_image.compressed_img.save(
32             f'compressed_{original_img}', buffer
33         )
34         return redirect('compress')
35
36     else:
37         form = CompressImageForm()
38         context = {
39             'form': form,
40         }
41
42         return render(request, 'image_compression/compress.html', context)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

?: (ckeditor.W001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported anymore and which does have unfixed security issues, see for example <https://ckeditor.com/cke4/release>. Current editor (maybe CKEditor 5 respectively django-ckeditor-5 after checking whether the CKEditor 5 license terms work for you) or switch to the non-free CKEditor 4 LTS package. See https://ckeditor.com/cke4/ for more information. A notice has been added by the django-ckeditor developers and we are not affiliated with CKSource and were not involved in the licensing change, so please refrain from complaining to us.

System check identified 1 issue (0 silenced).

August 30, 2024 - 16:35:55

Django version 4.2.14, using settings 'autocommonTasks\_main.settings'

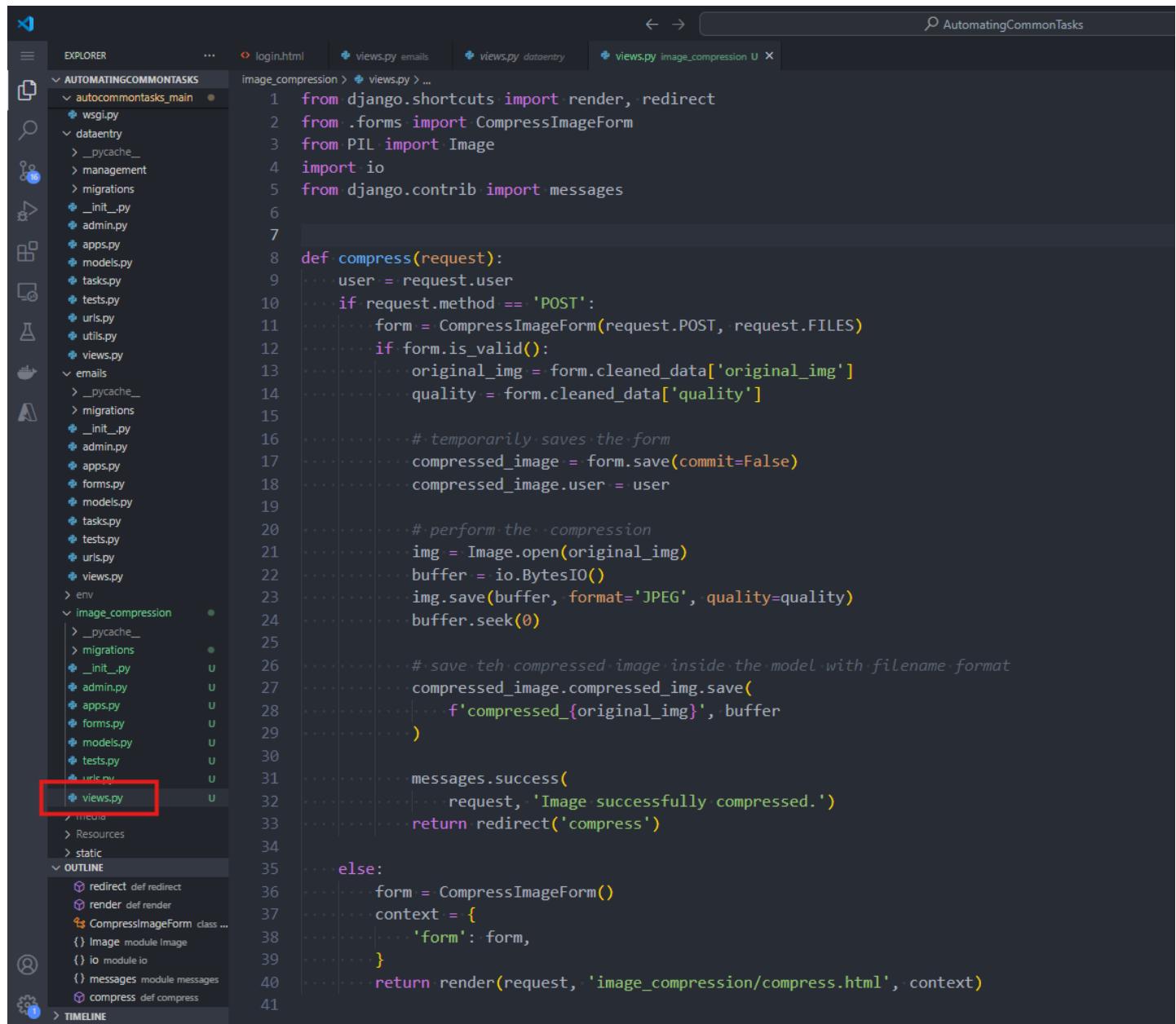
Starting development server at <http://127.0.0.1:8000>

Quit the server with CTRL-BREAK.

buffer cursor position or pointer at the beginning = > 0  
 buffer cursor position or pointer after image compression = > 1226514  
 buffer cursor position or pointer after return to zero = > 0

Internal Server Error: /image-compression/compress/  
 Traceback (most recent call last):  
 File "C:\Users\Rosi\OneDrive\Desktop\LEARNING DJANGO PROJECTS\AutomatingCommonTasks\env\lib\site-packages\django\db\backends\utils.py", line 89, in \_execute

17. The VIEWS.PY shall be:



The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Lists the project structure. A red box highlights the `views.py` file in the `image_compression` app, which is the current file being edited.
- EDITOR** pane: Displays the Python code for the `views.py` file in the `image_compression` app. The code implements a file compression feature using the `CompressImageForm` and `PIL` libraries.
- TOP BAR**: Shows the title "AutomatingCommonTasks" and navigation icons.

```
1 from django.shortcuts import render, redirect
2 from .forms import CompressImageForm
3 from PIL import Image
4 import io
5 from django.contrib import messages
6
7
8 def compress(request):
9     user = request.user
10    if request.method == 'POST':
11        form = CompressImageForm(request.POST, request.FILES)
12        if form.is_valid():
13            original_img = form.cleaned_data['original_img']
14            quality = form.cleaned_data['quality']
15
16            # temporarily saves the form
17            compressed_image = form.save(commit=False)
18            compressed_image.user = user
19
20            # perform the compression
21            img = Image.open(original_img)
22            buffer = io.BytesIO()
23            img.save(buffer, format='JPEG', quality=quality)
24            buffer.seek(0)
25
26            # save the compressed image inside the model with filename format
27            compressed_image.compressed_img.save(
28                f'compressed_{original_img}', buffer
29            )
30
31            messages.success(
32                request, 'Image successfully compressed.')
33            return redirect('compress')
34
35    else:
36        form = CompressImageForm()
37        context = {
38            'form': form,
39        }
40        return render(request, 'image_compression/compress.html', context)
41
```

18. Checking our ADMIN panel

## Django administration

Home > Image\_Compression > Compress images > commontask\_admin

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

DATAENTRY

Customers [+ Add](#)

Employees [+ Add](#)

Students [+ Add](#)

EMAILS

Email trackings [+ Add](#)

Emails [+ Add](#)

Lists [+ Add](#)

Sents [+ Add](#)

Subscribers [+ Add](#)

IMAGE\_COMPRESSION

Compress images [+ Add](#)

UPLOADS

Uploads [+ Add](#)

19. To set the image format to any format not just JPEG, we update our VIEWS.PY AS:

```
... # perform the compression
... img = Image.open(original_img)

... # set the image format based on the uploaded image's format
... output_format = img.format

... buffer = io.BytesIO()
... img.save(buffer, format=output_format, quality=quality)
... buffer.seek(0)

... # save the compressed image inside the model with filename format
```