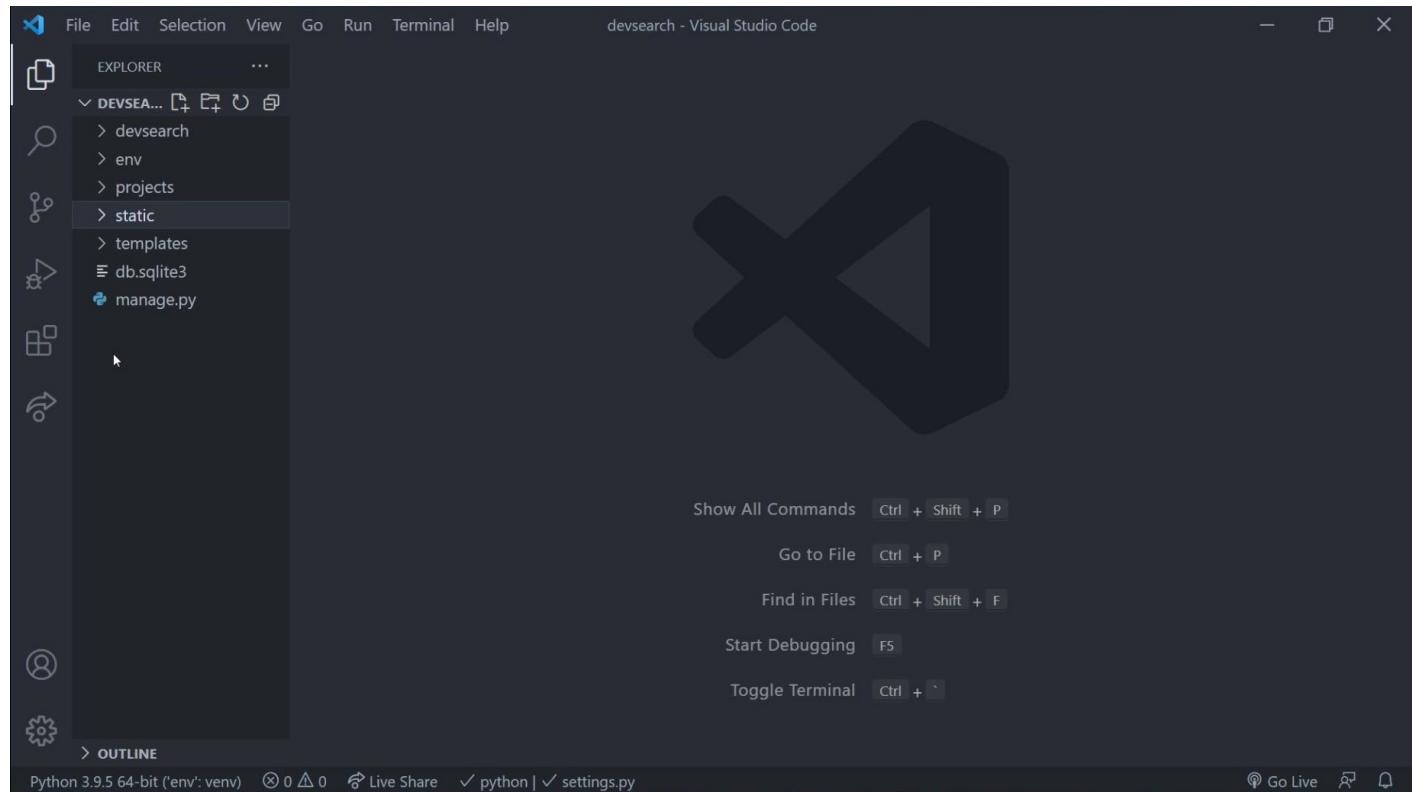


Template Styling and Static Files

Static files in Django projects are all CSS, JavaScript, image file (jpg, tiff, png, gif, icons, svg, etc), and in general, any kind of extrenal file to django.

HTML allows us to style our templates directly. However, this method does not allow code reuse without having to heavily modify each page individually. This is the reason a CSS file is a page styling instructions in a separate file (a static file).

Typically in Django, CSS files are place in a directory at the project level, as shown here:



We named our static file folder "static." For structuring reasons, the static folder also contains different folders named on the basis of the file type they contain:

```
+ static
+---images          # Contains icons, images, logos, svg, etc.
+---js              # Contains javascript files.
+---styles          # Our CSS files usually go here
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** main.css - devsearch - Visual Studio Code.
- Explorer Panel:** Shows a project structure under "DEVSERACH": devsearch, env, projects, static (with images, js, styles), templates, db.sqlite3, manage.py. The "main.css" file is selected.
- Editor Area:** Displays the content of "main.css".

```
static > styles > # main.css
1 |
```
- Bottom Status Bar:** Python 3.9.5 64-bit ('env': venv) ⑧ 0 △ 0 🔍 Live Share ✓ css | ✓ main.css Ln 1, Col 1 Spaces: 4 UTF-8 CRLF CSS ⚙ Go Live ⚙

For the project styling, we will use the file /static/styles/main.css

Just to get an output, we'll type the following code into the main.css file we just created:

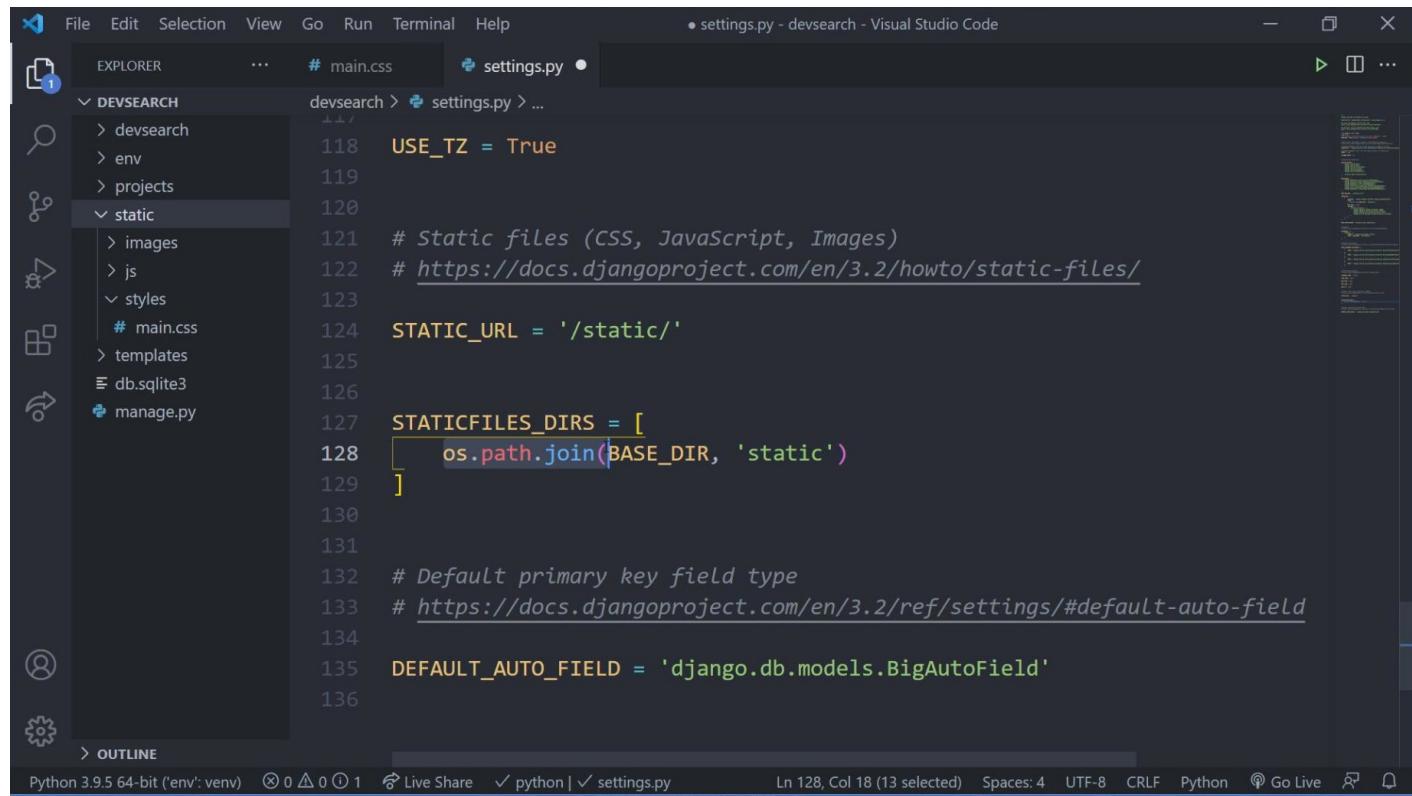
The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** main.css - devsearch - Visual Studio Code.
- Explorer Panel:** Shows a project structure under "DEVSERACH": devsearch, env, projects (selected), static, images, js, styles, templates, db.sqlite3, manage.py. The "main.css" file is selected.
- Editor Area:** Displays the content of "main.css".

```
static > styles > # main.css > p
1 h1{
2   color: red;
3 }
4
5 p{
6   color: blue;
7 }
```
- Bottom Status Bar:** Python 3.9.5 64-bit ('env': venv) ⑧ 0 △ 0 🔍 Live Share ✓ css | ✓ main.css Ln 7, Col 2 Spaces: 4 UTF-8 CRLF CSS ⚙ Go Live ⚙

Next, we need to inform Django where to find our project's static files. For that, we have to include into our /devsite/settings.py file.

At the bottom of the file, just below the line `STATIC_URL = '/static/'`, we have to add the variable `STATICFILES_DIRS`:



The screenshot shows the Visual Studio Code interface with the file `settings.py` open. The code editor displays the following Python code:

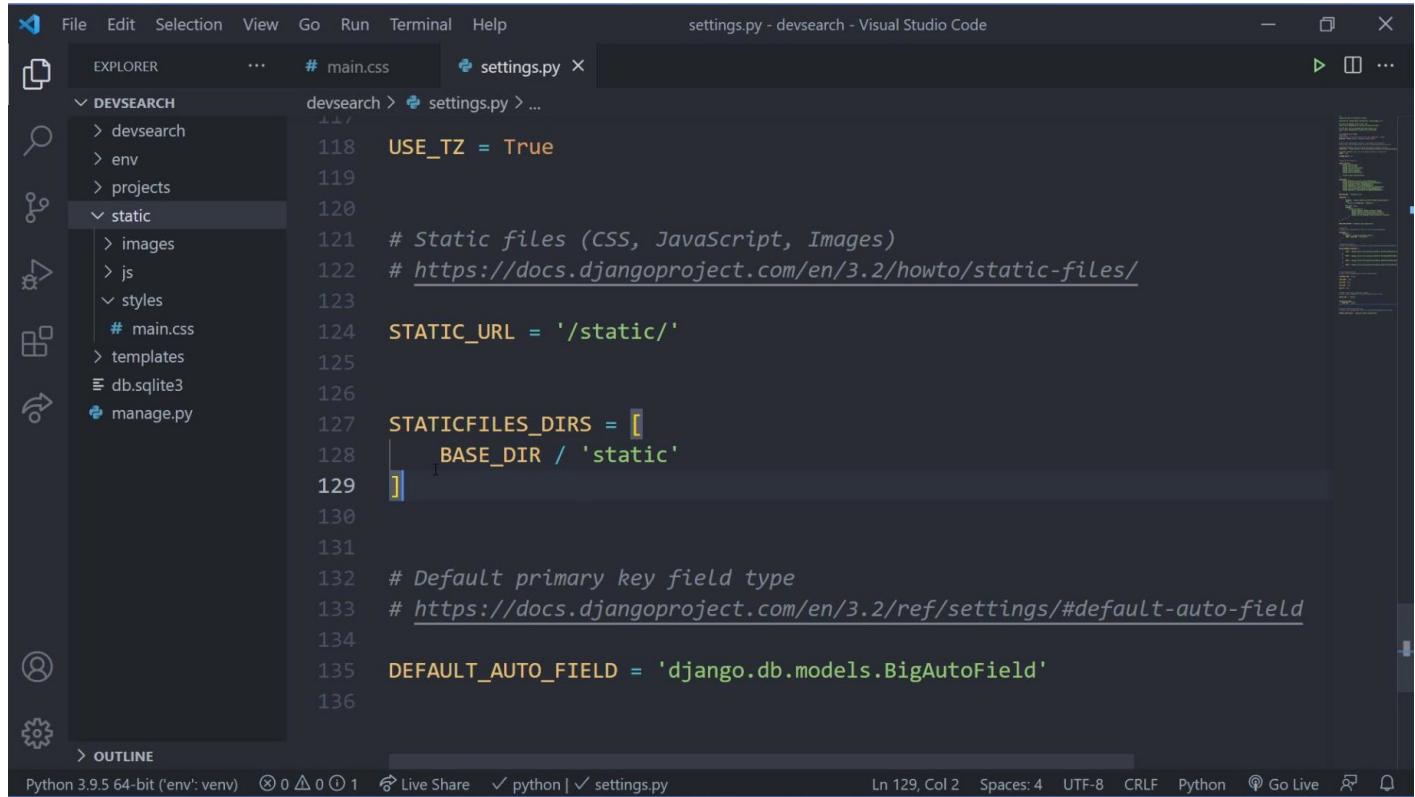
```
118     USE_TZ = True
119
120
121     # Static files (CSS, JavaScript, Images)
122     # https://docs.djangoproject.com/en/3.2/howto/static-files/
123
124     STATIC_URL = '/static/'
125
126
127     STATICFILES_DIRS = [
128         os.path.join(BASE_DIR, 'static')
129     ]
130
131
132     # Default primary key field type
133     # https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
134
135     DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
136
```

The line `STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]` is highlighted with a yellow background. The status bar at the bottom shows the Python version as `Python 3.9.5 64-bit ('env': venv)` and the current file as `Live Share python | settings.py`.

The variable name `STATICFILES_DIRS` is the name defined by the Django documentation, it is not a random variable name. The variable also accepts the path definition using `pathlib.Path`:

```
STATICFILES_DIRS = [
    Path(BASE_DIR, 'static'),
]
```

Another acceptable way to pass the new static path to django is:



The screenshot shows the Visual Studio Code interface with the settings.py file open. The code defines STATIC_URL and STATICFILES_DIRS to point to the static directory. It also sets USE_TZ to True and specifies the default primary key field type.

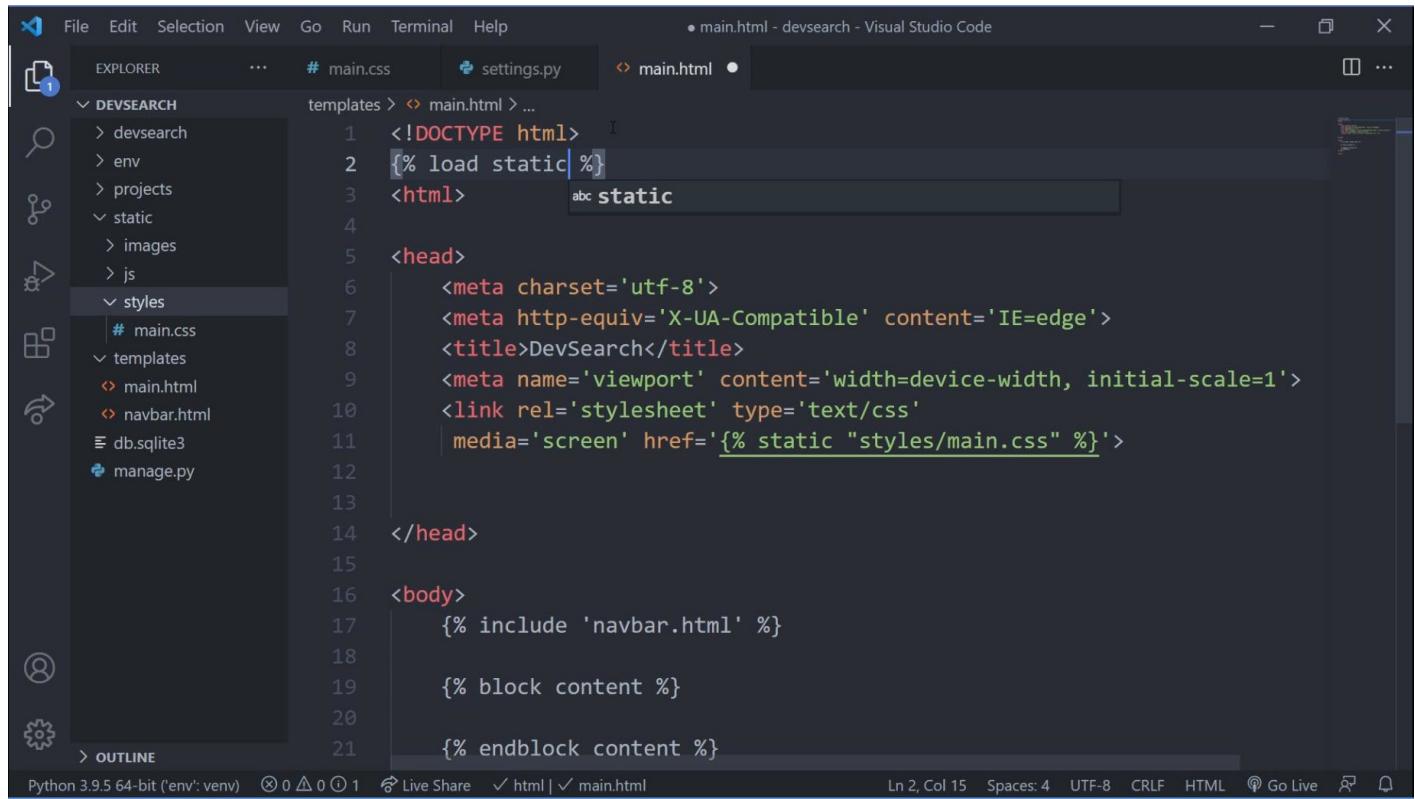
```

USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    BASE_DIR / 'static'
]
# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

but this is valid only in newer django versions.

Next, we need to use our style /static/styles/main.css with our django project. There are two things We need to modify our base html template in /template/main.html:



The screenshot shows the Visual Studio Code interface with the main.html template file open. The template includes the static tag to load the main.css file.

```

<!DOCTYPE html>
{% load static %}
<html>
    <head>
        <meta charset='utf-8'>
        <meta http-equiv='X-UA-Compatible' content='IE=edge'>
        <title>DevSearch</title>
        <meta name='viewport' content='width=device-width, initial-scale=1'>
        <link rel='stylesheet' type='text/css' media='screen' href='{% static "styles/main.css" %}'>
    </head>
    <body>
        {% include 'navbar.html' %}
        {% block content %}
        {% endblock content %}
    </body>

```

First, the template needs to know that it has to use a static file. For that we type in the template:

```
{% load static %}
```

And in order to include the specific styling files we include a variable:

```
href='{% static "styles/main.css" %}'
```

the static is a dynamic direction to the /static/ folder and the "styles/main.css" is the path to the main.css file. This is how it links up our stylesheet to our new "main.css" file in the /static/styles/ folder.

With all these modifications in our application, we should see two differences in our projects page:

The screenshot shows a web browser window with the URL 127.0.0.1:8000. The page title is 'Add Project'. Below it, the word 'Projects' is displayed in a large red font. A table follows, with columns for 'Project', 'Positive Votes', and 'Votes'. The data is as follows:

Project	Positive Votes	Votes
Ecommerce Website	30%	12
Portfolio Website	65%	6
Mumble Social Network	74%	89
Code Sniper	78%	80
Yogo Vive	98%	51

Each row has three links: 'Edit', 'Delete', and 'View'. At the bottom of the page, the word 'FOOTER' is written in blue.

Notice that the Project title page is now red colored and the footer is blue colored when they used to be all black.

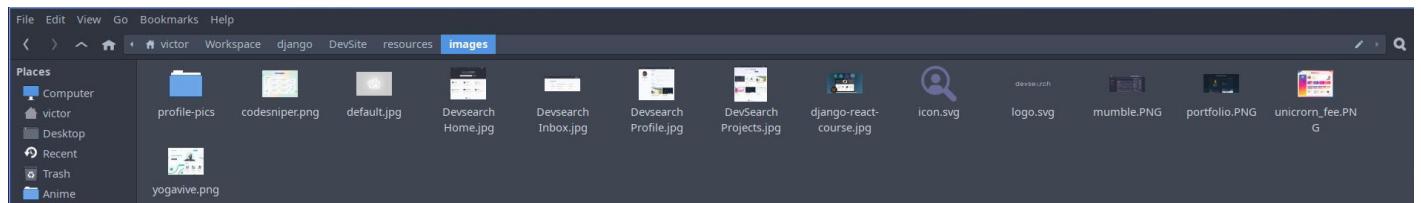
The **h1** in the main.css file refers to the HTML tag for header level 1 and the **p** refers to the HTML tag for paragraph.

All changes are automatic as long we keep using the specific styling with /static/styles/main.css.

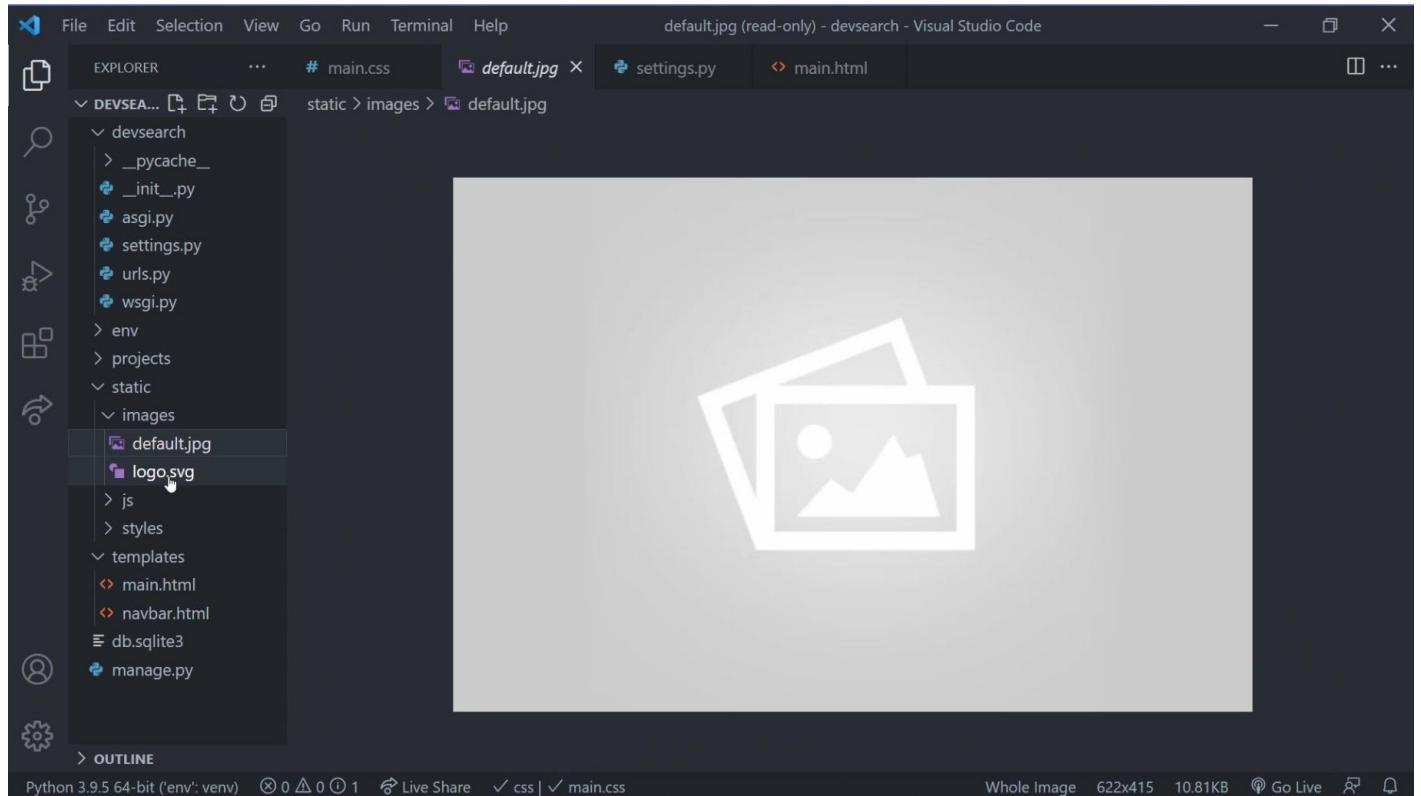
From now onwards, the specific styles in main.css are applied to our templates derived from the main.html template.

Add a logo to the Navbar

Copy the logo.svg and the default.jpg from the resources folder, that you downloaded from the repository, to the /static/images folder:



Place both files into your /static/images folder:



The following step is to modify the /templates/navbar.html file:

```

File Edit Selection View Go Run Terminal Help
navbar.html - devsearch - Visual Studio Code
EXPLORER ... navbar.html x
DEVSEARCH templates > navbar.html > hr
1  {% load static %}
2
3  
4  <a href="{% url 'create-project' %}>Add Project</a>
5  <hr>

```

The sidebar shows a project structure with files like `__init__.py`, `asgi.py`, `settings.py`, `urls.py`, `wsgi.py`, `env`, `projects`, and a `static` folder containing `images` (with `default.jpg` and `logo.svg`), `js`, `styles` (with `main.css`), and `templates` (with `main.html` and `navbar.html`).

The navbar area should have now a logo image on the left top corner of the page:

Project	Positive Votes	Votes	
Ecommerce Website	30%	12	June 15, 2021, 1:24 p.m. Edit Delete View
Portfolio Website	65%	6	June 15, 2021, 1:25 p.m. Edit Delete View
Mumble Social Network	74%	89	June 15, 2021, 1:25 p.m. Edit Delete View
Code Sniper	78%	80	June 15, 2021, 3:53 p.m. Edit Delete View
Yogo Vive	98%	51	June 15, 2021, 3:53 p.m. Edit Delete View

FOOTER

If you right-click the logo and then open the image on a right tab, you'll find the full file path for the logo image. Try it!

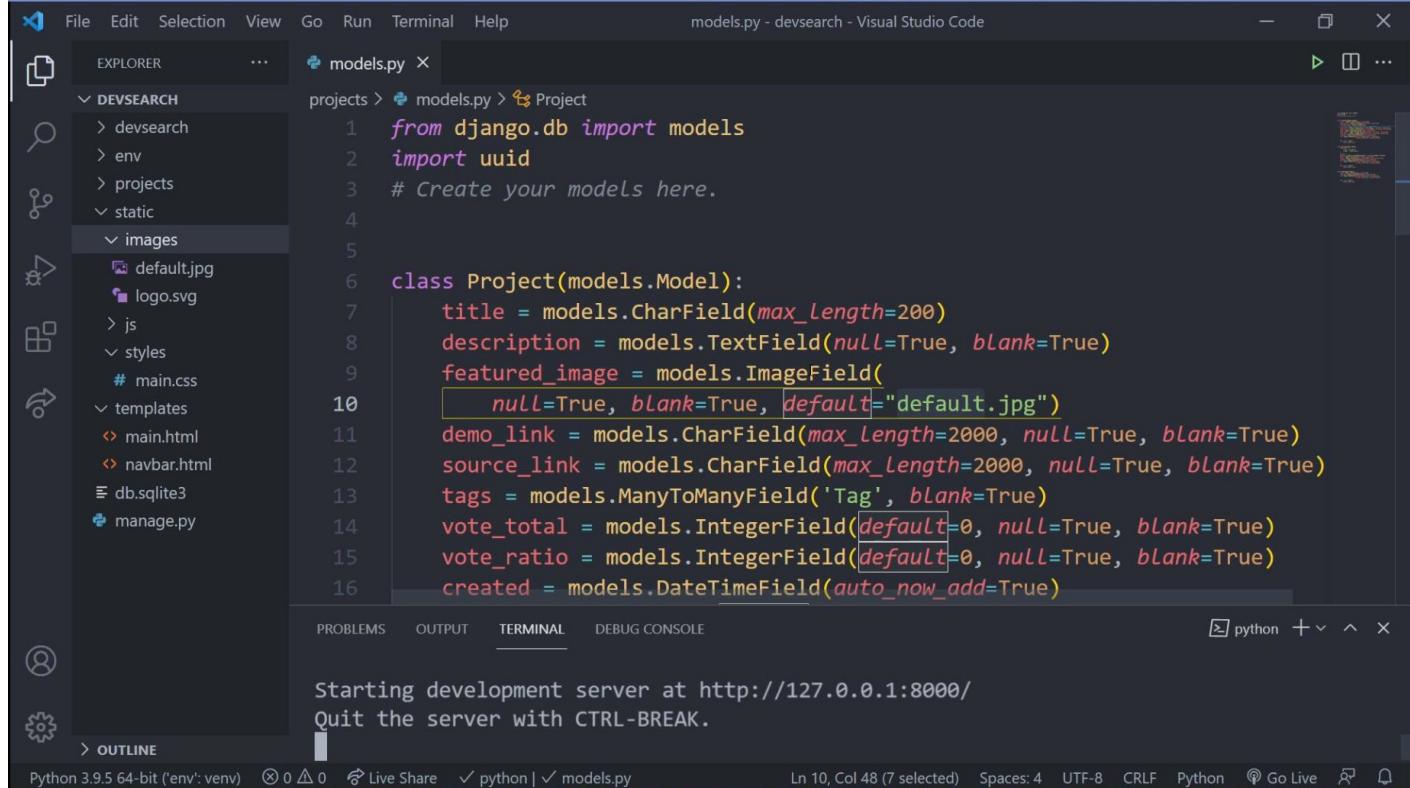
The static approach is enough for details such as logos or backgrounds, but if we want to add an image to each project in the projects page, this approach is inadequate.

There's another way to implement the images.

Adding Images for Each Project on the Projects Page

The way we're going to add individual images for each project is by storing them in a folder using our database to store the full image file path and retrieving each image from that folder.

In order to do this, we'll have to modify our project model in the `/projects/models.py` file.



```

File Edit Selection View Go Run Terminal Help
models.py - devsearch - Visual Studio Code
EXPLORER models.py x
DEVSERCH projects > models.py > Project
> devsearch
> env
> projects
> static
> images
> default.jpg
> logo.svg
> js
> styles
# main.css
> templates
main.html
navbar.html
db.sqlite3
manage.py

from django.db import models
import uuid
# Create your models here.

class Project(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField(null=True, blank=True)
    featured_image = models.ImageField(
        null=True, blank=True, default="default.jpg")
    demo_link = models.CharField(max_length=2000, null=True, blank=True)
    source_link = models.CharField(max_length=2000, null=True, blank=True)
    tags = models.ManyToManyField('Tag', blank=True)
    vote_total = models.IntegerField(default=0, null=True, blank=True)
    vote_ratio = models.IntegerField(default=0, null=True, blank=True)
    created = models.DateTimeField(auto_now_add=True)

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

Python 3.9.5 64-bit ('env': venv) ⚡ 0 △ 0 🔍 Live Share ✓ python | ✓ models.py
Ln 10, Col 48 (7 selected) Spaces: 4 UTF-8 CRLF Python ⚡ Go Live ⚡

```

We included the code into the `Project(models.Model)` class:

```
featured_image = models.ImageField(null=True, blank=True,
default="default.jpg")
```

The `models.ImageField` is set so it can be `null` (we don't have to define a specific file path), and `blank` (shows a blank image or default image if defined), and if that's the case, then use a default image instead `default="default.jpg"` which is our copied `default.jpg` image.

IMPORTANT NOTE:

If you didn't include the Pillow module during your environment set up you'll get an error (`fields.E210`):

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "DEVSERACH" with files like "models.py", "default.jpg", "logo.svg", "js", "styles", "main.css", "templates", "main.html", "navbar.html", "db.sqlite3", and "manage.py".
- Code Editor:** The file "models.py" is open, containing Python code for a "Project" model. The problematic line is:


```
1  from django.db import models
2  import uuid
3  # Create your models here.
4
5
6  class Project(models.Model):
7      title = models.CharField(max_length=200)
8      description = models.TextField(null=True, blank=True)
9      featured_image = models.ImageField(
10          null=True, blank=True, default="default.jpg")
```
- Terminal:** Shows a SystemCheckError message:


```
raise SystemCheckError(msg)
django.core.management.base.SystemCheckError: SystemCheckError: System check identified some issues:
```
- Problems:** A list of errors, starting with:


```
projects.Project.featured_image: (fields.E210) Cannot use ImageField because Pillow is not installed.
```
- Bottom Status Bar:** Shows "Python 3.9.5 64-bit ('env': venv)" and "Ln 10, Col 48 (7 selected)".

To correct the error stop your django server and within your virtual environment run:

```
python -m pip install pillow
```

or if you used pipenv:

```
pipenv install pillow
```

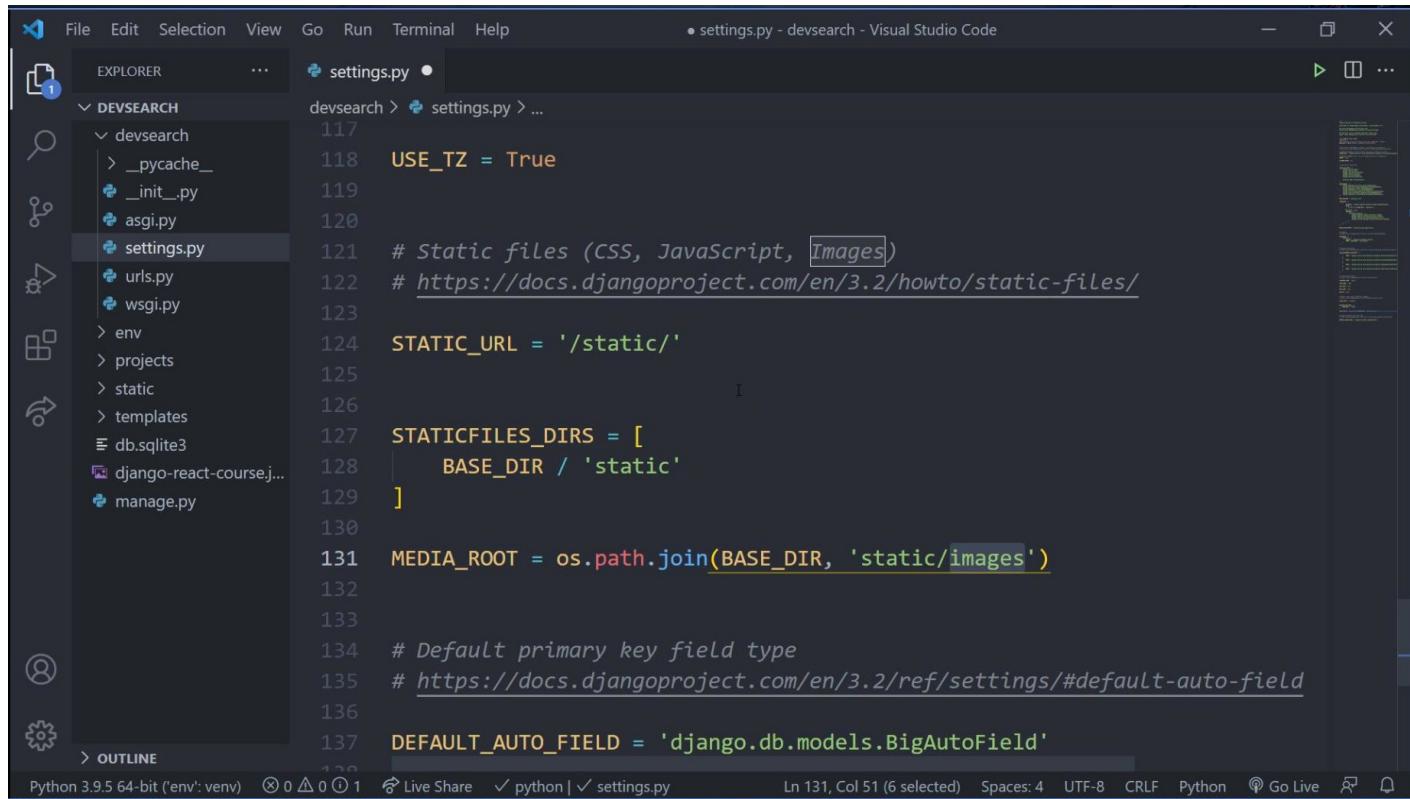
and check if the error disappeared when running the server again:

```
python manage.py runserver
```

Do not forget to Migrate the Model Again! Run the commands:

```
python manage.py makemigrations
python manage.py migrate
```

Next, we need to indicate django in which folder to place the images. We do that by adding some code into the **/devsite/settings.py**



```

File Edit Selection View Go Run Terminal Help
• settings.py - devsearch - Visual Studio Code
EXPLORER ... settings.py
DEVSEARCH
devsearch > settings.py > ...
117
118     USE_TZ = True
119
120
121     # Static files (CSS, JavaScript, Images)
122     # https://docs.djangoproject.com/en/3.2/howto/static-files/
123
124     STATIC_URL = '/static/'
125
126
127     STATICFILES_DIRS = [
128         BASE_DIR / 'static'
129     ]
130
131     MEDIA_ROOT = os.path.join(BASE_DIR, 'static/images')
132
133
134     # Default primary key field type
135     # https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
136
137     DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

Python 3.9.5 64-bit ('env': venv) ⚡ 0 ▲ 0 ⓘ 1 🔍 Live Share ✘ python | ✘ settings.py In 131, Col 51 (6 selected) Spaces: 4 UTF-8 CRLF Python ⓘ Go Live ⌂

and we added the code line:

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'static/images')
```

The **MEDIA_ROOT** constant is used by Django to identify any folder containing any **user generated content**. If you wish to place your project images in some other folder, then just create the folder and change the **MEDIA_ROOT** accordingly.

In this project we'll continue to use the **/static/images** folder just to simplify things.

Testing Where the Project Image is Placed

Open up the admin-panel. Open a project and then select an image to be uploaded. Use the resources folder from the git repository:

The screenshot shows a split interface. On the left is VS Code with the file `settings.py` open. The code defines static files and media roots. On the right is a browser window showing the URL `127.0.0.1:8000/admin/projects/pro...`. A green notification bar at the top says "The project 'Ecommerce Website' was changed successfully." Below it, a sidebar lists five projects: PROJECT, **Ecommerce Website**, Portfolio Website, Mumble Social Network, and Code Sniper.

```

117
118 USE_TZ = True
119
120
121 # Static files (CSS, JavaScript, Images)
122 # https://docs.djangoproject.com/en/2.2/ref/contrib/staticfiles/
123
124 STATIC_URL = '/static/'
125
126
127 STATICFILES_DIRS = [
128     BASE_DIR / 'static'
129 ]
130
131 MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
132
133
134 # Default primary key field type
135 # https://docs.djangoproject.com/en/2.2/ref/settings/#std:setting-DEFAULT_AUTO_FIELD
136
137 DEFAULT_AUTO_FIELD = 'django.db.models.AutoField'

```

go to Featured Picture:

The screenshot shows a split interface. On the left is VS Code with the file `settings.py` open. On the right is a browser window showing the URL `127.0.0.1:8000/admin/projects/pro...`. The page displays a rich text editor for a project's details. It includes fields for "Featured image" (set to `django-react-course.jpg`), "Demo link", "Source link", "Tags" (React, Django, Python, JavaScript), and "Vote total" (12). The "Tags" field has a note: "Hold down 'Control', or 'Command' on a Mac, to select more than one."

and select an image from the resources folder (`django-react-course.jpg`) and save it. You should see that the image was saved in **`static/images`** or the folder you had set up in `/devsite/settings.py` with the **`MEDIA_ROOT`** constant as shown here:

The screenshot shows two windows side-by-side. On the left is Visual Studio Code with the file `settings.py` open. The code defines static files and media root paths. On the right is a browser window showing the Django administration site, specifically the 'Projects' section. A success message at the top says 'The project "Ecommerce Website" was changed successfully.'

```

117
118     USE_TZ = True
119
120
121     # Static files (CSS, JavaScript, Images)
122     # https://docs.djangoproject.com/en/2.0/howto/static-files/
123
124     STATIC_URL = '/static/'
125
126
127     STATICFILES_DIRS = [
128         BASE_DIR / 'static'
129     ]
130
131     MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
132
133
134     # Default primary key field type
135     # https://docs.djangoproject.com/en/2.0/ref/settings/#default-auto-field
136
137     DEFAULT_AUTO_FIELD = 'django.db.models.AutoField'

```

Rendering Individual Project Images

In this section we'll have to modify the `single-project.html` template to showcase images of the referred project. The whole process is a little bit tricky, so try to follow the guide as close as possible.

Let's begin. Open up the `/projects/templates/projects/single-project.html` file and modify the file as shown:

The screenshot shows Visual Studio Code with the file `single-project.html` open. The template uses Jinja2 syntax to render a project's title, description, and featured image. The image URL is dynamically generated from the `project.featured_image.url` variable.

```

1  {% extends 'main.html' %}
2
3  {% block content %}
4
5      
6
7      <h1>{{project.title}}</h1>
8      <hr>
9      {% for tag in project.tags.all %}
10         <span style="border:1px solid black; padding: 2px; margin-right: 10px;">{{tag}}</span>
11     {% endfor %}
12     <hr>
13     <br>
14     <p>{{project.description}}</p>
15
16     {% endblock %}

```

The `style` parameter limits the image size to a width of 500px (pixels)

The added line of code

```
&lt;-----&gt; &lt;-----&gt; &lt;-----&gt;</pre>         |           |      |
| <code>https://www.example.com/media/images/orange.jpg</code> |           |      |

So, another step that has to be taken is to set up the `MEDIA_URL` in the `/devsite/settings.py` file. The `MEDIA_URL` just points to the middle part of the directory. It must end in a slash if set to a non-empty value. You will need to configure these files to be served in both development and production environments. If `MEDIA_URL` is set to an empty string, will in some cases cause errors or bugs.

Check this post in [stackoverflow](#) for further explanation.

```

settings.py X
devsearch > settings.py > ...
134 # Static files (CSS, JavaScript, Images)
135 # https://docs.djangoproject.com/en/3.2/howto/static-files/
136
137 STATIC_URL = '/static/'
138 MEDIA_URL = '/images/'
139
140 STATICFILES_DIRS = [
141 BASE_DIR / 'static'
142]
143
144 MEDIA_ROOT = os.path.join(BASE_DIR, 'static/images')
145 STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
146
147 # Default primary key field type
148 # https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
149
150 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
151

```

Python 3.9.5 64-bit ('env': venv) ⚡ 0 △ 0 ⏪ Live Share ✓ python | ✓ settings.py Ln 138, Col 23 Spaces: 4 UTF-8 CRLF Python ⚡ Go Live ⌂ ⌂

## Alternative Ways to Show the Image

### Modify the Django Project urls.py File

Next, we need to do some modifications to the `/devsite/urls.py` file in order to find the individual project images.

```

File Edit Selection View Go Run Terminal Help • urls.py - devsearch - Visual Studio Code
EXPLORER ... > single-project.html settings.py urls.py
DEVSEARCH
 devsearch > urls.py > ...
 > __pycache__
 > __init__.py
 > asgi.py
 > settings.py
 urls.py
 wsgi.py
 > env
 > projects
 > static
 > templates
 db.sqlite3
 manage.py
urlpatterns = [
 path('admin/', admin.site.urls),
 path('', include('projects.urls'))
]
urlpatterns += static(settings.MEDIA_URL,
 document_root=settings.MEDIA_ROOT)

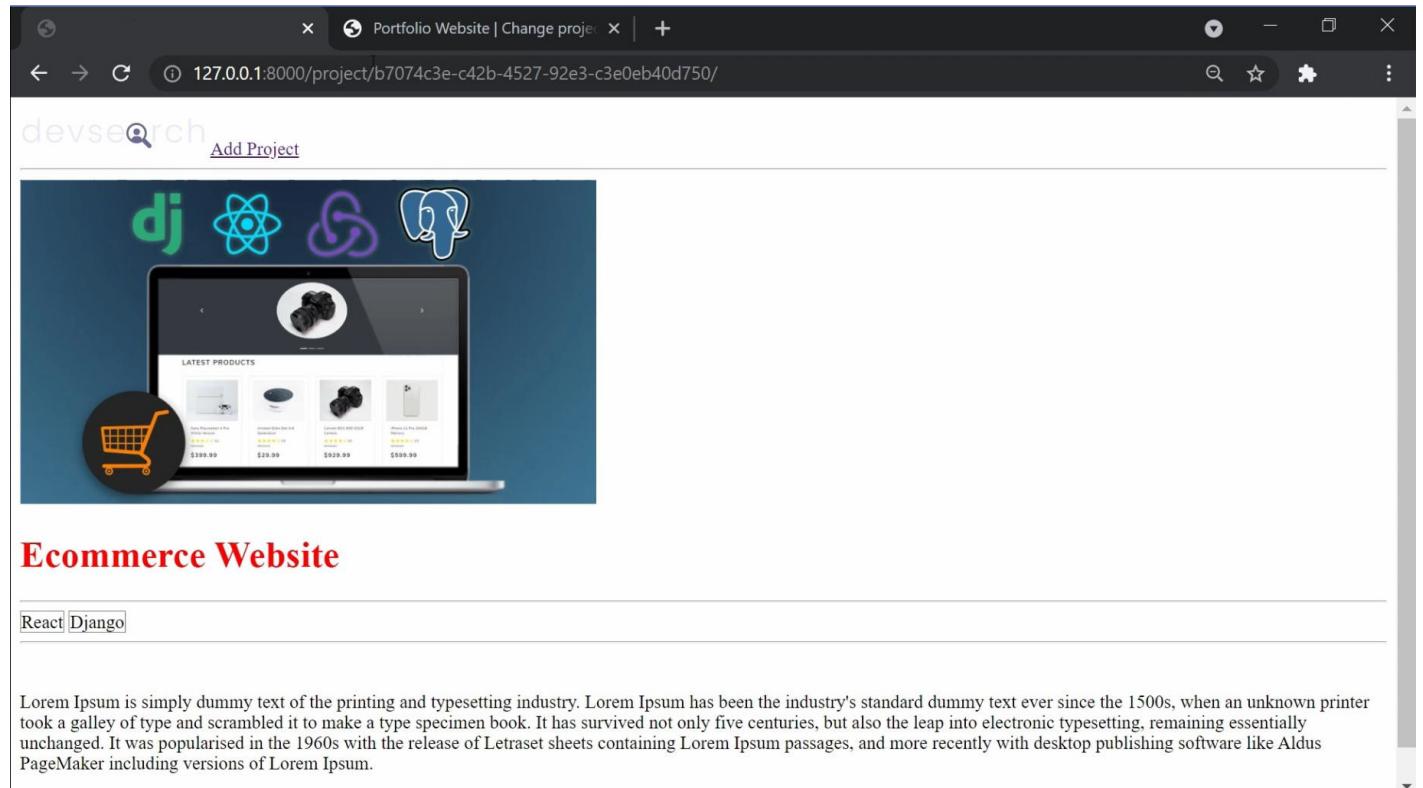
```

Python 3.9.5 64-bit ('env': venv) ⚡ 0 △ 0 ⏪ Live Share ✓ python | ✓ urls.py Ln 10, Col 1 Spaces: 4 UTF-8 CRLF Python ⚡ Go Live ⌂ ⌂

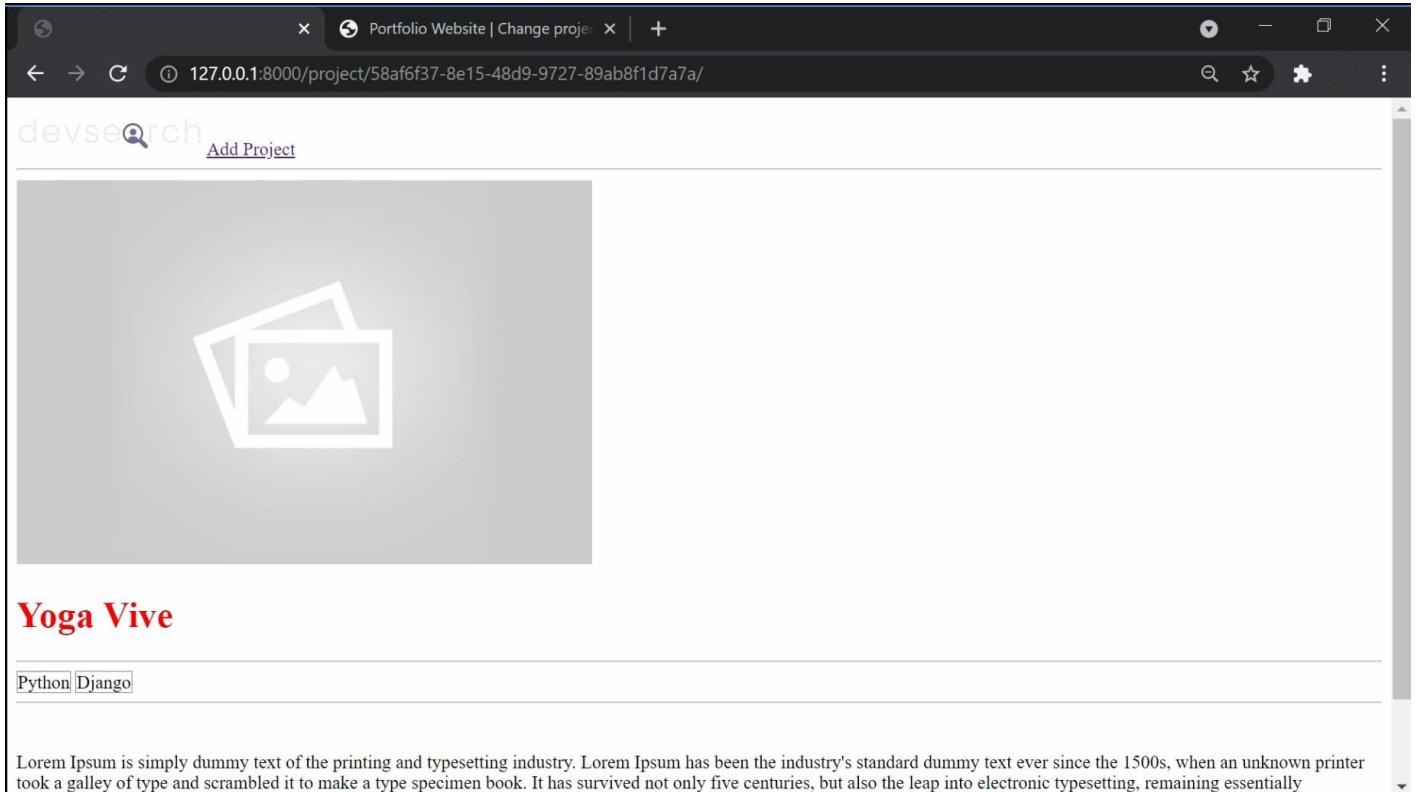
What the changes are doing is finding the constants "**MEDIA\_URL**" and "**MEDIA\_ROOT**" in order to associate the individual image file path in the **/static/images** folder and convert those file paths into a url and making the files visible through the associated url file path.

The conversion from file path to url is done by the static function imported from **django.conf.urls.static**

The end result should look something like this:



and one with the default image:

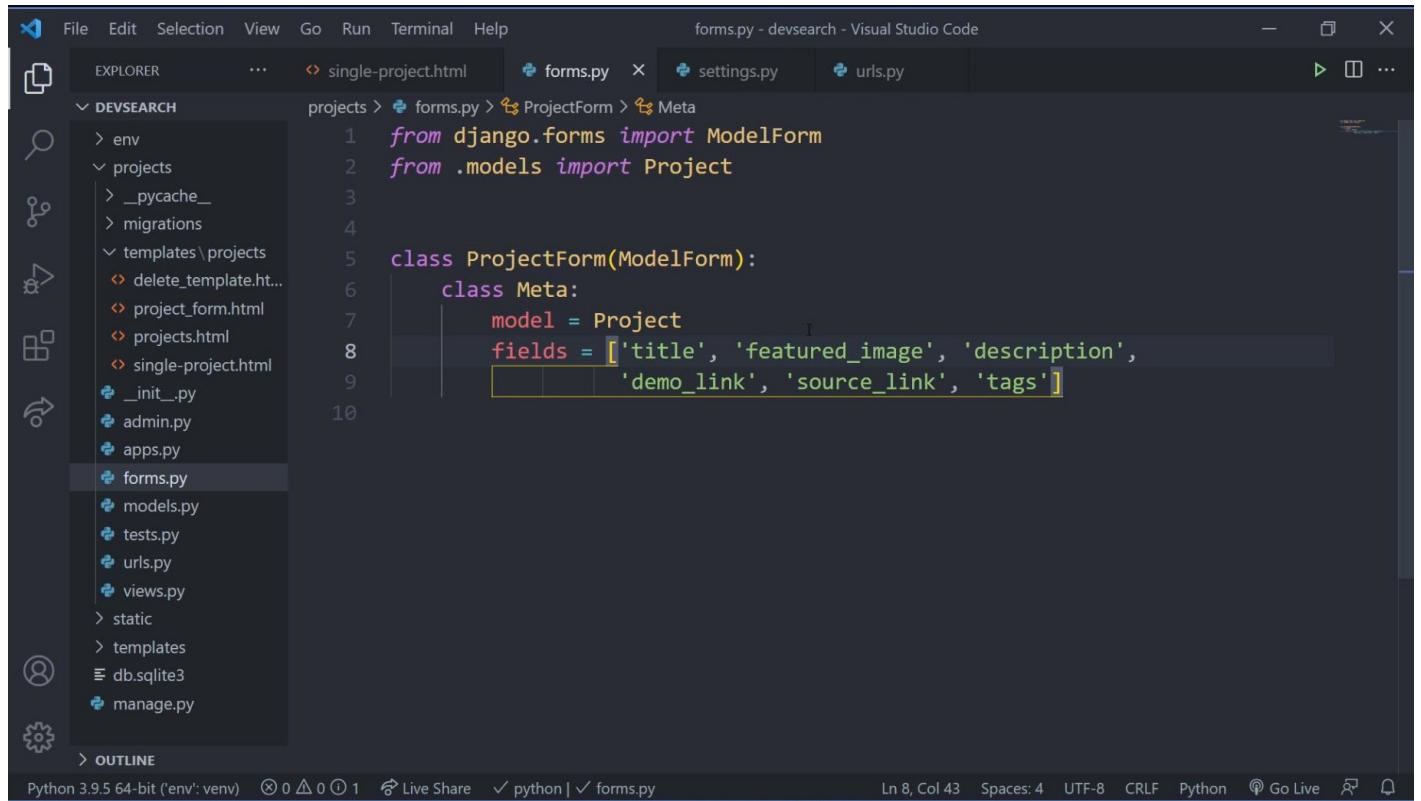


The screenshot shows a web browser window with the URL `127.0.0.1:8000/project/58af6f37-8e15-48d9-9727-89ab8f1d7a7a/`. The page has a header with the text "devse<sup>arch</sup>" and a search icon, followed by a link "Add Project". Below the header is a large gray placeholder image area containing a white icon of two overlapping photographs. The main title of the project is "Yoga Vive", displayed in red text. Underneath the title are two small tags: "Python" and "Django". A descriptive paragraph follows, starting with "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially".

Please remember that we uploaded the image via the administration panel at `127.0.0.1/admin/` with modifications directly to the database. However, we should let any user to upload what ever image desires for her/his project. We will modify our forms in order to allow this type of CRUD operation.

## Modifying the Project Form to Allow Custom Images

First, we need to modify the `/projects/forms.py` file to add the new field we added to upload the image `featured_image` in the nested **class Meta**:



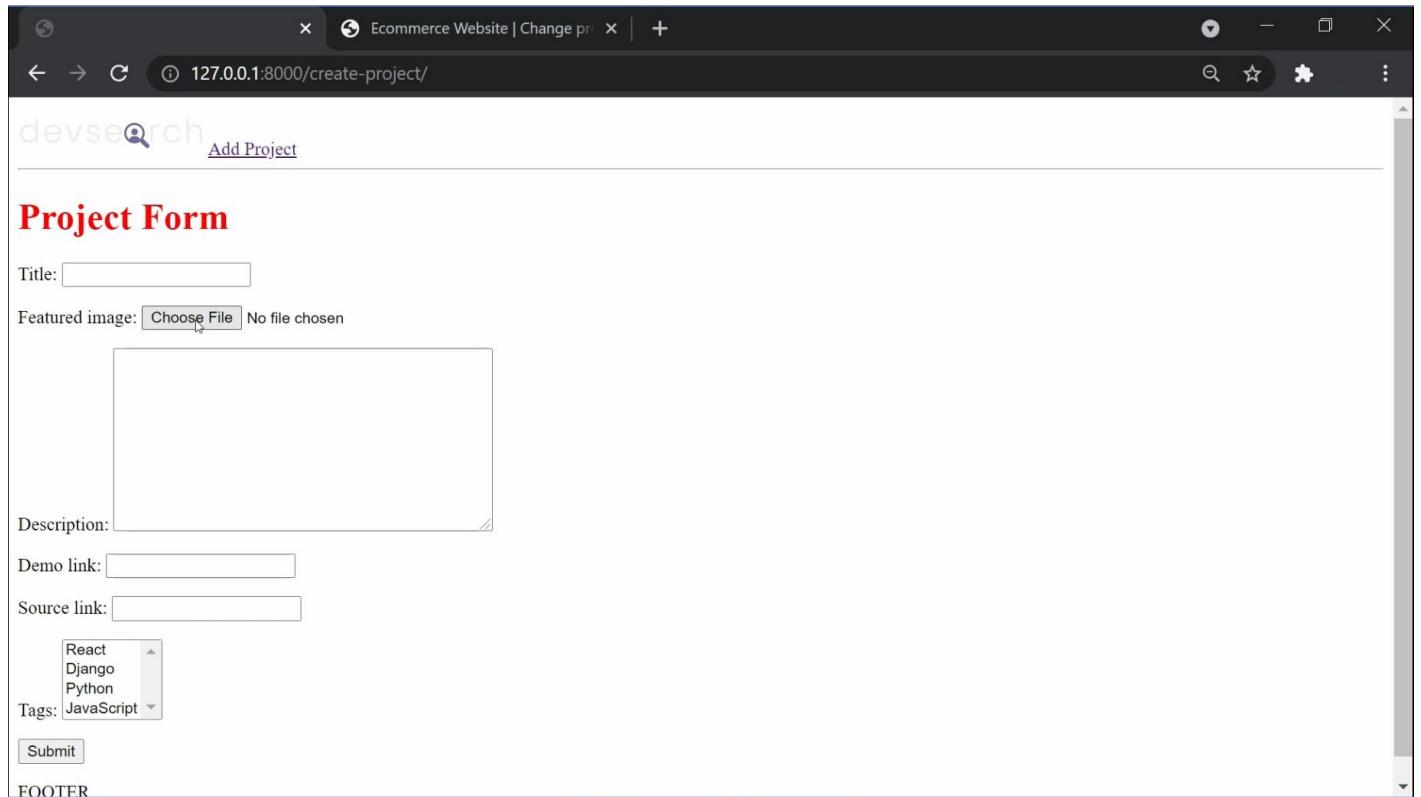
```

File Edit Selection View Go Run Terminal Help
forms.py - devsearch - Visual Studio Code
EXPLORER ... forms.py x settings.py urls.py
DEVSEARCH projects > forms.py > ProjectForm > Meta
1 from django.forms import ModelForm
2 from .models import Project
3
4
5 class ProjectForm(ModelForm):
6 class Meta:
7 model = Project
8 fields = ['title', 'featured_image', 'description',
9 'demo_link', 'source_link', 'tags']
10

```

Python 3.9.5 64-bit ('env': venv) ⚡ 0 △ 0 ⓘ 1 🔍 Live Share ✓ python | ✓ forms.py Ln 8, Col 43 Spaces: 4 UTF-8 CRLF Python ⓘ Go Live ⌂

So our form now has the new field included:



Ecommerce Website | Change pr... +

devse**rch** [Add Project](#)

## Project Form

Title:

Featured image:  Choose File No file chosen

Description:

Demo link:

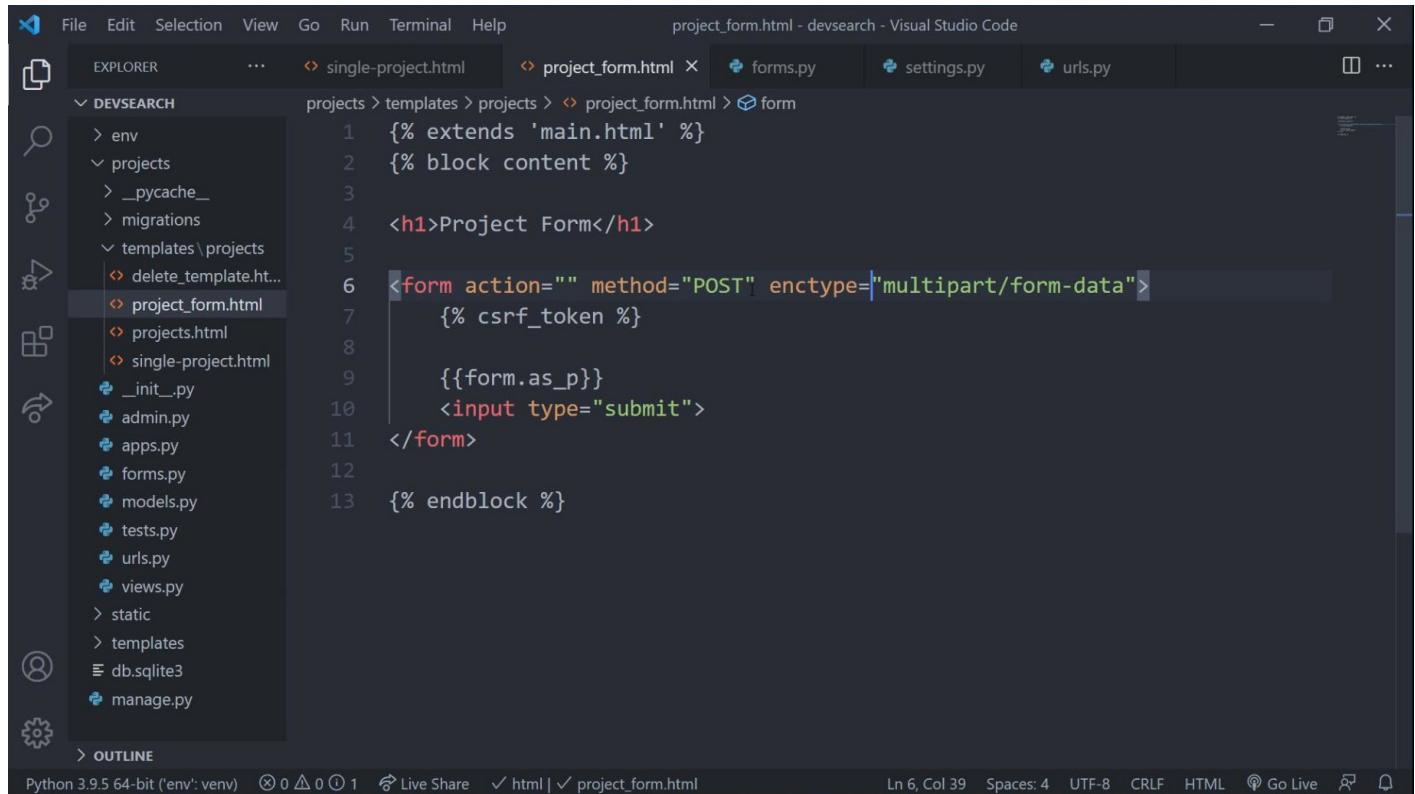
Source link:

Tags:

FOOTER

Still, the form won't process the image selection process just yet. You can actually go and select an image, but it won't place it in the `/static/image` folder.

We need to modify the `/projects/templates/projects/project_form.html` template:



The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** project\_form.html - devsearch - Visual Studio Code.
- Explorer Bar (Left):**
  - DEVSEARCH section:
    - > env
    - < projects
      - > \_\_pycache\_\_
      - > migrations
    - < templates\projects
      - < delete\_template.html
      - < project\_form.html
      - < projects.html
      - < single-project.html
    - < \_\_init\_\_.py
    - < admin.py
    - < apps.py
    - < forms.py
    - < models.py
    - < tests.py
    - < urls.py
    - < views.py
  - > static
  - > templates
  - db.sqlite3
  - manage.py
- Code Editor (Center):**

```
1 {% extends 'main.html' %}\n2 {% block content %}\n3\n4 <h1>Project Form</h1>\n5\n6 <form action="" method="POST" enctype="multipart/form-data">\n7 {% csrf_token %}\n8\n9 {{form.as_p}}\n10 <input type="submit">\n11 </form>\n12\n13 {% endblock %}
```
- Bottom Status Bar:** Python 3.9.5 64-bit ('env': venv) | Live Share | ✓ html | ✓ project\_form.html | Ln 6, Col 39 | Spaces: 4 | UTF-8 | CRLF | HTML | Go Live | ⚡ | 🔍

the line for form has changed to:

```
<form action="" method="POST" enctype="multipart/form-data">
```

The **multipart/form-data** instructs the form to break up the data into specific fields that we can recover later on the **views.py** file.

Next, in the **/projects/views.py**, we need to modify our `createProject` and `updateProject` classes:

```

File Edit Selection View Go Run Terminal Help • views.py - devsearch - Visual Studio Code
EXPLORER ... single-project.html project_form.html views.py forms.py settings.py urls.py ...
DESEARCH projects > views.py > createProject
> env
< projects
> __pycache__
> migrations
< templates\projects
> delete_template.htm...
> project_form.html
> projects.html
> single-project.html
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
> static
> templates
db.sqlite3
manage.py
OUTLINE
12
13 def project(request, pk):
14 projectObj = Project.objects.get(id=pk)
15 return render(request, 'projects/single-project.html', {'project': pro...
16
17
18 def createProject(request):
19 form = ProjectForm()
20
21 if request.method == 'POST':
22 form = ProjectForm(request.POST, request.FILES)
23 if form.is_valid():
24 form.save()
25 return redirect('projects')
26
27 context = {'form': form}
28 return render(request, "projects/project_form.html", context)
29
30
31 def updateProject(request, pk):
32 project = Project.objects.get(id=pk)
33
34
35 if request.method == 'POST':
36 form = ProjectForm(request.POST, request.FILES, instance=project)
37 if form.is_valid():
38 form.save()
39 return redirect('projects')
40
41 context = {'form': form}
42 return render(request, "projects/project_form.html", context)
43
44
45 def deleteProject(request, pk):
46 project = Project.objects.get(id=pk)
47 if request.method == 'POST':
48 project.delete()
49 return redirect('projects')
50 context = {'object': project}

```

Python 3.9.5 64-bit ('env': venv) ⚡ 0 ⚡ 0 ⚡ 1 🔍 Live Share ✅ python | ✅ views.py Ln 22, Col 55 Spaces: 4 UTF-8 CRLF Python ⚡ Go Live ⚡ 🔍

and see that we added a new argument to the createProject form when a "POST" request is made:

```
form = ProjectForm(request.POST, request.FILES)
```

```

File Edit Selection View Go Run Terminal Help • views.py - devsearch - Visual Studio Code
EXPLORER ... single-project.html project_form.html views.py forms.py settings.py urls.py ...
DESEARCH projects > views.py > updateProject
> env
< projects
> __pycache__
> migrations
< templates\projects
> delete_template.htm...
> project_form.html
> projects.html
> single-project.html
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
> static
> templates
db.sqlite3
manage.py
OUTLINE
30
31 def updateProject(request, pk):
32 project = Project.objects.get(id=pk)
33 form = ProjectForm(instance=project)
34
35 if request.method == 'POST':
36 form = ProjectForm(request.POST, request.FILES, instance=project)
37 if form.is_valid():
38 form.save()
39 return redirect('projects')
40
41 context = {'form': form}
42 return render(request, "projects/project_form.html", context)
43
44
45 def deleteProject(request, pk):
46 project = Project.objects.get(id=pk)
47 if request.method == 'POST':
48 project.delete()
49 return redirect('projects')
50 context = {'object': project}

```

Python 3.9.5 64-bit ('env': venv) ⚡ 0 ⚡ 0 ⚡ 1 🔍 Live Share ✅ python | ✅ views.py Ln 34, Col 1 Spaces: 4 UTF-8 CRLF Python ⚡ Go Live ⚡ 🔍

and the modification is:

```
form = ProjectForm(request.POST, request.FILES, instance=project)
```

And that should be all the necessary steps to include images in each new projects and update images to existing projects.

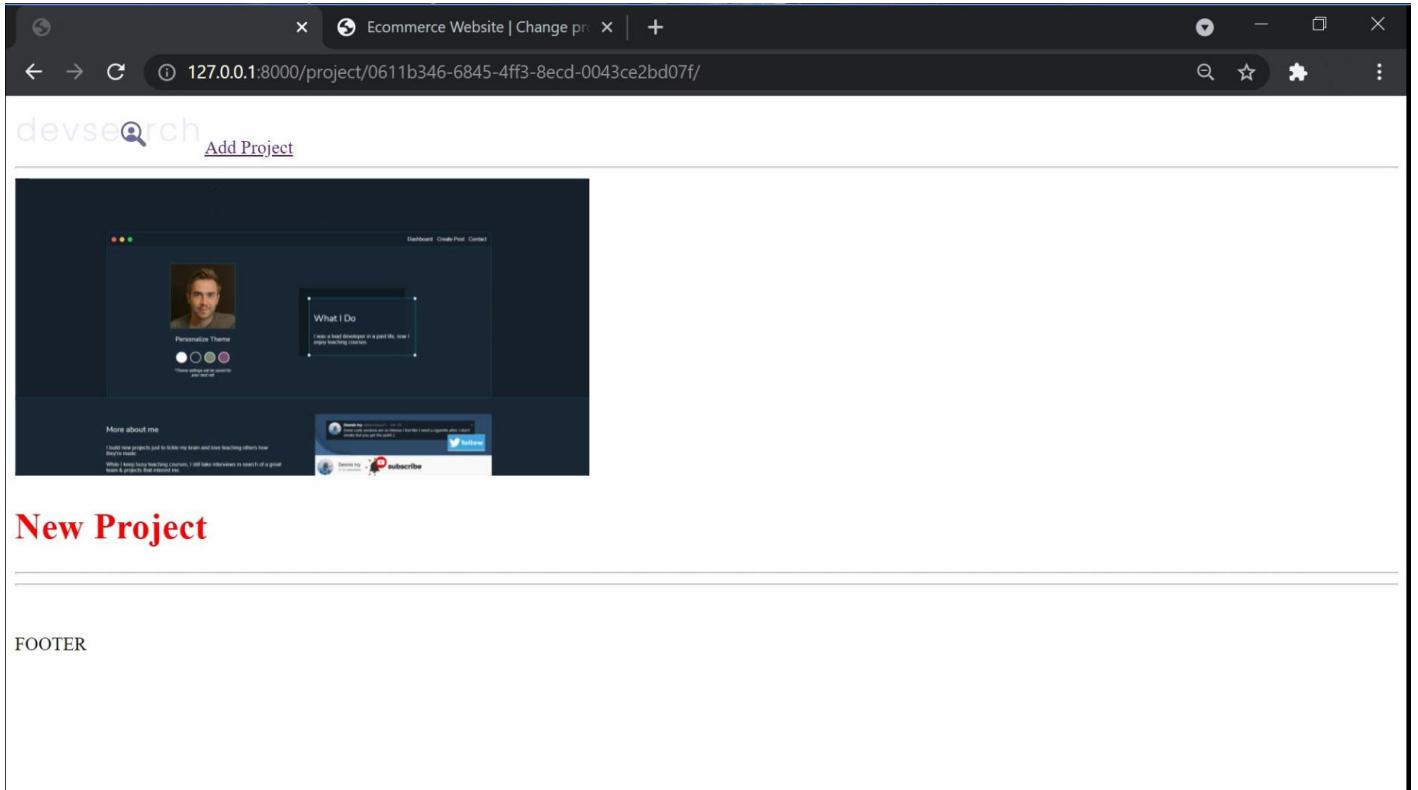
Go ahead and test the form by creating a new project:

The screenshot shows a web browser window with the URL `127.0.0.1:8000/create-project/`. The page title is "Project Form". The form fields are as follows:

- Title:
- Featured image:  Choose File No file chosen
- Description:
- Demo link:
- Source link:
- Tags:  React  Django  Python  JavaScript
- Buttons:

At the bottom of the page, there is a footer section labeled "FOOTER".

and submit a new image in the form:



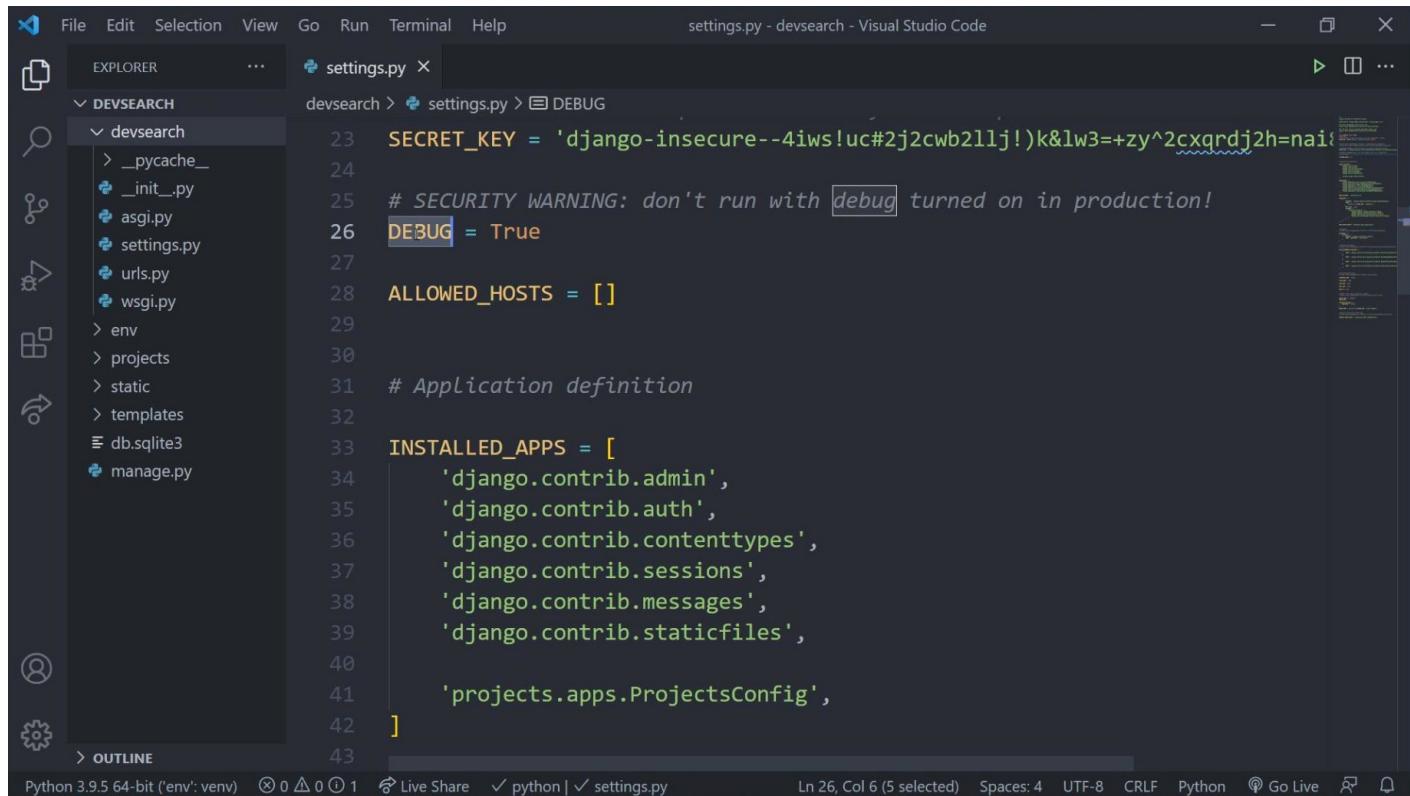
And finally, try to update the images for existing projects in your projects page. **Go ahead and test it.**

This should be all about setting images on individual forms. We accomplished the following:

1. Set up an image or icon on the navbar using static files.
2. Set up dynamic images for individual project and their pages.
3. Modified the forms in order to enter images and store them in specific folders. You can set up any folder to do this, it not necessarily done using the `/static/images` folder. You could use a folder inside the application folder, something such as `/projects/images` and you would have to modify the **MEDIA\_ROOT** and **MEDIA\_URL** constants in the `/devsite/settings.py` file.

## About Static Files in Production

All what we've done up to here has been done with the DEBUG constant (flag) set to True. That is, we've been working on debug mode. In the `/devsite/settings.py` file:



The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** settings.py - devsearch - Visual Studio Code.
- Explorer Bar (Left):** Shows the project structure under "DESEARCH": devsearch (containing \_\_pycache\_\_, \_\_init\_\_.py, asgi.py, settings.py, urls.py, wsgi.py), env, projects, static, templates, db.sqlite3, and manage.py.
- Code Editor (Center):** The settings.py file is open. The DEBUG variable is highlighted in blue. The code is as follows:

```
SECRET_KEY = 'django-insecure--4iws!uc#2j2cwb2llj!k&lw3=+zy^2cxqrjdj2h=nai'
SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
Application definition
INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'projects.apps.ProjectsConfig',
]
```

- Bottom Status Bar:** Python 3.9.5 64-bit ('env': venv) | Live Share | python | settings.py | Ln 26, Col 6 (5 selected) | Spaces: 4 | UTF-8 | CRLF | Python | Go Live | Bell icon.

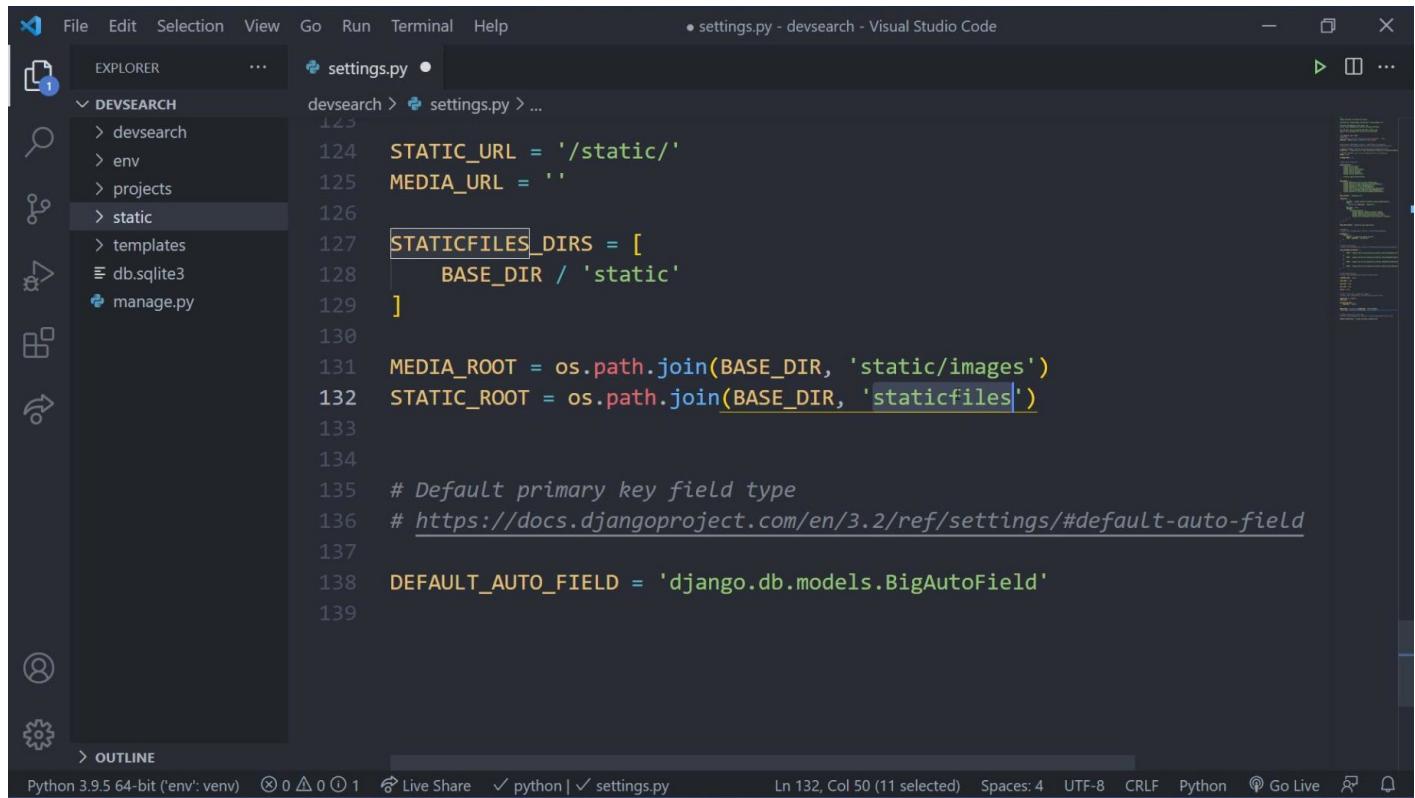
However, once we migrate our site onto Production mode (DEBUG = False), the /static/ folder **WON'T BE AVAILABLE**, i. e., all the styling files (CSS, JavaScript, JAVA, etc) won't be accessible for the Production site.

## How Do We Run The Production Site Then?

Very much like when we migrate our databases, we have to migrate all our static files into a new folder, basically, a copy of the contents of the static folder we've been using during development. We could be tempted to copy it manually, and we could, but Django can do that for us instead.

### Procedure

1. Set up a constant flag named **STATIC\_ROOT** in the **/devsite/setting.py** file:



```
File Edit Selection View Go Run Terminal Help • settings.py - devsearch - Visual Studio Code
EXPLORER ...
DESEARCH devsearch > settings.py > ...
123
124 STATIC_URL = '/static/'
125 MEDIA_URL = ''
126
127 STATICFILES_DIRS = [
128 BASE_DIR / 'static'
129]
130
131 MEDIA_ROOT = os.path.join(BASE_DIR, 'static/images')
132 STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
133
134
135 # Default primary key field type
136 # https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
137
138 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
139
OUTLINE
Python 3.9.5 64-bit ('env': venv) ⚡ 0 △ 0 ⌂ 1 🔍 Live Share ✅ python | ✅ settings.py Ln 132, Col 50 (11 selected) Spaces: 4 UTF-8 CRLF Python ⚡ Go Live 🔍
```

which is pretty similar to **MEDIA\_ROOT**. Here we chose '**staticfiles**' as the name of the static folder for the site in production mode. It can be any other given name than '**staticfiles**', it doesn't really matter what name we assign to it, but preferably it has to be something related to it.

2. Open the terminal window in your IDE and run the command:

```
python manage.py collectstatic
```

And as you can see, it's a very similar procedure to migrate your model modifications to the database tables.

File Edit Selection View Go Run Terminal Help

settings.py - devsearch - Visual Studio Code

EXPLORER

DEVSEA... 🔍 ⏪ ⏴ ⏵ ⏷

> devsearch  
> env  
> projects  
> static  
staticfiles  
> admin  
images  
> styles  
> templates  
db.sqlite3  
manage.py

settings.py X

devsearch > settings.py > ...

```
124 STATIC_URL = '/static/'
125 MEDIA_URL = ''
126
127 STATICFILES_DIRS = [
128 BASE_DIR / 'static'
129]
130
131 MEDIA_ROOT = os.path.join(BASE_DIR, 'static/images')
132 STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
133
134
```

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE

(env) python manage.py collectstatic

```
137 static files copied to 'C:\Users\Ivy\Desktop\devsearch\staticfiles'.
```

+ ^ x

python powershell

You can see that the staticfiles folder is created automatically and all the contents from the static folder are bundled up together and copied to the new folder. There's also a new addition with the admin folder. It contains all the styling files for the admin panel.

If you have to do modifications to the contents of the static folder, then, like migrating any modification to the database, you'll have to migrate the contents of the static folder to the staticfiles folder for the Production site.

## What's Running The Site in Production Mode?

The debug mode provide us with sensitive information about possible errors during development such as if we type a route that doesn't exist, then the site running Django will show us the documented errors produced while trying to find the non existing page.

Production mode (`DEBUG = False`) cancels all warning and error messages.

To run in Production mode we have to add a list of ***ALLOWED\_HOSTS*** to the `setting.py` file:

```

settings.py - devsearch - Visual Studio Code
File Edit Selection View Go Run Terminal Help
settings.py > ...
SECRET_KEY = 'django-insecure--4iws!uc#2j2cwb2llj!k&lw3=+zy^2cxqrdj2h=nai'
SECURITY WARNING: don't run with debug turned on in production!
DEBUG = False
ALLOWED_HOSTS = ['localhost', '127.0.0.1']
Application definition
INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
]
System check identified no issues (0 silenced).
June 16, 2021 - 14:00:51
Django version 3.2.4, using settings 'devsearch.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

```

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE

python + ^ x

Python 3.9.5 64-bit ('env': venv) Live Share python | settings.py Ln 29, Col 1 Spaces: 4 UTF-8 CRLF Python Go Live

If you try to run the site right now you would find that all your static images and styling can not be seen:

Project	Positive Votes	Votes	Date	Actions
Ecommerce Website	30%	12	June 15, 2021, 1:24 p.m.	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View</a>
Yoga Vive	98%	51	June 16, 2021, 5:14 p.m.	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View</a>
Code Sniper	75%	80	June 16, 2021, 5:14 p.m.	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View</a>
Portfolio Website	65%	6	June 16, 2021, 5:15 p.m.	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View</a>
Mumble Social Network	89%	74	June 16, 2021, 5:15 p.m.	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View</a>

FOOTER

The reason being that we need to include a url path to the **staticfiles** folder and add a middleware package that serves the static files, but not the user uploaded content.

The modifications to the urls:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "DESEARCH". The "urls.py" file is selected.
- Code Editor:** Displays the following Python code in the "urls.py" file:

```
5
6 terns = [
7 th('admin/', admin.site.urls),
8 th('', include('projects.urls'))
9
10
11
12 terns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
13 terns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```
- Terminal:** Shows log entries from the terminal:

```
[16/Jun/2021 14:01:07] "GET /project/b7074c3e-c42b-4527-92e3-c3e0eb40d750/ HTTP/1.1" 200 1255
[16/Jun/2021 14:01:07] "GET /static/styles/main.css HTTP/1.1" 404 179
[16/Jun/2021 14:01:07] "GET /static/images/logo.svg HTTP/1.1" 404 179
[16/Jun/2021 14:01:07] "GET /django-react-course.jpg HTTP/1.1" 404 179
```
- Status Bar:** Shows "Python 3.9.5 64-bit ('env': venv) 0 ▲ 0 ① 1 🔍 Live Share ✘ python 1 ✘ urls.py" and "In 13, Col 73 Spaces: 4 UTF-8 CRLF Python ⚙ Go Live 🔍".

where we added the line:

```
urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)
```

The middle ware is "whitenoise" and we have to install it in our virtual environment.

```
python -m pip install whitenoise
```

and then do some modifications to the MIDDLEWARE list in the settings.py file.

```

File Edit Selection View Go Run Terminal Help
settings.py - devsearch - Visual Studio Code
EXPLORER ... settings.py urls.py
DEVSEARCH devsearch
devsearch > _pycache_
> __init__.py
> asgi.py
> settings.py
> urls.py
> wsgi.py
> env
> projects
> static
> staticfiles
> templates
db.sqlite3
manage.py
INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'projects.apps.ProjectsConfig',
]
MIDDLEWARE = [
 'django.middleware.security.SecurityMiddleware',
 'whitenoise.middleware.WhiteNoiseMiddleware',
 'django.contrib.sessions.middleware.SessionMiddleware',
 'django.middleware.common.CommonMiddleware',
 'django.middleware.csrf.CsrfViewMiddleware',
 'django.contrib.auth.middleware.AuthenticationMiddleware',
 'django.contrib.messages.middleware.MessageMiddleware',
 'django.middleware.clickjacking.XFrameOptionsMiddleware'
]

```

Python 3.9.5 64-bit ('env': venv) ① 0 △ 0 ① 1 Live Share ② python | settings.py Ln 48, Col 5 Spaces: 4 UTF-8 CRLF Python ④ Go Live ⑤ ⑥ ⑦

and we should be able to see:

Project	Positive Votes	Votes	Date	Action
Ecommerce Website	30%	12	June 15, 2021, 1:24 p.m.	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View</a>
Yoga Vive	98%	51	June 16, 2021, 5:14 p.m.	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View</a>
Code Sniper	75%	80	June 16, 2021, 5:14 p.m.	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View</a>
Portfolio Website	65%	6	June 16, 2021, 5:15 p.m.	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View</a>
Mumble Social Network	89%	74	June 16, 2021, 5:15 p.m.	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View</a>

FOOTER

but we won't be able to see the user uploaded content:

The screenshot shows a browser window with the URL `127.0.0.1:8000/project/b7074c3e-c42b-4527-92e3-c3e0eb40d750/`. The page has a header with the text "devse**r**ch" and a magnifying glass icon, followed by a link "Add Project". Below the header is a small green thumbnail image. The main title "Ecommerce Website" is displayed in large red font. Underneath the title, there are two buttons: "React" and "Django". A large text block follows, containing placeholder text (Lorem Ipsum) about the history of dummy text in printing and typesetting.

devse**r**ch Add Project

Ecommerce Website

React Django

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

FOOTER

This is an issue that will be solved later on.