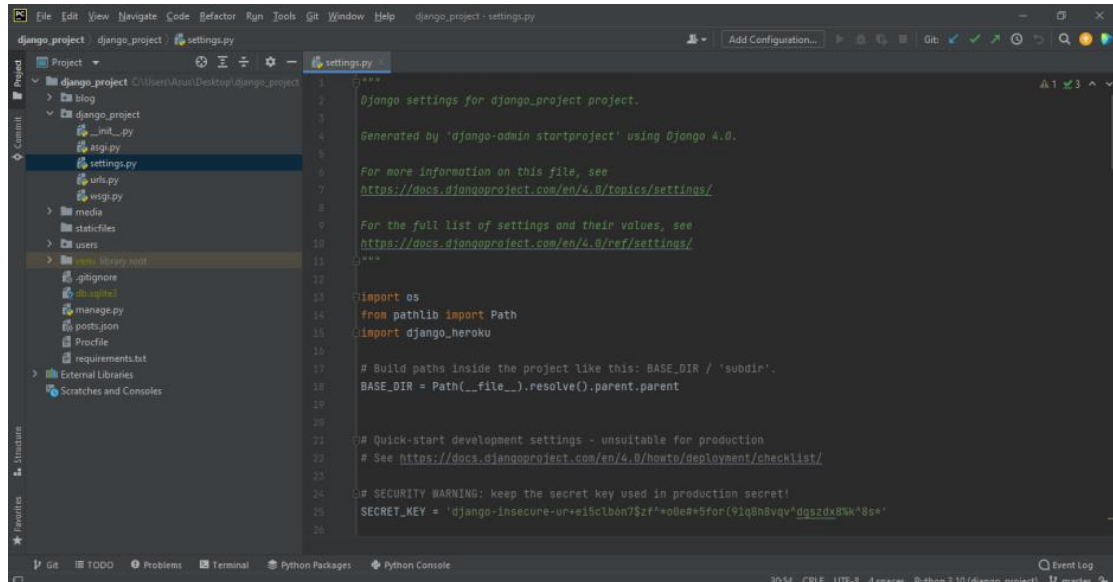


Name: Gerard Anthony Resaba
Year/Course/Section: 3/BSCS/A

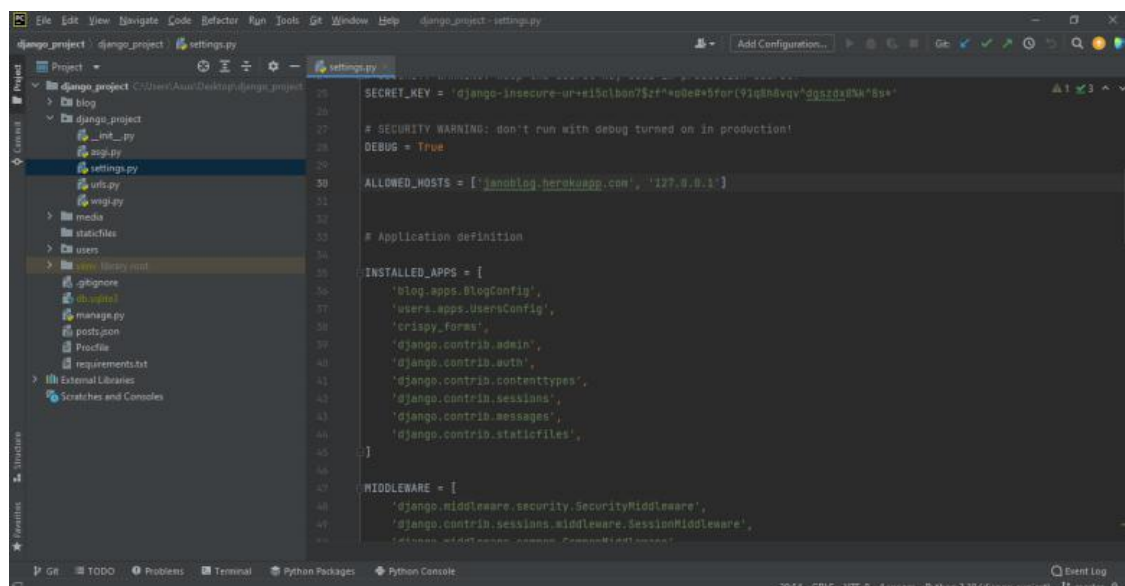
DJANGO ACTIVITY

➤ django_project > django_project > settings.py



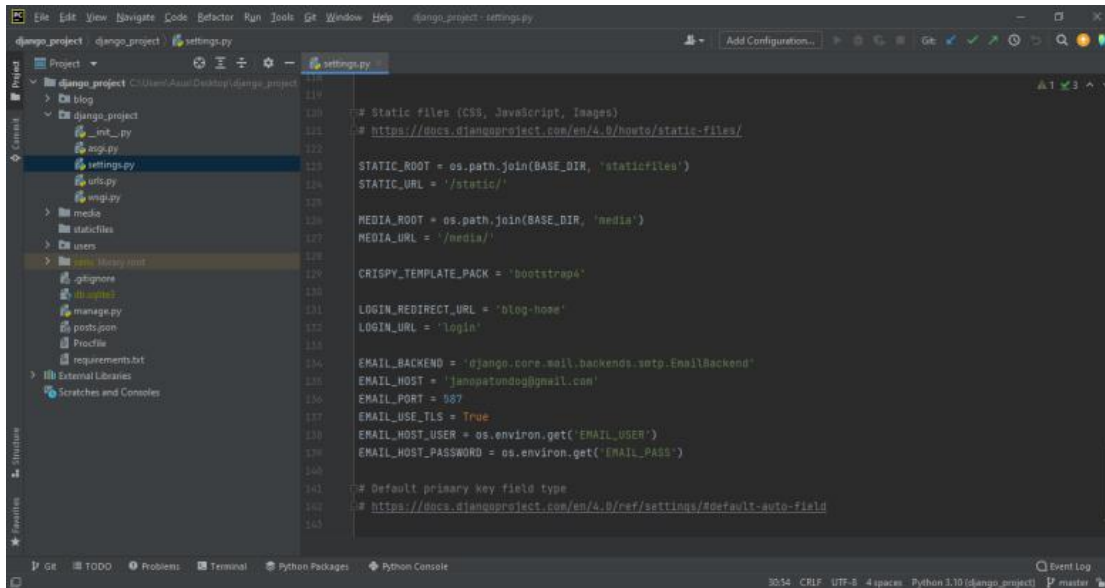
The screenshot shows a code editor with the file explorer on the left displaying the project structure. The main editor window shows the contents of `settings.py`. The file contains comments about the settings, links to Django documentation, and the beginning of the configuration code. The `SECRET_KEY` is set to a placeholder value.

```
1  """  
2  Django settings for django_project project.  
3  
4  Generated by 'django-admin startproject' using Django 4.0.  
5  
6  For more information on this file, see  
7  https://docs.djangoproject.com/en/4.0/topics/settings/  
8  
9  For the full list of settings and their values, see  
10 https://docs.djangoproject.com/en/4.0/ref/settings/  
11 """  
12  
13 import os  
14 from pathlib import Path  
15 import django_heroku  
16  
17 # Build paths inside the project like this: BASE_DIR / 'subdir'.  
18 BASE_DIR = Path(__file__).resolve().parent.parent  
19  
20  
21 # Quick-start development settings - unsuitable for production  
22 # See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/  
23  
24 # SECURITY WARNING: Keep the secret key used in production secret!  
25 SECRET_KEY = 'django-insecure-ur+e15clb6k7$zf*o0e0+5for(91q8h8vqv*0gszdx8xk*8s*'  
26
```

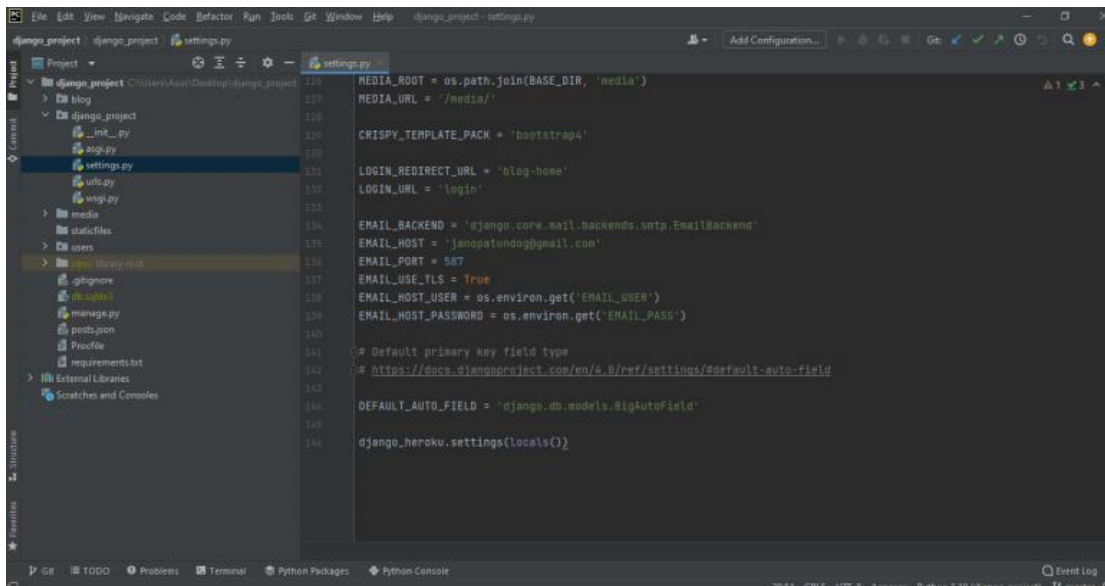


The screenshot shows the same code editor with the `settings.py` file updated. The `SECRET_KEY` is now a longer, more secure string. The `DEBUG` flag is set to `True`. The `ALLOWED_HOSTS` list is updated to include `localhost` and `127.0.0.1`. The `INSTALLED_APPS` list is updated to include the necessary Django and project apps. The `MIDDLEWARE` list is also updated.

```
25 SECRET_KEY = 'django-insecure-ur+e15clb6k7$zf*o0e0+5for(91q8h8vqv*0gszdx8xk*8s*'  
26  
27 # SECURITY WARNING: don't run with debug turned on in production!  
28 DEBUG = True  
29  
30 ALLOWED_HOSTS = ['localhost', '127.0.0.1']  
31  
32  
33 # Application definition  
34  
35 INSTALLED_APPS = [  
36     'blog.apps.BlogConfig',  
37     'users.apps.UsersConfig',  
38     'crispy_forms',  
39     'django.contrib.admin',  
40     'django.contrib.auth',  
41     'django.contrib.contenttypes',  
42     'django.contrib.sessions',  
43     'django.contrib.messages',  
44     'django.contrib.staticfiles',  
45 ]  
46  
47 MIDDLEWARE = [  
48     'django.middleware.security.SecurityMiddleware',  
49     'django.contrib.sessions.middleware.SessionMiddleware',  
50     'django.middleware.common.CommonMiddleware',  
51     'django.middleware.csrf.CsrfViewMiddleware',  
52 ]
```

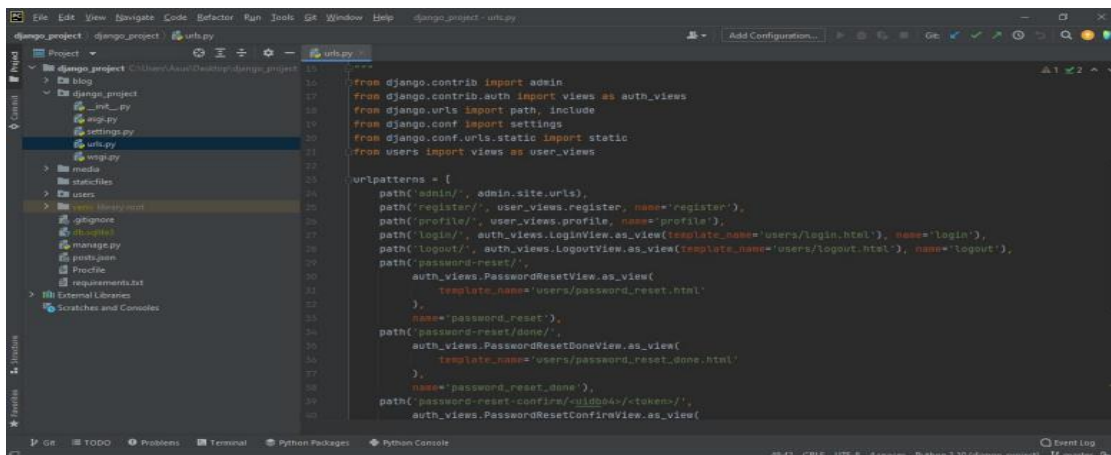


```
119 # Static files (CSS, JavaScript, Images)
120 # https://docs.djangoproject.com/en/4.0/howto/static-files/
121 STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
122 STATIC_URL = '/static/'
123
124 MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
125 MEDIA_URL = '/media/'
126
127 CRISPY_TEMPLATE_PACK = 'bootstrap4'
128
129 LOGIN_REDIRECT_URL = 'blog-home'
130 LOGIN_URL = 'login'
131
132 EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
133 EMAIL_HOST = 'janopetundog@gmail.com'
134 EMAIL_PORT = 587
135 EMAIL_USE_TLS = True
136 EMAIL_HOST_USER = os.environ.get('EMAIL_USER')
137 EMAIL_HOST_PASSWORD = os.environ.get('EMAIL_PASS')
138
139 # Default primary key field type
140 # https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
```

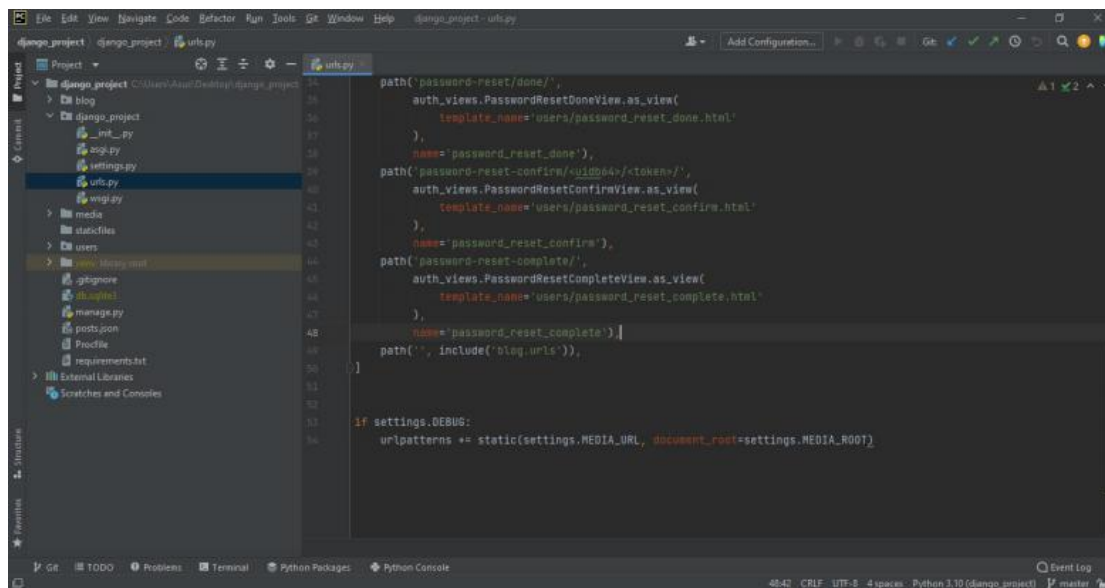


```
126 MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
127 MEDIA_URL = '/media/'
128
129 CRISPY_TEMPLATE_PACK = 'bootstrap4'
130
131 LOGIN_REDIRECT_URL = 'blog-home'
132 LOGIN_URL = 'login'
133
134 EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
135 EMAIL_HOST = 'janopetundog@gmail.com'
136 EMAIL_PORT = 587
137 EMAIL_USE_TLS = True
138 EMAIL_HOST_USER = os.environ.get('EMAIL_USER')
139 EMAIL_HOST_PASSWORD = os.environ.get('EMAIL_PASS')
140
141 # Default primary key field type
142 # https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
143
144 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
145
146 django_heroku.settings(locals())
```

➤ **django_project > django_project > urls.py**



```
14 from django.contrib import admin
15 from django.contrib.auth import views as auth_views
16 from django.urls import path, include
17 from django.conf import settings
18 from django.conf.urls.static import static
19 from users import views as user_views
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('register/', user_views.register, name='register'),
24     path('profile/', user_views.profile, name='profile'),
25     path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name='login'),
26     path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.html'), name='logout'),
27     path('password-reset/',
28         auth_views.PasswordResetView.as_view(
29             template_name='users/password_reset.html'
30         ),
31         name='password_reset'),
32     path('password-reset/done/',
33         auth_views.PasswordResetDoneView.as_view(
34             template_name='users/password_reset_done.html'
35         ),
36         name='password_reset_done'),
37     path('password-reset-confirm<uidb64>/<token>/',
38         auth_views.PasswordResetConfirmView.as_view(
```

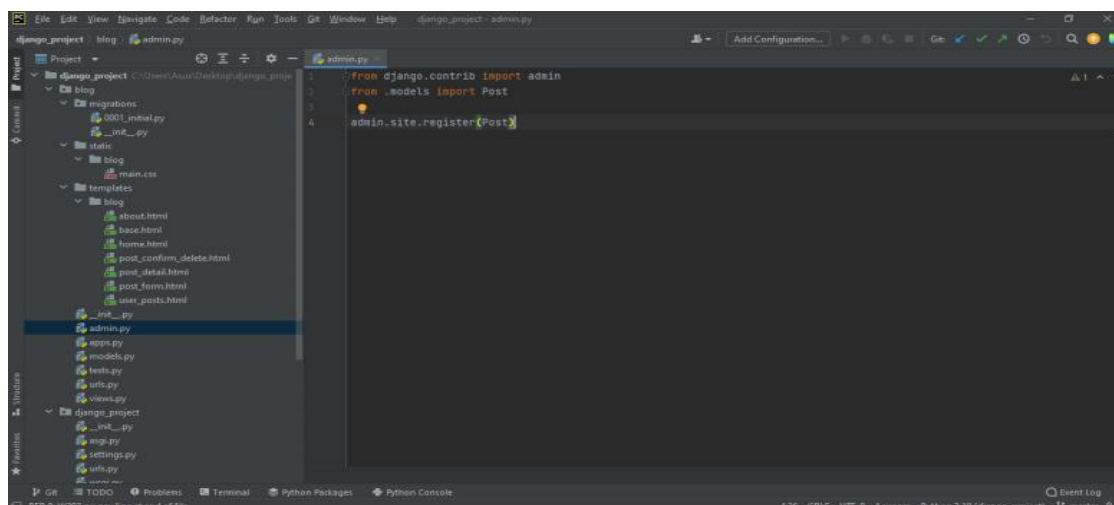


```
path('password-reset/done/',
    auth_views.PasswordResetDoneView.as_view(
        template_name='users/password_reset_done.html'
    ),
    name='password_reset_done'),
path('password-reset-confirm/<uidb64>/<token>/',
    auth_views.PasswordResetConfirmView.as_view(
        template_name='users/password_reset_confirm.html'
    ),
    name='password_reset_confirm'),
path('password-reset-complete/',
    auth_views.PasswordResetCompleteView.as_view(
        template_name='users/password_reset_complete.html'
    ),
    name='password_reset_complete')
path('', include('blog.urls'))

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

BLOG APP

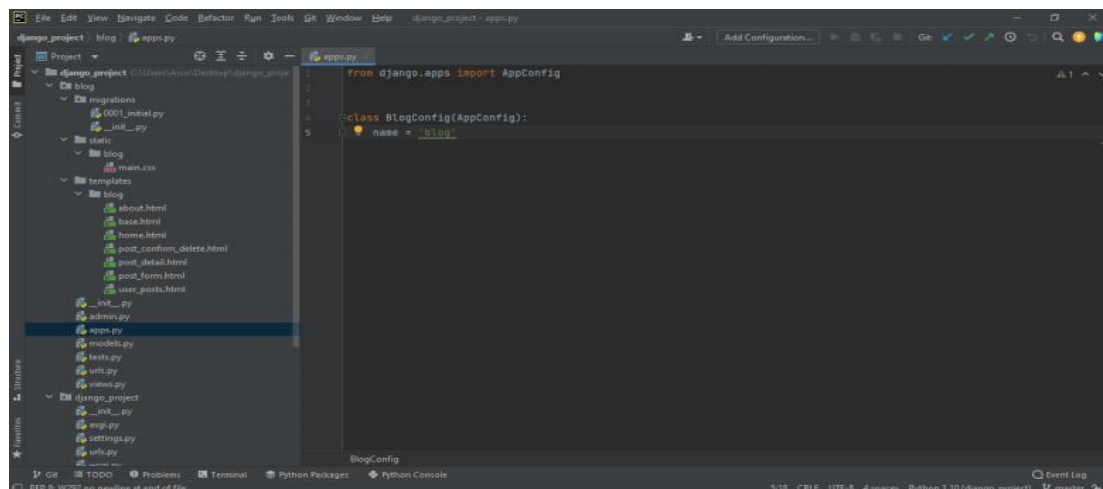
➤ `django_project > blog > admin.py`



```
from django.contrib import admin
from .models import Post

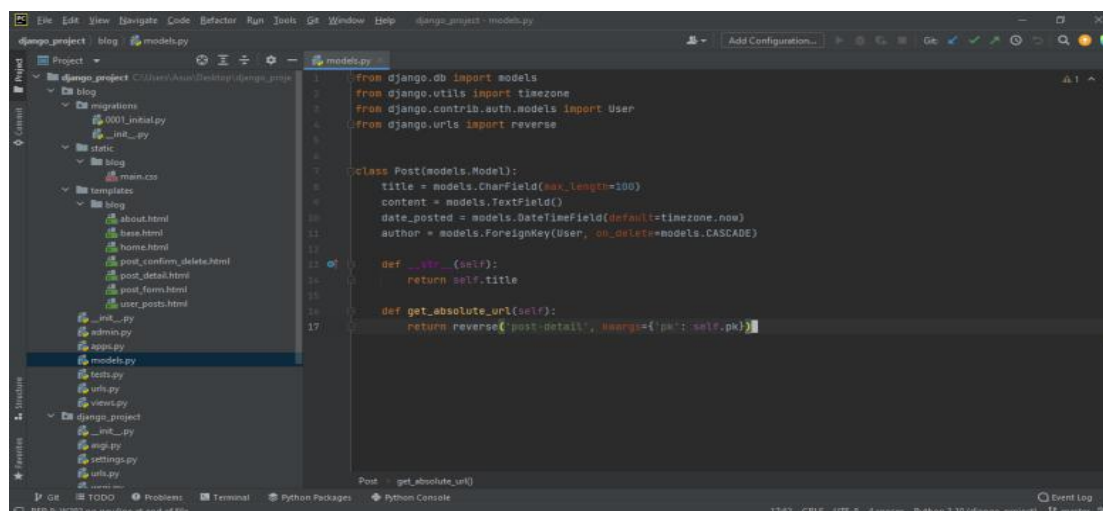
admin.site.register(Post)
```

➤ **django_project > blog > apps.py**



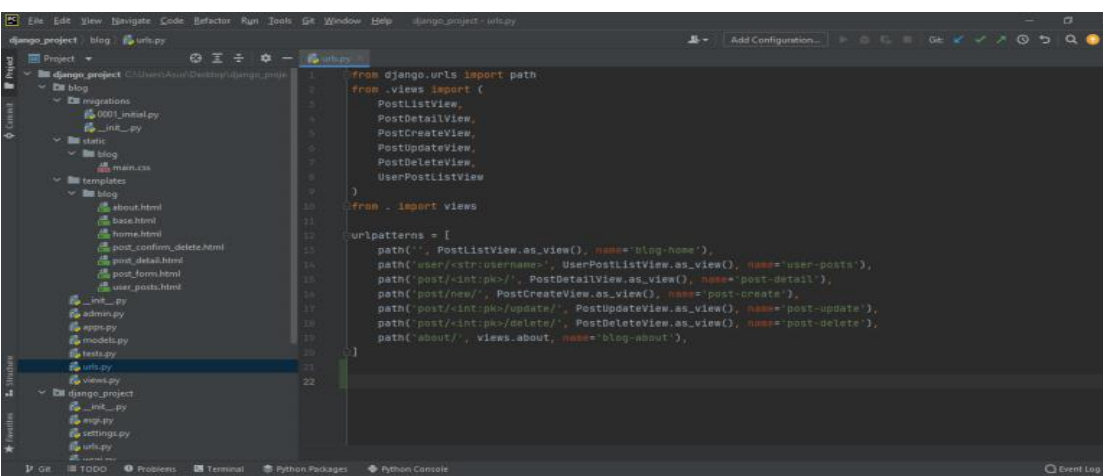
```
1 from django.apps import AppConfig
2
3 class BlogConfig(AppConfig):
4     name = 'blog'
```

➤ **django_project > blog > models.py**



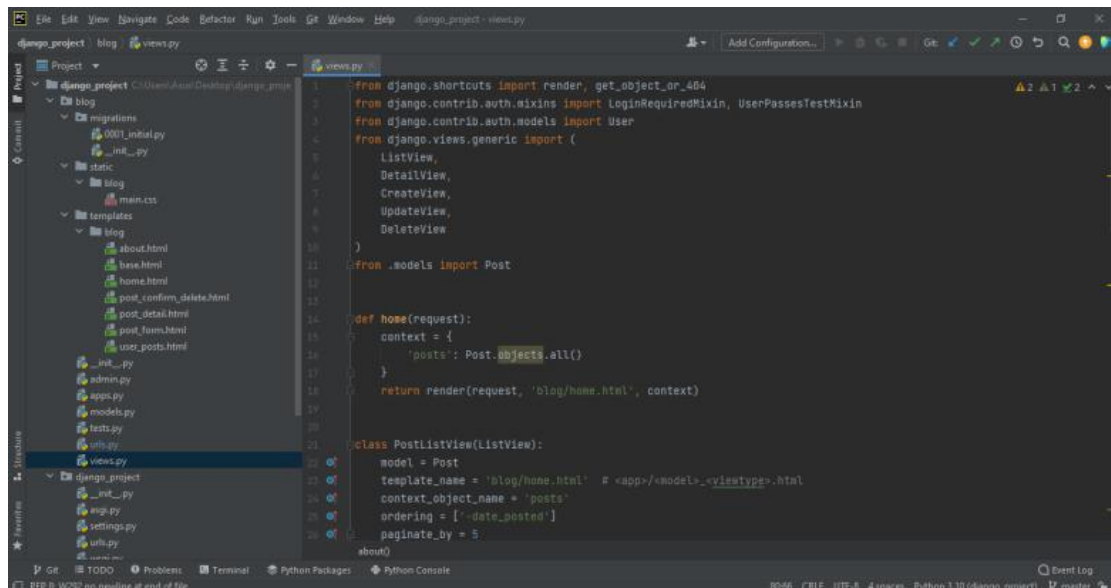
```
1 from django.db import models
2 from django.utils import timezone
3 from django.contrib.auth.models import User
4 from django.urls import reverse
5
6 class Post(models.Model):
7     title = models.CharField(max_length=100)
8     content = models.TextField()
9     date_posted = models.DateTimeField(default=timezone.now)
10    author = models.ForeignKey(User, on_delete=models.CASCADE)
11
12    def __str__(self):
13        return self.title
14
15    def get_absolute_url(self):
16        return reverse('post-detail', kwargs={'pk': self.pk})
```

➤ **django_project > blog > urls.py**



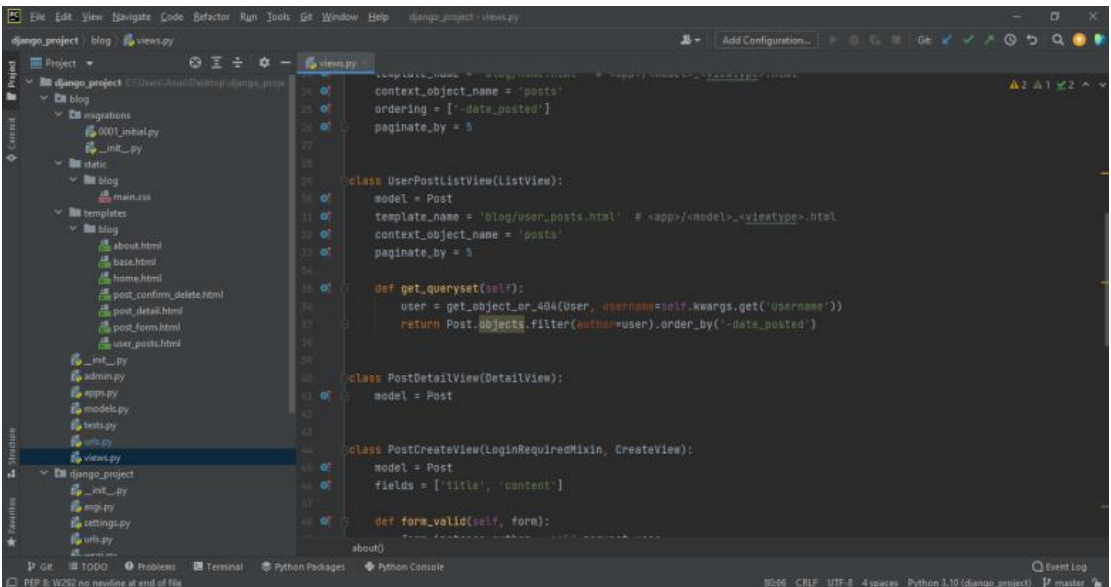
```
1 from django.urls import path
2 from .views import (
3     PostListView,
4     PostDetailView,
5     PostCreateView,
6     PostUpdateView,
7     PostDeleteView,
8     UserPostListView
9 )
10
11 from . import views
12
13 urlpatterns = [
14     path('', PostListView.as_view(), name='blog-home'),
15     path('user/<str:username>', UserPostListView.as_view(), name='user-posts'),
16     path('post/<int:pk>', PostDetailView.as_view(), name='post-detail'),
17     path('post/new/', PostCreateView.as_view(), name='post-create'),
18     path('post/<int:pk>/update/', PostUpdateView.as_view(), name='post-update'),
19     path('post/<int:pk>/delete/', PostDeleteView.as_view(), name='post-delete'),
20     path('about/', views.about, name='blog-about'),
21 ]
```

➤ django_project > blog > views.py



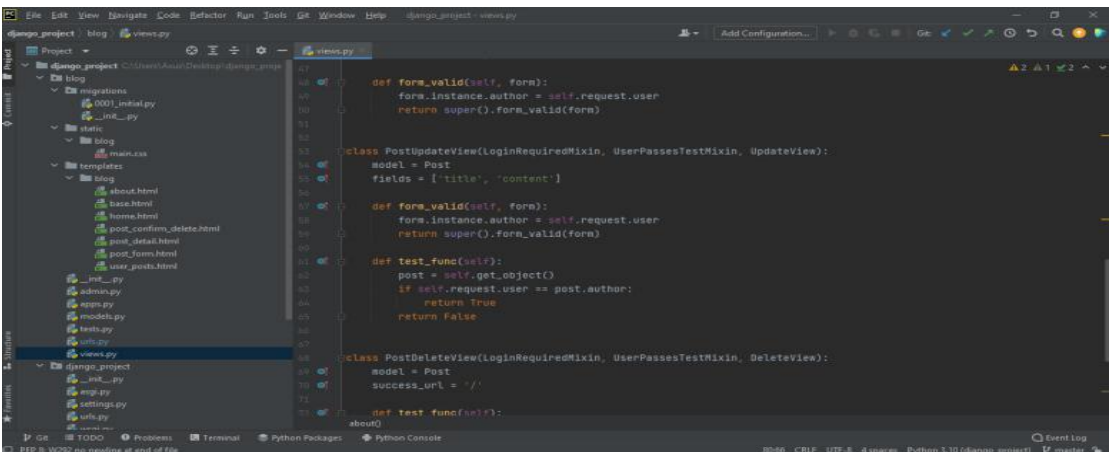
The screenshot shows a code editor with the Django Project Explorer on the left. The 'blog' app is selected, and the 'views.py' file is open. The code defines a 'home' function and a 'PostListView' class. The 'home' function uses 'render' to display a list of posts. 'PostListView' inherits from 'ListView' and is configured with 'Post' as the model, 'blog/home.html' as the template, and 'posts' as the context object name. It also sets 'ordering' to '-date_posted' and 'paginate_by' to 5.

```
1 from django.shortcuts import render, get_object_or_404
2 from django.contrib.auth.mixins import LoginRequiredMixin, UserPassesTestMixin
3 from django.contrib.auth.models import User
4 from django.views.generic import (
5     ListView,
6     DetailView,
7     CreateView,
8     UpdateView,
9     DeleteView
10 )
11
12 from .models import Post
13
14 def home(request):
15     context = {
16         'posts': Post.objects.all()
17     }
18     return render(request, 'blog/home.html', context)
19
20 class PostListView(ListView):
21     model = Post
22     template_name = 'blog/home.html' # <app>/<model>/<viewtype>.html
23     context_object_name = 'posts'
24     ordering = ['-date_posted']
25     paginate_by = 5
26
27     def get_queryset(self):
28         return Post.objects.all()
29
30     def get_context_data(self, **kwargs):
31         context = super().get_context_data(**kwargs)
32         context['paginator'] = self.paginator
33         context['page_obj'] = self.page_obj
34         return context
35
36     def get_template_names(self):
37         return ['blog/home.html']
```



The screenshot shows the same code editor with the 'views.py' file. Additional views have been added: 'UserPostListView' (a class-based view for user posts), 'PostDetailView' (a class-based view for a single post), 'PostCreateView' (a class-based view for creating a new post), and 'PostUpdateView' (a class-based view for updating a post). The 'PostDetailView' uses 'DetailView' and 'Post' as the model. 'PostCreateView' inherits from 'LoginRequiredMixin' and 'CreateView', with 'Post' as the model and 'title' and 'content' as the fields. 'PostUpdateView' inherits from 'LoginRequiredMixin', 'UserPassesTestMixin', and 'UpdateView', with 'Post' as the model and 'title' and 'content' as the fields. A 'test_func' method is added to 'PostUpdateView' to ensure only the author can update a post. The 'PostDeleteView' class is also defined but its code is partially obscured.

```
26 class UserPostListView(ListView):
27     model = Post
28     template_name = 'blog/user_posts.html' # <app>/<model>/<viewtype>.html
29     context_object_name = 'posts'
30     paginate_by = 5
31
32     def get_queryset(self):
33         user = get_object_or_404(User, username=self.kwargs.get('username'))
34         return Post.objects.filter(author=user).order_by('-date_posted')
35
36 class PostDetailView(DetailView):
37     model = Post
38
39 class PostCreateView(LoginRequiredMixin, CreateView):
40     model = Post
41     fields = ['title', 'content']
42
43     def form_valid(self, form):
44         self.request.user.save_post(form.cleaned_data)
45         return super().form_valid(form)
46
47 class PostUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
48     model = Post
49     fields = ['title', 'content']
50
51     def form_valid(self, form):
52         form.instance.author = self.request.user
53         return super().form_valid(form)
54
55     def test_func(self):
56         post = self.get_object()
57         if self.request.user == post.author:
58             return True
59         return False
60
61 class PostDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
62     model = Post
63     success_url = '/'
64
65     def test_func(self):
66         post = self.get_object()
67         if self.request.user == post.author:
68             return True
69         return False
```



The screenshot shows the same code editor with the 'views.py' file. The final set of views is shown, including the 'PostDeleteView' class. The 'PostDeleteView' inherits from 'LoginRequiredMixin', 'UserPassesTestMixin', and 'DeleteView', with 'Post' as the model and '/' as the success URL. A 'test_func' method is added to 'PostDeleteView' to ensure only the author can delete a post. The 'PostUpdateView' class is also shown with its 'test_func' method.

```
67 class PostDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
68     model = Post
69     success_url = '/'
70
71     def test_func(self):
72         post = self.get_object()
73         if self.request.user == post.author:
74             return True
75         return False
76
77 def main():
78     print('Hello World!')
```

```

def test_func(self):
    post = self.get_object()
    if self.request.user == post.author:
        return True
    return False

class PostDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
    model = Post
    success_url = '/'

    def test_func(self):
        post = self.get_object()
        if self.request.user == post.author:
            return True
        return False

def about(request):
    return render(request, 'blog/about.html', {'title': 'About'})

```

BLOG APP STATIC

➤ `django_project > blog > static > blog > main.css`

```

body {
    background: #fafafa;
    color: #333333;
    margin-top: 5rem;
}

h1, h2, h3, h4, h5, h6 {
    color: #444444;
}

ul {
    margin: 0;
}

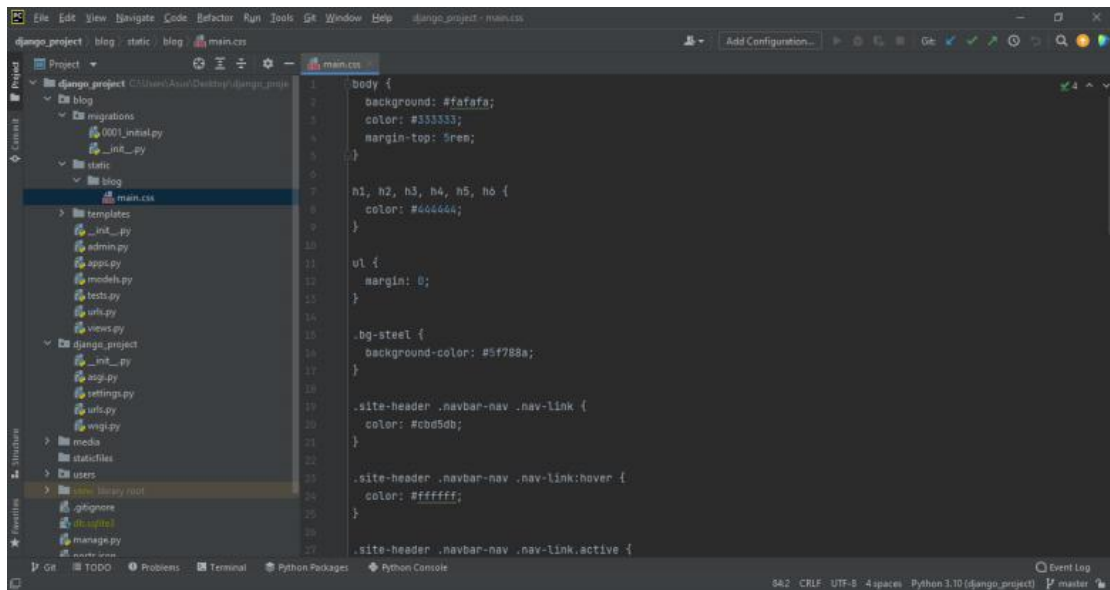
.bg-steel {
    background-color: #5f788a;
}

.site-header .navbar-nav .nav-link {
    color: #c0c0c0;
}

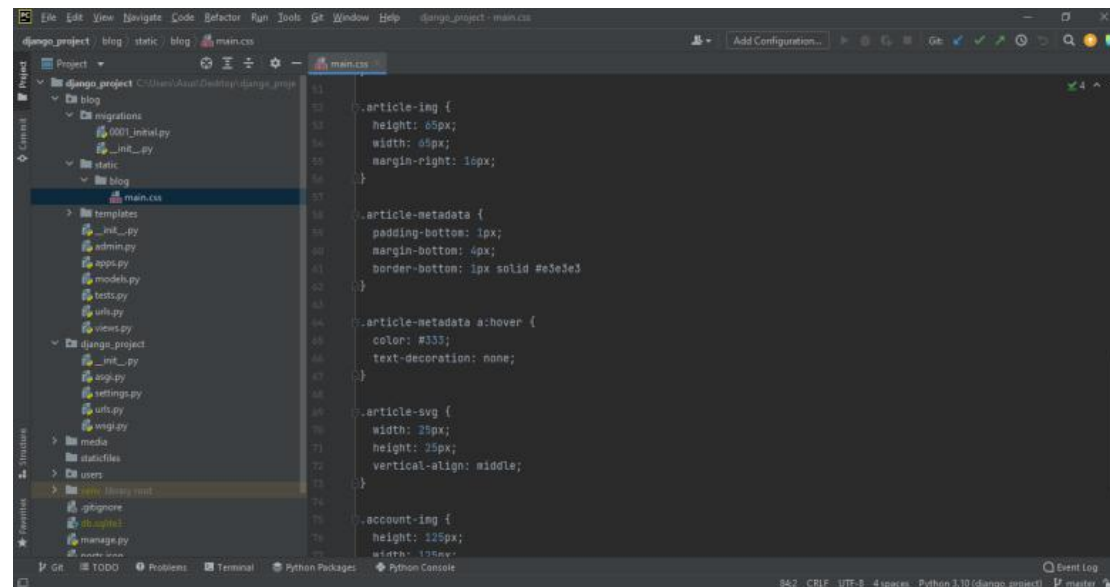
.site-header .navbar-nav .nav-link:hover {
    color: #ffffff;
}

.site-header .navbar-nav .nav-link.active {
    color: #ffffff;
}

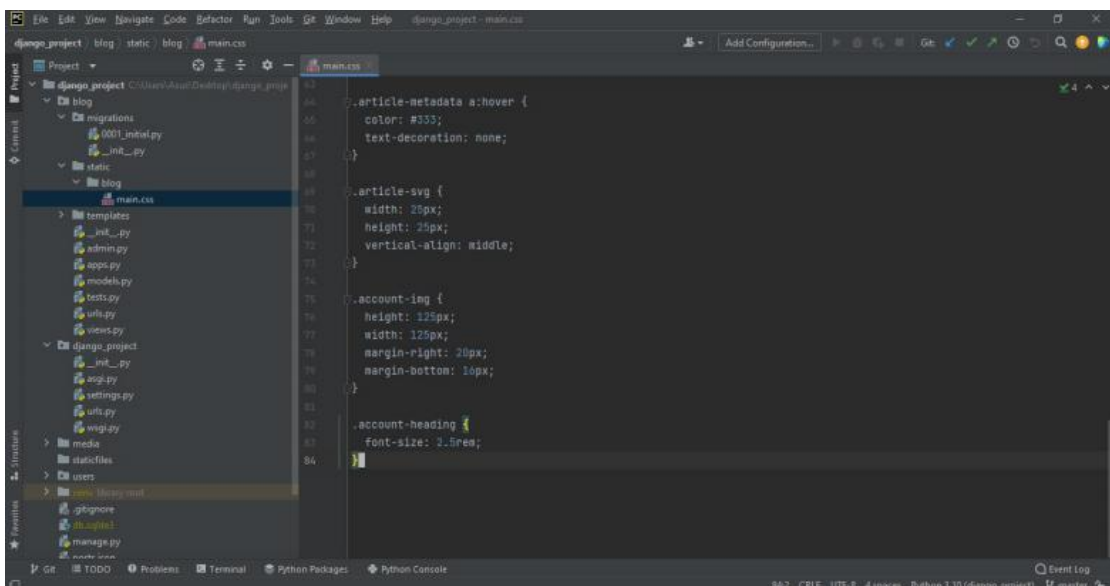
```

```
1 body {
2   background: #fafafa;
3   color: #333333;
4   margin-top: 5rem;
5 }
6
7 h1, h2, h3, h4, h5, h6 {
8   color: #666666;
9 }
10
11 ul {
12   margin: 0;
13 }
14
15 .bg-steel {
16   background-color: #5f788a;
17 }
18
19 .site-header .navbar-nav .nav-link {
20   color: #cbe5db;
21 }
22
23 .site-header .navbar-nav .nav-link:active {
24   color: #ffffff;
25 }
26
27 .site-header .navbar-nav .nav-link.active {
```



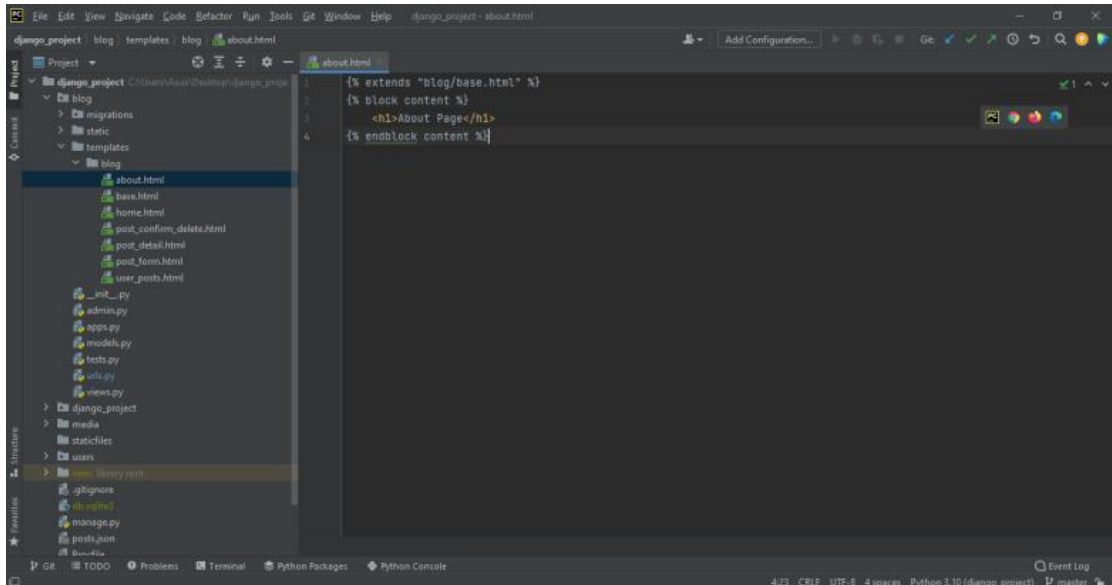
```
51
52
53 .article-img {
54   height: 85px;
55   width: 85px;
56   margin-right: 10px;
57 }
58
59 .article-metadata {
60   padding-bottom: 1px;
61   margin-bottom: 4px;
62   border-bottom: 1px solid #e3e3e3;
63 }
64
65 .article-metadata a:active {
66   color: #333;
67   text-decoration: none;
68 }
69
70 .article-svg {
71   width: 25px;
72   height: 25px;
73   vertical-align: middle;
74 }
75
76 .account-img {
77   height: 125px;
78   width: 125px;
```



```
63
64 .article-metadata a:active {
65   color: #333;
66   text-decoration: none;
67 }
68
69 .article-svg {
70   width: 25px;
71   height: 25px;
72   vertical-align: middle;
73 }
74
75 .account-img {
76   height: 125px;
77   width: 125px;
78   margin-right: 20px;
79   margin-bottom: 10px;
80 }
81
82 .account-heading {
83   font-size: 2.5rem;
84 }
```

BLOG APP TEMPLATES

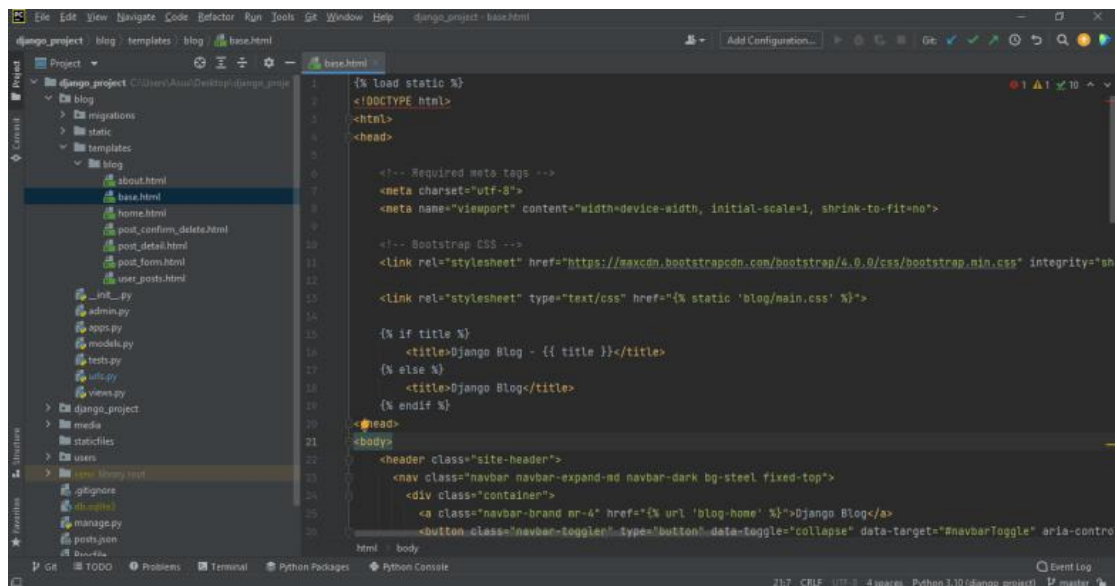
- `django_project > blog > templates > blog > about.html`



The screenshot shows the Visual Studio Code editor with the file `about.html` open. The file explorer on the left shows the project structure, with `about.html` selected under `blog > templates > blog`. The code editor displays the following HTML template:

```
1 {% extends "blog/base.html" %}
2 {% block content %}
3     <h1>About Page</h1>
4 {% endblock content %}
```

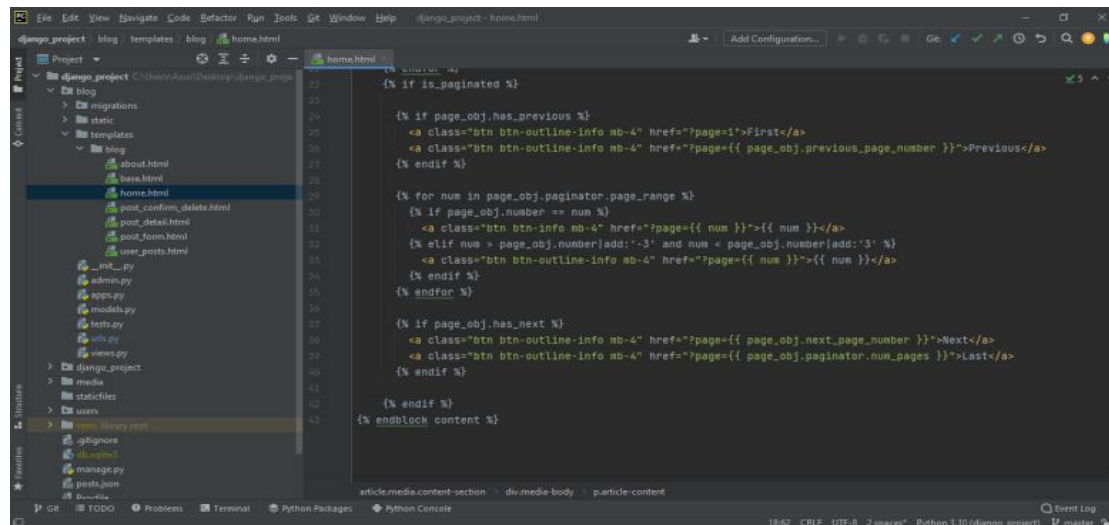
- `django_project > blog > templates > blog > base.html`



The screenshot shows the Visual Studio Code editor with the file `base.html` open. The file explorer on the left shows the project structure, with `base.html` selected under `blog > templates > blog`. The code editor displays the following HTML template:

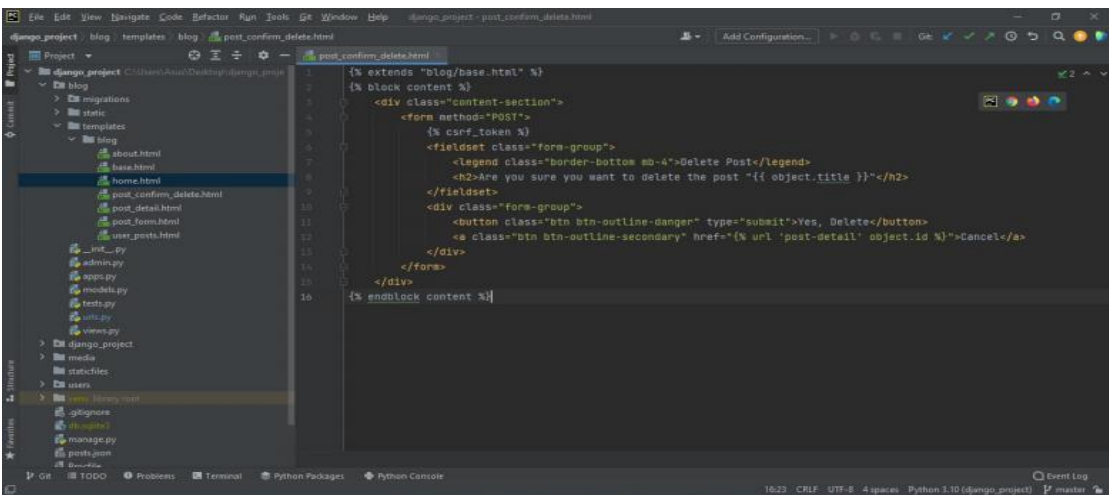
```
1 {% load static %}
2 <!DOCTYPE html>
3 <html>
4 <head>
5
6     <!-- Required meta tags -->
7     <meta charset="utf-8">
8     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
9
10    <!-- Bootstrap CSS -->
11    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-I..
12
13    <link rel="stylesheet" type="text/css" href="{% static 'blog/main.css' %}">
14
15    {% if title %}
16    <title>Django Blog - {{ title }}</title>
17    {% else %}
18    <title>Django Blog</title>
19    {% endif %}
20 </head>
21 <body>
22
23     <header class="site-header">
24         <nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">
25             <div class="container">
26                 <a class="navbar-brand mr-4" href="{% url 'blog-home' %}">Django Blog</a>
27                 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarToggle" aria-contro
28             </div>
29         </nav>
30     </header>
31
32     <div class="container">
33         <div class="row">
34             <div class="col-md-8">
35                 <div class="card">
36                     <div class="card-body">
37                         <h2>Blog</h2>
38                     </div>
39                 </div>
40             </div>
41             <div class="col-md-4">
42                 <div class="card">
43                     <div class="card-body">
44                         <h2>About</h2>
45                     </div>
46                 </div>
47             </div>
48         </div>
49     </div>
50 </body>
51 </html>
```


django_project > blog > templates > blog > home.html



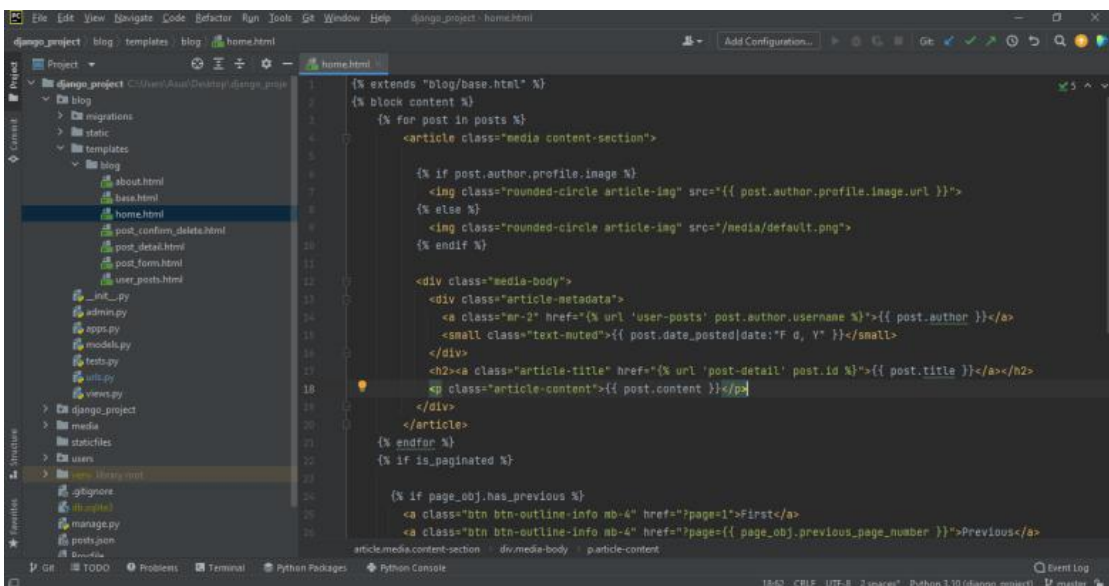
The screenshot shows the Django project structure in VS Code. The file explorer on the left shows the project hierarchy: django_project > blog > templates > blog > home.html. The main editor displays the content of home.html, which includes pagination logic and a loop for displaying posts.

```
1 {% extends "blog/base.html" %}
2 {% block content %}
3     {% if is_paginated %}
4
5     {% if page_obj.has_previous %}
6         <a class="btn btn-outline-info mb-4" href="?page=1">First</a>
7         <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.previous_page_number }}">Previous</a>
8     {% endif %}
9
10    {% for num in page_obj.paginator.page_range %}
11        {% if page_obj.number == num %}
12            <a class="btn btn-info mb-4" href="?page={{ num }}">{{ num }}</a>
13        {% elif num > page_obj.number|add:'-3' and num < page_obj.number|add:'3' %}
14            <a class="btn btn-outline-info mb-4" href="?page={{ num }}">{{ num }}</a>
15        {% endif %}
16    {% endfor %}
17
18    {% if page_obj.has_next %}
19        <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.next_page_number }}">Next</a>
20        <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.paginator.num_pages }}">Last</a>
21    {% endif %}
22
23    {% endblock content %}
```



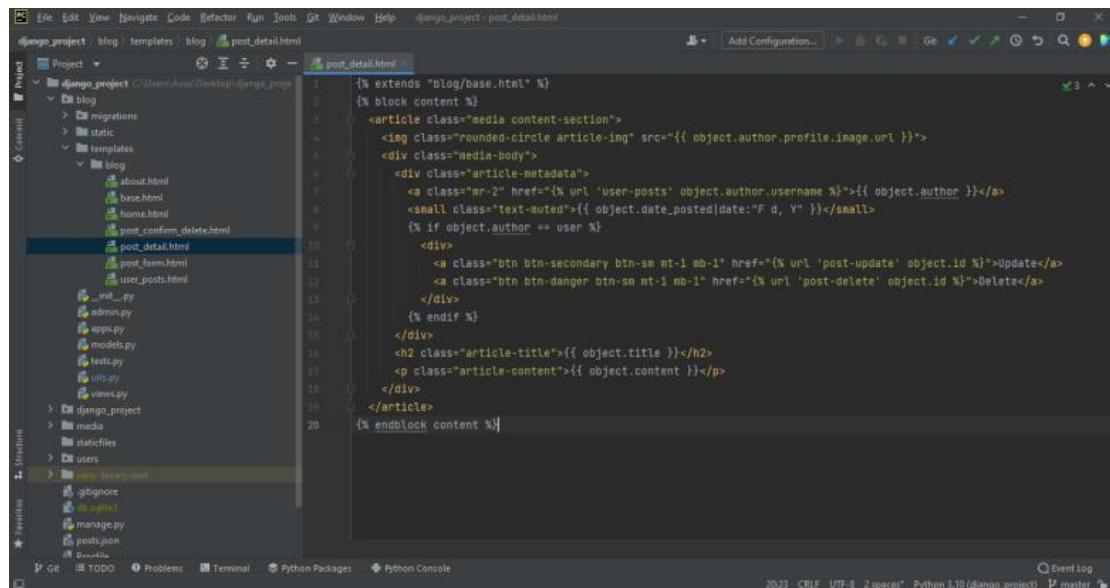
The screenshot shows the content of post_confirm_delete.html, which is a Django view template for deleting a post. It includes a form with a delete button and a confirmation message.

```
1 {% extends "blog/base.html" %}
2 {% block content %}
3     <div class="content-section">
4         <form method="POST">
5             {% csrf_token %}
6             <fieldset class="form-group">
7                 <legend class="border-bottom mb-4">Delete Post</legend>
8                 <h2>Are you sure you want to delete the post "{{ object.title }}"</h2>
9             </fieldset>
10            <div class="form-group">
11                <button class="btn btn-outline-danger" type="submit">Yes, Delete</button>
12                <a class="btn btn-outline-secondary" href="{% url 'post-detail' object.id %}">Cancel</a>
13            </div>
14        </form>
15    </div>
16    {% endblock content %}
```



The screenshot shows the content of home.html, which is a Django view template for displaying a list of posts. It includes a loop for displaying posts and pagination logic.

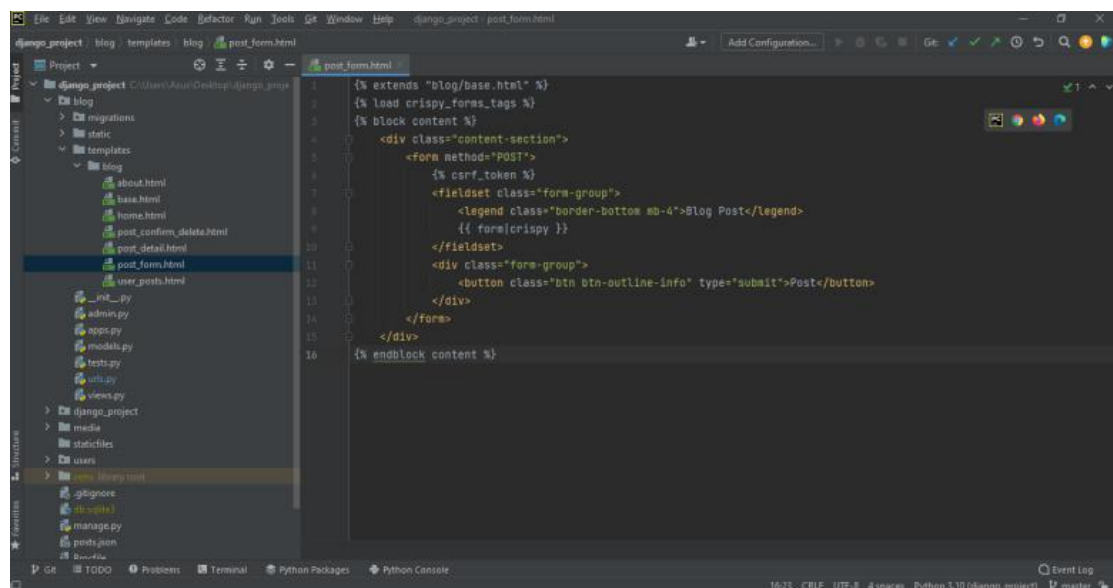
```
1 {% extends "blog/base.html" %}
2 {% block content %}
3     {% for post in posts %}
4         <article class="media content-section">
5
6             {% if post.author.profile.image %}
7                 
8             {% else %}
9                 
10            {% endif %}
11
12            <div class="media-body">
13                <div class="article-metadata">
14                    <a class="mr-2" href="{% url 'user-posts' post.author.username %}">{{ post.author }}</a>
15                    <small class="text-muted">{{ post.date_posted|date:"F d, Y" }}</small>
16                </div>
17                <h2><a class="article-title" href="{% url 'post-detail' post.id %}">{{ post.title }}</a></h2>
18                <div class="article-content">{{ post.content }}</div>
19            </div>
20        </article>
21    {% endfor %}
22    {% if is_paginated %}
23
24    {% if page_obj.has_previous %}
25        <a class="btn btn-outline-info mb-4" href="?page=1">First</a>
26        <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.previous_page_number }}">Previous</a>
27    {% endif %}
28
29    {% for num in page_obj.paginator.page_range %}
30        {% if page_obj.number == num %}
31            <a class="btn btn-info mb-4" href="?page={{ num }}">{{ num }}</a>
32        {% elif num > page_obj.number|add:'-3' and num < page_obj.number|add:'3' %}
33            <a class="btn btn-outline-info mb-4" href="?page={{ num }}">{{ num }}</a>
34        {% endif %}
35    {% endfor %}
36
37    {% if page_obj.has_next %}
38        <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.next_page_number }}">Next</a>
39        <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.paginator.num_pages }}">Last</a>
40    {% endif %}
41
42    {% endblock content %}
```



The screenshot shows the Visual Studio Code editor with a Django project open. The file explorer on the left shows the project structure: `django_project` > `blog` > `templates` > `blog`. The file `post_detail.html` is selected and its content is displayed in the editor. The code is a Django template that extends `blog/base.html` and uses Django's template language to display a post's details, including the author's profile image, username, date posted, and buttons for updating or deleting the post.

```
1 {% extends "blog/base.html" %}
2 {% block content %}
3 <article class="media content-section">
4   
5   <div class="media-body">
6     <div class="article-metadata">
7       <a class="mr-2" href="{% url 'user-posts' object.author.username %}">{{ object.author }}</a>
8       <small class="text-muted">{{ object.date_posted|date:"F d, Y" }}</small>
9       {% if object.author == user %}
10         <div>
11           <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{% url 'post-update' object.id %}">Update</a>
12           <a class="btn btn-danger btn-sm mt-1 mb-1" href="{% url 'post-delete' object.id %}">Delete</a>
13         </div>
14       {% endif %}
15     </div>
16     <h2 class="article-title">{{ object.title }}</h2>
17     <p class="article-content">{{ object.content }}</p>
18   </div>
19 </article>
20 {% endblock content %}
```

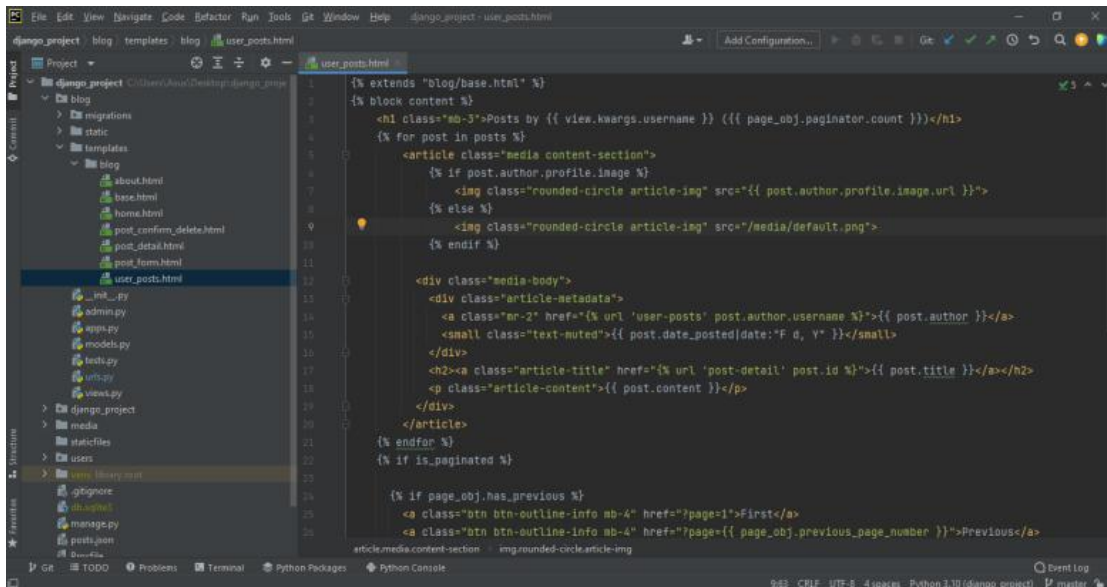
➤ `django_project > blog > templates > blog > post_confirm_delete.html`



The screenshot shows the Visual Studio Code editor with the same Django project. The file explorer on the left shows the project structure: `django_project` > `blog` > `templates` > `blog`. The file `post_form.html` is selected and its content is displayed in the editor. The code is a Django template that extends `blog/base.html` and uses Django's template language to display a form for creating a new post. The form includes a CSRF token, a legend, and a submit button.

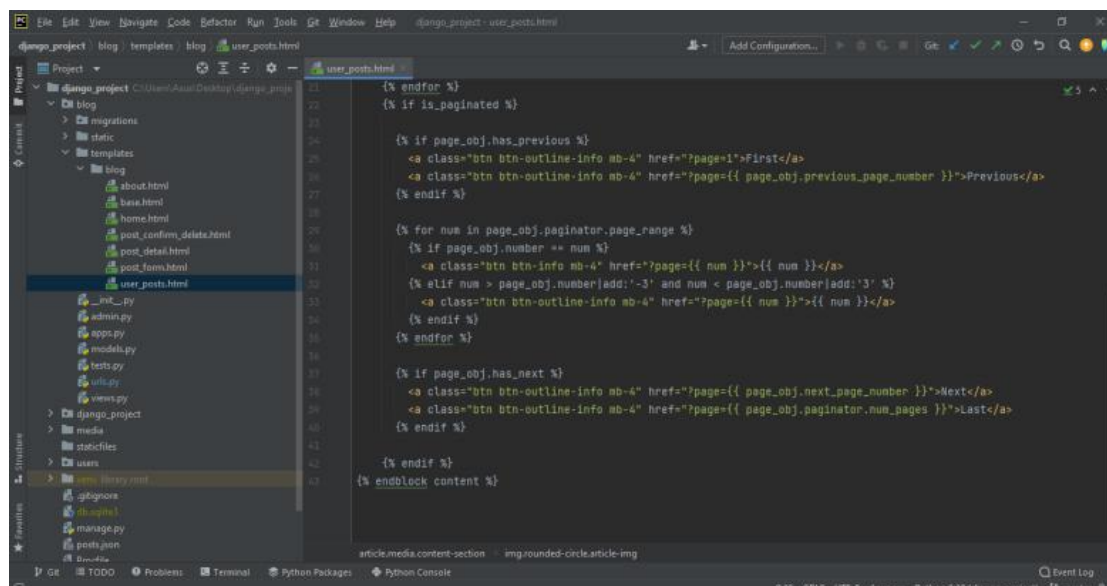
```
1 {% extends "blog/base.html" %}
2 {% load crispy_forms_tags %}
3 {% block content %}
4   <div class="content-section">
5     <form method="POST">
6       {{ csrf.token }}
7       <fieldset class="form-group">
8         <legend class="border-bottom mb-4">Blog Post</legend>
9         {{ form|crispy }}
10       </fieldset>
11       <div class="form-group">
12         <button class="btn btn-outline-info" type="submit">Post</button>
13       </div>
14     </form>
15   </div>
16 {% endblock content %}
```

➤ **django_project > blog > templates > blog > post_detail.html**



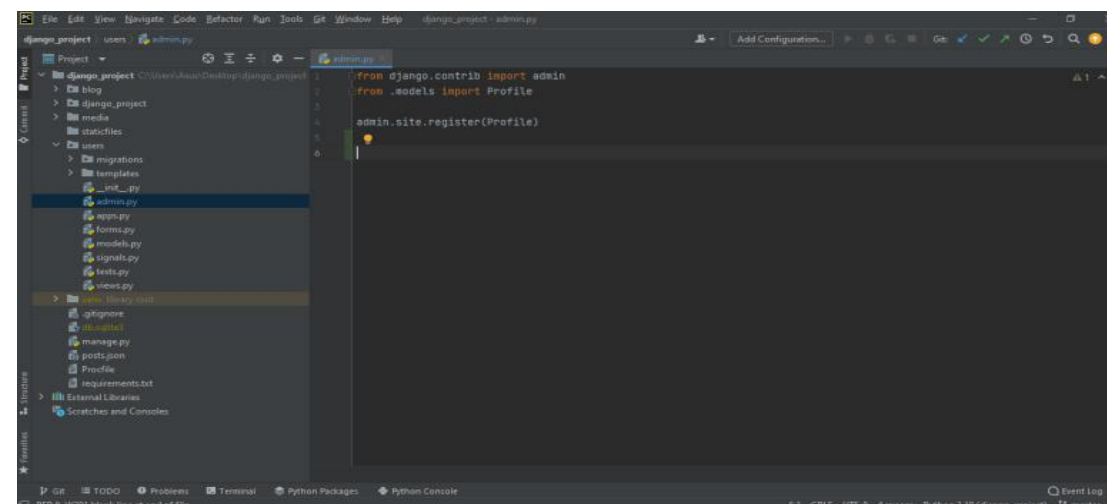
```
1 {% extends "blog/base.html" %}
2 {% block content %}
3     <h1 class="mb-3">Posts by {{ view.kwargs.username }} ({{ page_obj.paginator.count }})</h1>
4     {% for post in posts %}
5         <article class="media content-section">
6             {% if post.author.profile.image %}
7                 
8             {% else %}
9                 
10            {% endif %}
11
12            <div class="media-body">
13                <div class="article-metadata">
14                    <a class="mr-2" href="{% url 'user-posts' post.author.username %}">{{ post.author }}</a>
15                    <small class="text-muted">{{ post.date_posted|date:"F d, Y" }}</small>
16                </div>
17                <h2><a class="article-title" href="{% url 'post-detail' post.id %}">{{ post.title }}</a></h2>
18                <p class="article-content">{{ post.content }}</p>
19            </div>
20        </article>
21    {% endfor %}
22    {% if is_paginated %}
23
24        {% if page_obj.has_previous %}
25            <a class="btn btn-outline-info mb-4" href="?page=1">First</a>
26            <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.previous_page_number }}">Previous</a>
27        {% endif %}
28
29        {% for num in page_obj.paginator.page_range %}
30            {% if page_obj.number == num %}
31                <a class="btn btn-info mb-4" href="?page={{ num }}">{{ num }}</a>
32            {% elif num > page_obj.number|add:'-3' and num < page_obj.number|add:'3' %}
33                <a class="btn btn-outline-info mb-4" href="?page={{ num }}">{{ num }}</a>
34            {% endif %}
35        {% endfor %}
36
37        {% if page_obj.has_next %}
38            <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.next_page_number }}">Next</a>
39            <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.paginator.num_pages }}">Last</a>
40        {% endif %}
41    {% endif %}
42{% endblock content %}
```

➤ **django_project > blog > templates > blog > post_form.html**

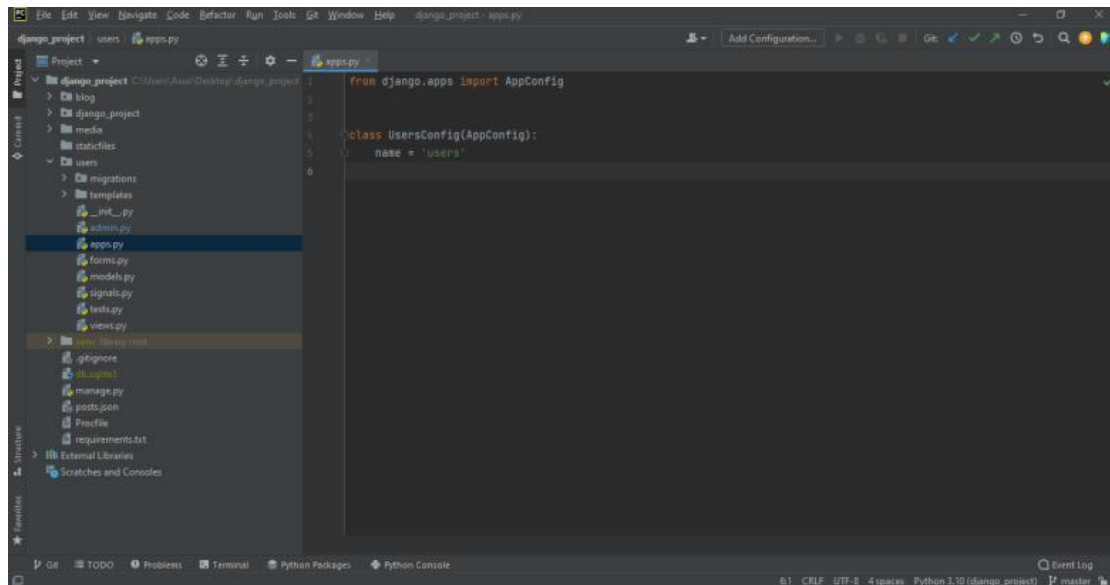


```
1 {% extends "blog/base.html" %}
2 {% block content %}
3     <h1 class="mb-3">Posts by {{ view.kwargs.username }} ({{ page_obj.paginator.count }})</h1>
4     {% for post in posts %}
5         <article class="media content-section">
6             {% if post.author.profile.image %}
7                 
8             {% else %}
9                 
10            {% endif %}
11
12            <div class="media-body">
13                <div class="article-metadata">
14                    <a class="mr-2" href="{% url 'user-posts' post.author.username %}">{{ post.author }}</a>
15                    <small class="text-muted">{{ post.date_posted|date:"F d, Y" }}</small>
16                </div>
17                <h2><a class="article-title" href="{% url 'post-detail' post.id %}">{{ post.title }}</a></h2>
18                <p class="article-content">{{ post.content }}</p>
19            </div>
20        </article>
21    {% endfor %}
22    {% if is_paginated %}
23
24        {% if page_obj.has_previous %}
25            <a class="btn btn-outline-info mb-4" href="?page=1">First</a>
26            <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.previous_page_number }}">Previous</a>
27        {% endif %}
28
29        {% for num in page_obj.paginator.page_range %}
30            {% if page_obj.number == num %}
31                <a class="btn btn-info mb-4" href="?page={{ num }}">{{ num }}</a>
32            {% elif num > page_obj.number|add:'-3' and num < page_obj.number|add:'3' %}
33                <a class="btn btn-outline-info mb-4" href="?page={{ num }}">{{ num }}</a>
34            {% endif %}
35        {% endfor %}
36
37        {% if page_obj.has_next %}
38            <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.next_page_number }}">Next</a>
39            <a class="btn btn-outline-info mb-4" href="?page={{ page_obj.paginator.num_pages }}">Last</a>
40        {% endif %}
41    {% endif %}
42{% endblock content %}
```

➤ **django_project > blog > templates > blog > user_posts.html**



```
1 from django.contrib import admin
2 from .models import Profile
3
4 admin.site.register(Profile)
```

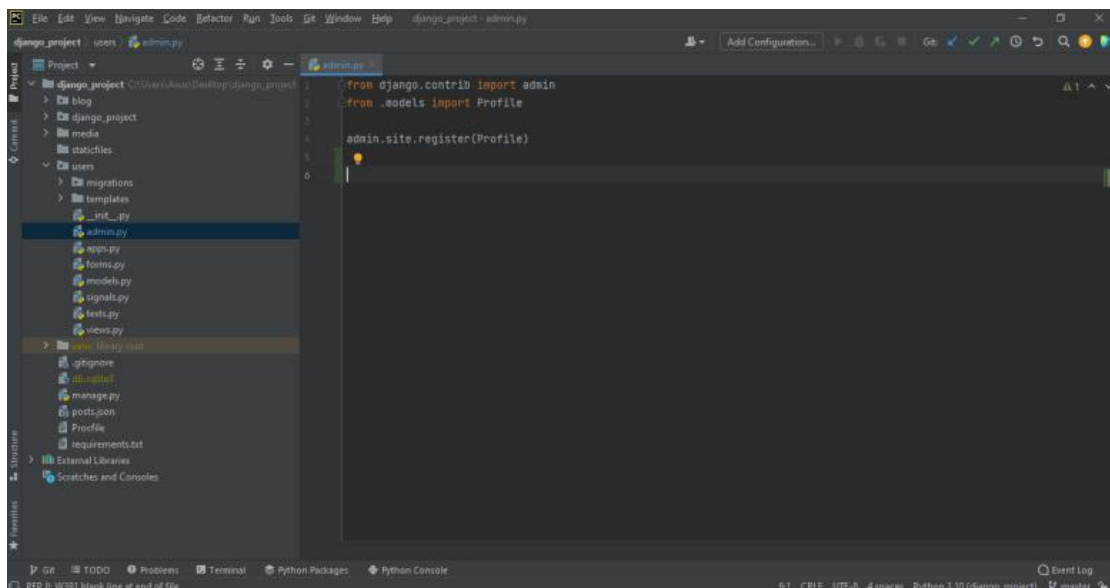



```
from django.apps import AppConfig

class UsersConfig(AppConfig):
    name = 'users'
```

USERS APP

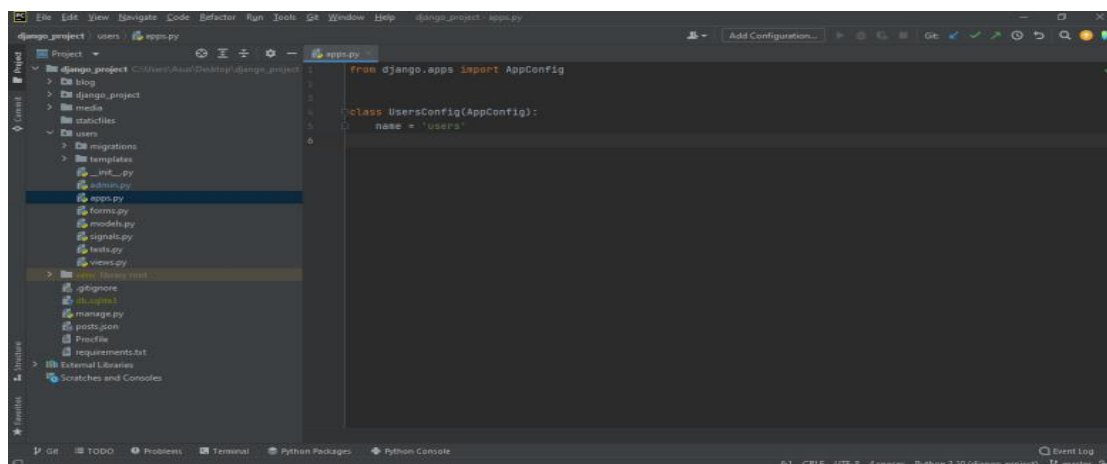
➤ **django_project > users > admin.py**



```
from django.contrib import admin
from .models import Profile

admin.site.register(Profile)
```

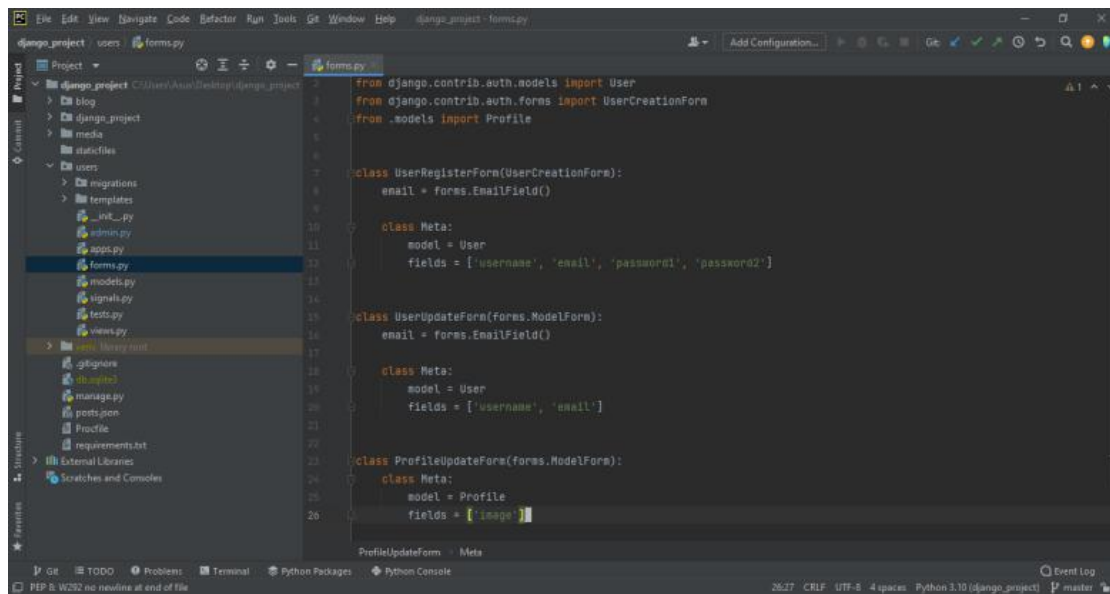
➤ **django_project > users > apps.py**



```
from django.apps import AppConfig

class UsersConfig(AppConfig):
    name = 'users'
```

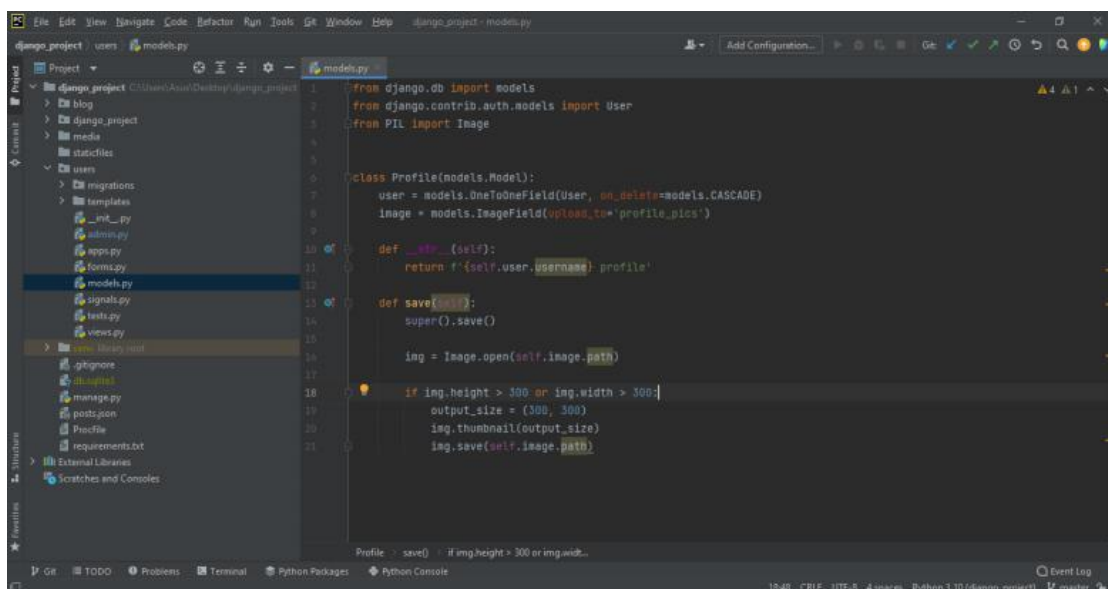
➤ **django_project > users > forms.py**



The screenshot shows the Django Project IDE with the file explorer on the left displaying the project structure. The main editor window shows the contents of `forms.py` in the `users` app. The code defines three Django forms: `UserRegisterForm`, `UserUpdateForm`, and `ProfileUpdateForm`. Each form inherits from `forms.ModelForm` and is associated with a specific model via its `Meta` class's `model` attribute. The `fields` attribute in each `Meta` class lists the form fields to be rendered.

```
1 from django.contrib.auth.models import User
2 from django.contrib.auth.forms import UserCreationForm
3 from .models import Profile
4
5
6
7 class UserRegisterForm(UserCreationForm):
8     email = forms.EmailField()
9
10
11     class Meta:
12         model = User
13         fields = ['username', 'email', 'password1', 'password2']
14
15
16 class UserUpdateForm(forms.ModelForm):
17     email = forms.EmailField()
18
19
20     class Meta:
21         model = User
22         fields = ['username', 'email']
23
24
25 class ProfileUpdateForm(forms.ModelForm):
26
27     class Meta:
28         model = Profile
29         fields = ['image']
```

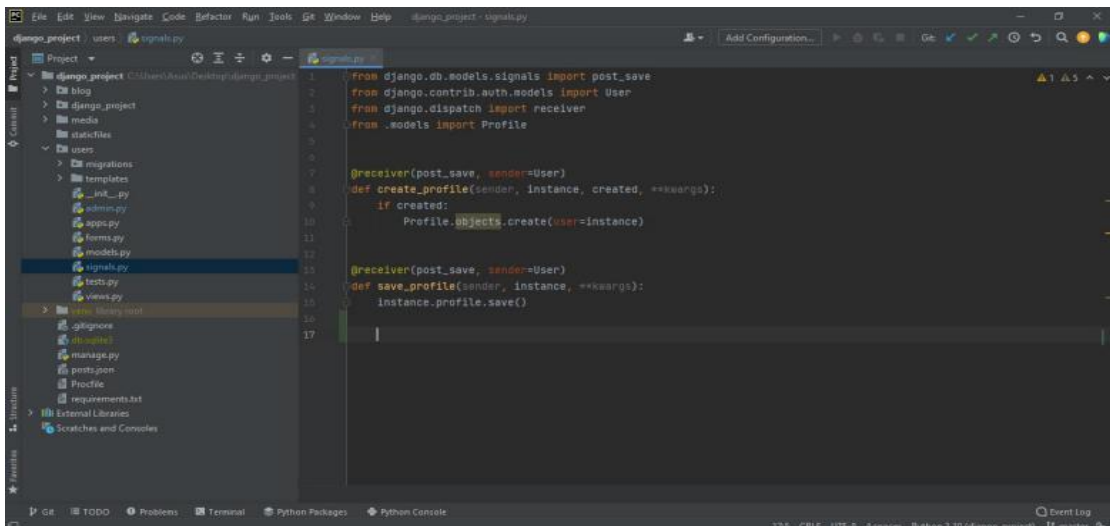
➤ **django_project > users > models.py**



The screenshot shows the Django Project IDE with the file explorer on the left. The main editor window displays the `models.py` file in the `users` app. It defines a `Profile` model that has a one-to-one relationship with the `User` model. The `Profile` model includes an `image` field of type `ImageField`. The `__str__` method returns the profile name, and the `save` method includes logic to generate a thumbnail for the profile image if it is larger than 300x300 pixels.

```
1 from django.db import models
2 from django.contrib.auth.models import User
3 from PIL import Image
4
5
6 class Profile(models.Model):
7     user = models.OneToOneField(User, on_delete=models.CASCADE)
8     image = models.ImageField(upload_to='profile_pics')
9
10
11     def __str__(self):
12         return f'{self.user.username} profile'
13
14
15     def save(self):
16         super().save()
17
18         img = Image.open(self.image.path)
19
20         if img.height > 300 or img.width > 300:
21             output_size = (300, 300)
22             img.thumbnail(output_size)
23             img.save(self.image.path)
```

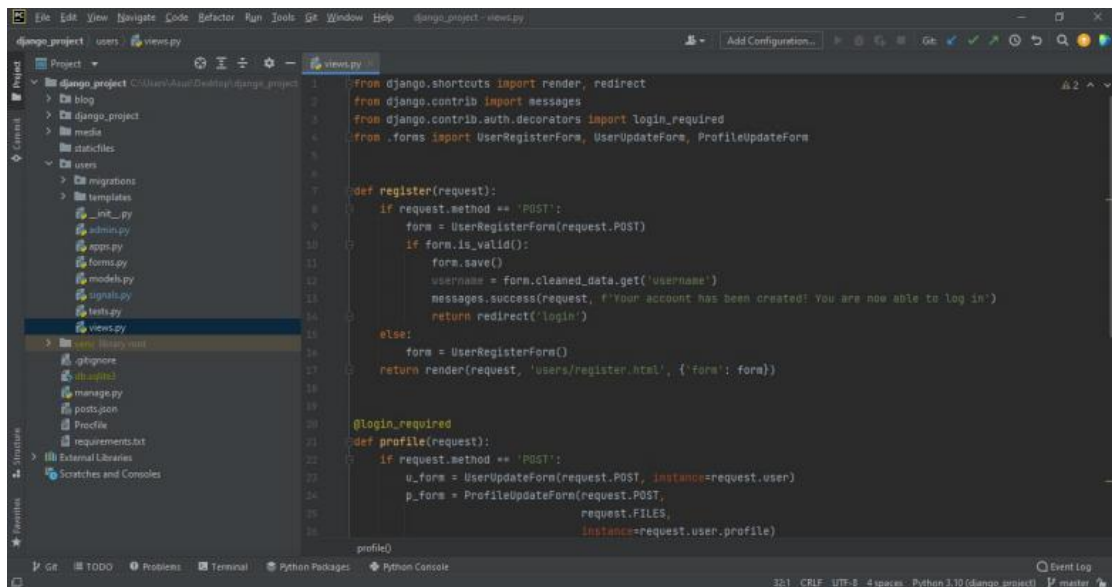
➤ **django_project > users > signals.py**



The screenshot shows the Django Project IDE with the file explorer on the left. The main editor window displays the `signals.py` file in the `users` app. It defines two Django signals: `create_profile` and `save_profile`. Both signals are triggered by the `post_save` signal of the `User` model. The `create_profile` signal handler creates a new `Profile` instance when a new user is created. The `save_profile` signal handler saves the `Profile` instance when a user's profile is updated.

```
1 from django.db.models.signals import post_save
2 from django.contrib.auth.models import User
3 from django.dispatch import receiver
4 from .models import Profile
5
6
7 @receiver(post_save, sender=User)
8 def create_profile(sender, instance, created, **kwargs):
9     if created:
10         Profile.objects.create(user=instance)
11
12
13 @receiver(post_save, sender=User)
14 def save_profile(sender, instance, **kwargs):
15     instance.profile.save()
16
17
```

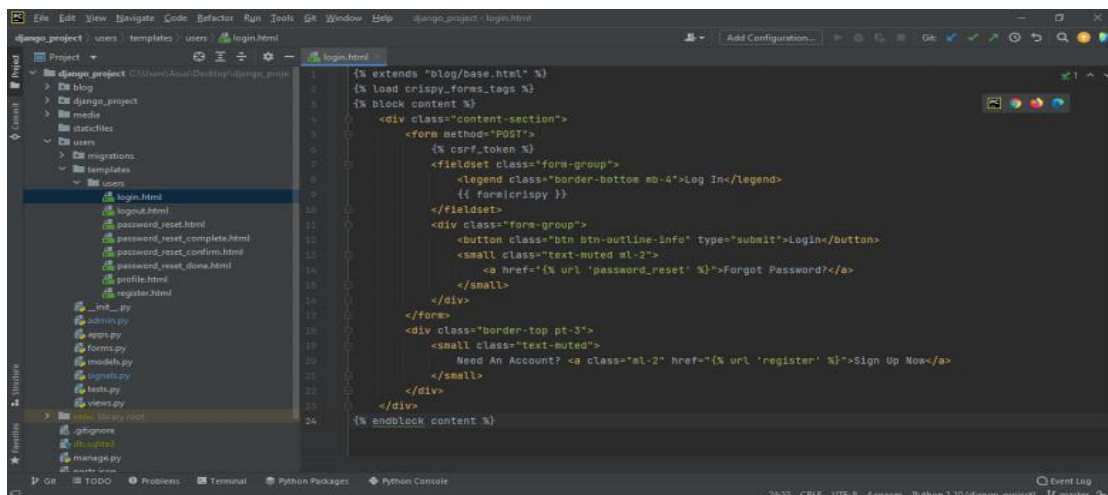

➤ django_project > users > views.py



```
1 from django.shortcuts import render, redirect
2 from django.contrib import messages
3 from django.contrib.auth.decorators import login_required
4 from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm
5
6
7 def register(request):
8     if request.method == 'POST':
9         form = UserRegisterForm(request.POST)
10        if form.is_valid():
11            form.save()
12            username = form.cleaned_data.get('username')
13            messages.success(request, f'Your account has been created! You are now able to log in')
14            return redirect('login')
15        else:
16            form = UserRegisterForm()
17        return render(request, 'users/register.html', {'form': form})
18
19 @login_required
20 def profile(request):
21     if request.method == 'POST':
22         u_form = UserUpdateForm(request.POST, instance=request.user)
23         p_form = ProfileUpdateForm(request.POST,
24                                 request.FILES,
25                                 instance=request.user.profile)
26         if u_form.is_valid() and p_form.is_valid():
27             u_form.save()
28             p_form.save()
29             messages.success(request, f'Your account has been updated!')
30             return redirect('profile')
31         else:
32             u_form = UserUpdateForm(instance=request.user)
33             p_form = ProfileUpdateForm(instance=request.user.profile)
34
35     context = {
36         'u_form': u_form,
37         'p_form': p_form
38     }
39     return render(request, 'users/profile.html', context)
```

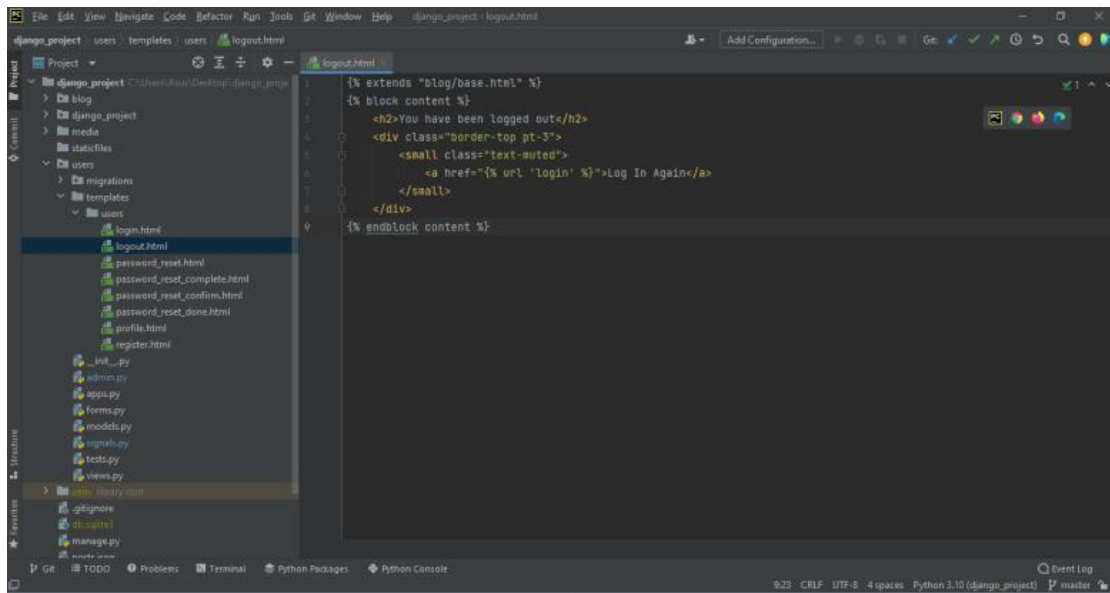
Users App Templates

➤ django_project > users > templates > login.html



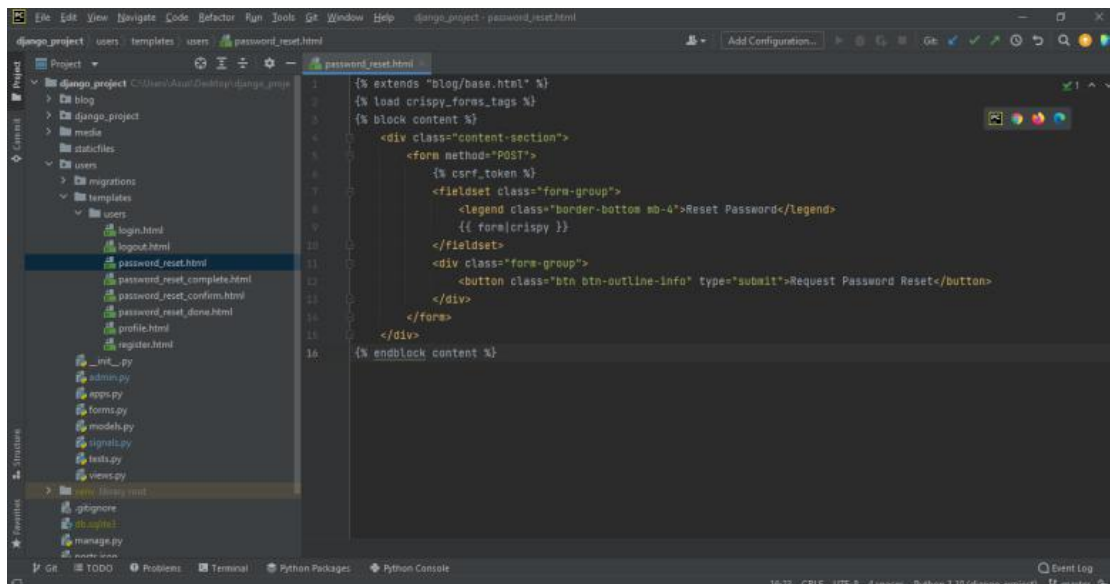
```
1 {% extends "blog/base.html" %}
2 {% load crispy_forms_tags %}
3 {% block content %}
4     <div class="content-section">
5         <form method="POST">
6             {% csrf_token %}
7             <fieldset class="form-group">
8                 <legend class="border-bottom mb-4">Log In</legend>
9                 {{ form|crispy }}
10            </fieldset>
11            <div class="form-group">
12                <button class="btn btn-outline-info" type="submit">Login</button>
13                <small class="text-muted ml-2">
14                    <a href="{% url 'password_reset' %}">Forgot Password?</a>
15                </small>
16            </div>
17        </form>
18        <div class="border-top pt-3">
19            <small class="text-muted">
20                Need An Account? <a class="ml-2" href="{% url 'register' %}">Sign Up Now</a>
21            </small>
22        </div>
23    </div>
24 {% endblock content %}
```

➤ **django_project > users > templates > users > logout.html**



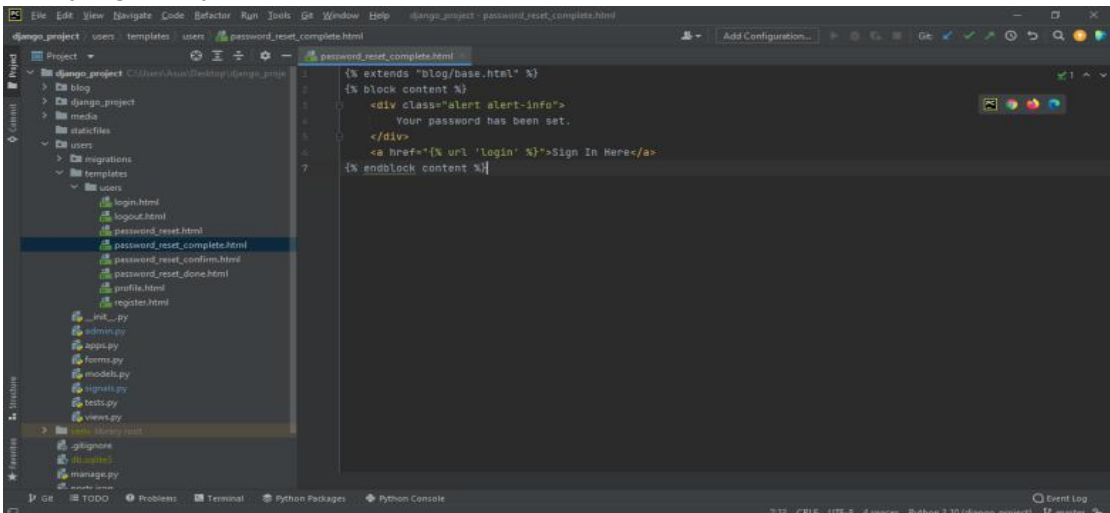
```
1 {% extends "blog/base.html" %}
2 {% block content %}
3     <h2>You have been logged out</h2>
4     <div class="border-top pt-3">
5         <small class="text-muted">
6             <a href="{% url 'login' %}">Log In Again</a>
7         </small>
8     </div>
9 {% endblock content %}
```

➤ **django_project > users > templates > users > password_reset.html**



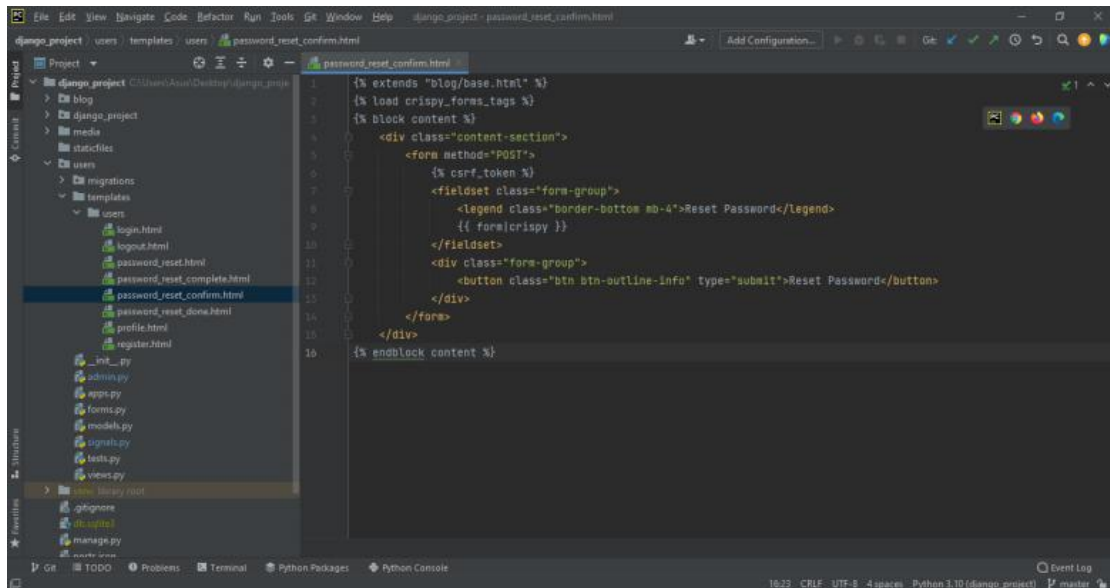
```
1 {% extends "blog/base.html" %}
2 {% load crispy_forms_tags %}
3 {% block content %}
4     <div class="content-section">
5         <form method="POST">
6             {% csrf_token %}
7             <fieldset class="form-group">
8                 <legend class="border-bottom mb-4">Reset Password</legend>
9                 {{ form|crispy }}
10            </fieldset>
11            <div class="form-group">
12                <button class="btn btn-outline-info" type="submit">Request Password Reset</button>
13            </div>
14        </form>
15    </div>
16{% endblock content %}
```

➤ **django_project > users > templates > users > password_reset_complete.html**



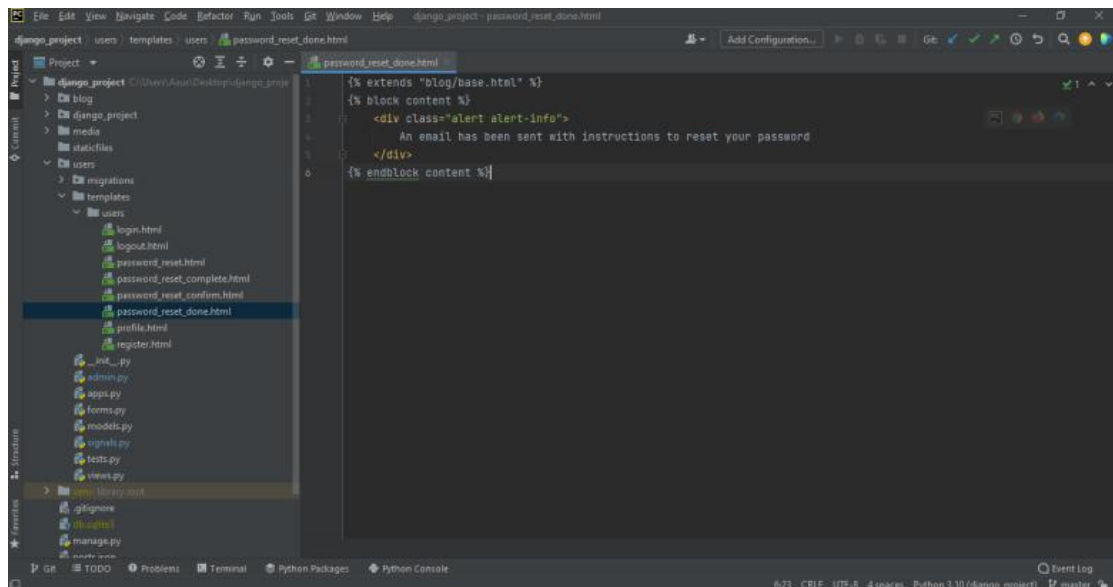
```
1 {% extends "blog/base.html" %}
2 {% block content %}
3     <div class="alert alert-info">
4         Your password has been set.
5     </div>
6     <a href="{% url 'login' %}">Sign In Here</a>
7 {% endblock content %}
```

- django_project > users > templates > users > password_reset_confirm.html



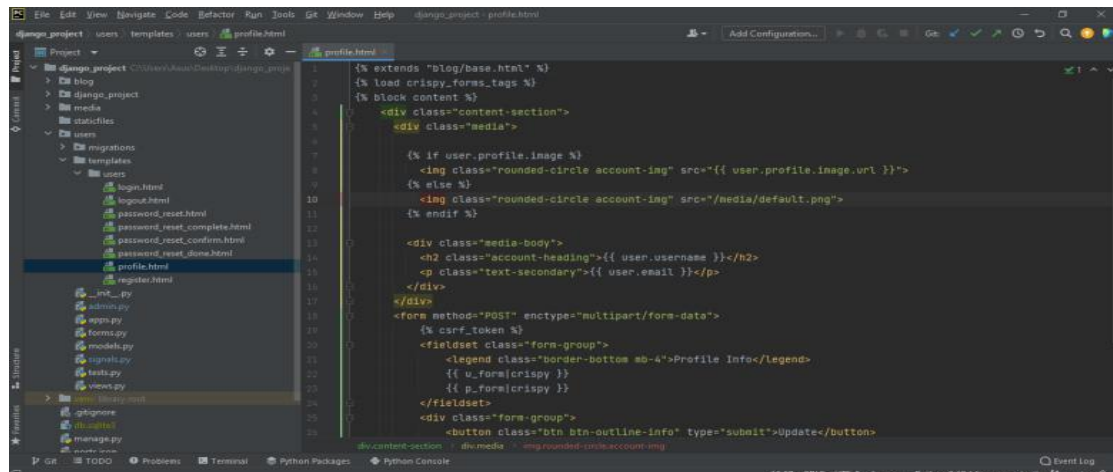
```
1 {% extends "blog/base.html" %}
2 {% load crispy_forms_tags %}
3 {% block content %}
4     <div class="content-section">
5         <form method="POST">
6             {% csrf_token %}
7             <fieldset class="form-group">
8                 <legend class="border-bottom mb-4">Reset Password</legend>
9                 {{ form|crispy }}
10            </fieldset>
11            <div class="form-group">
12                <button class="btn btn-outline-info" type="submit">Reset Password</button>
13            </div>
14        </form>
15    </div>
16 {% endblock content %}
```

- django_project > users > templates > users > password_reset_done.html



```
1 {% extends "blog/base.html" %}
2 {% block content %}
3     <div class="alert alert-info">
4         An email has been sent with instructions to reset your password
5     </div>
6 {% endblock content %}
```

- django_project > users > templates > users > profile.html



```
1 {% extends "blog/base.html" %}
2 {% load crispy_forms_tags %}
3 {% block content %}
4     <div class="content-section">
5         <div class="media">
6             {% if user.profile.image %}
7                 
8             {% else %}
9                 
10            {% endif %}
11            <div class="media-body">
12                <h2 class="account-heading">{{ user.username }}</h2>
13                <p class="text-secondary">{{ user.email }}</p>
14            </div>
15        </div>
16        <form method="POST" enctype="multipart/form-data">
17            {% csrf_token %}
18            <fieldset class="form-group">
19                <legend class="border-bottom mb-4">Profile Info</legend>
20                {{ u_form|crispy }}
21                {{ p_form|crispy }}
22            </fieldset>
23            <div class="form-group">
24                <button class="btn btn-outline-info" type="submit">Update</button>
25            </div>
26        </form>
27    </div>
28 {% endblock content %}
```

```

10 
11 {% endif %}
12
13 <div class="media-body">
14   <h2 class="account-heading">{{ user.username }}</h2>
15   <p class="text-secondary">{{ user.email }}</p>
16 </div>
17 </div>
18
19 <form method="POST" enctype="multipart/form-data">
20   {% csrf_token %}
21   <fieldset class="form-group">
22     <legend class="border-bottom mb-4">Profile Info</legend>
23     {{ u_form|crispy }}
24     {{ p_form|crispy }}
25   </fieldset>
26   <div class="form-group">
27     <button class="btn btn-outline-info" type="submit">Update</button>
28   </div>
29 </form>
30 {% endblock content %}

```

➤ **django_project > users > templates > users > register.html**

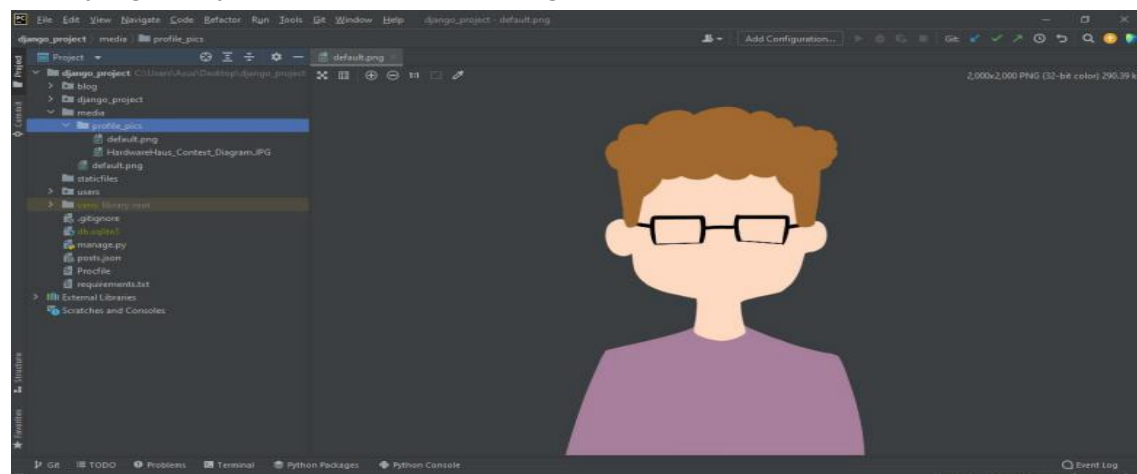
```

1 {% extends "blog/base.html" %}
2 {% load crispy_forms_tags %}
3 {% block content %}
4   <div class="content-section">
5     <form method="POST">
6       {% csrf_token %}
7       <fieldset class="form-group">
8         <legend class="border-bottom mb-4">Join Today</legend>
9         {{ form|crispy }}
10      </fieldset>
11      <div class="form-group">
12        <button class="btn btn-outline-info" type="submit">Sign Up</button>
13      </div>
14    </form>
15    <div class="border-top pt-3">
16      <small class="text-muted">
17        Already Have An Account? <a class="ml-3" href="{% url 'login' %}">Sign In</a>
18      </small>
19    </div>
20  </div>
21 {% endblock content %}

```

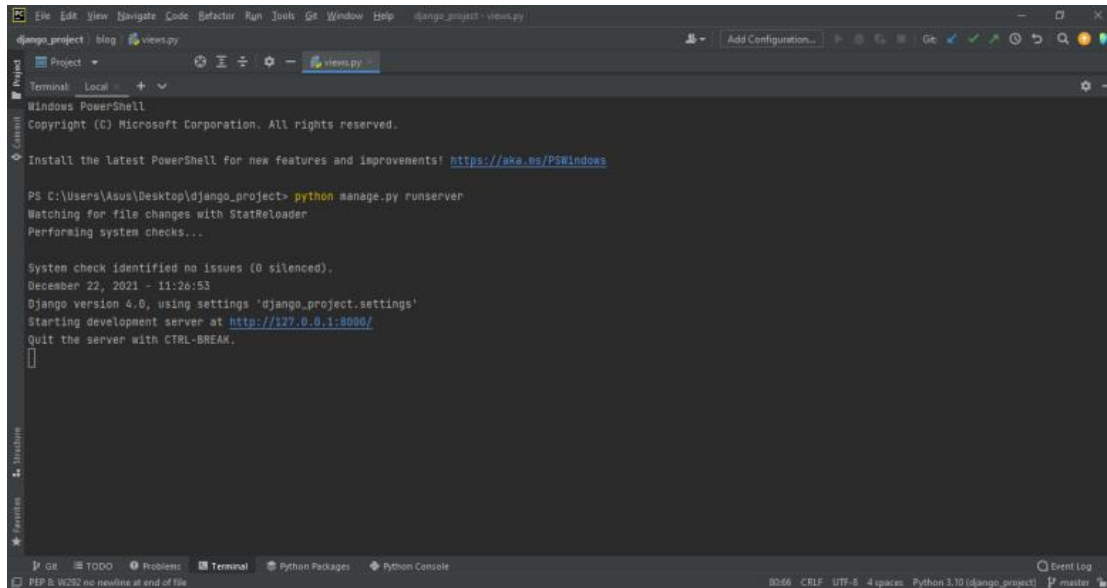
MEDIA FILES

➤ **django_project > media > default.png**



RUNSERVER COMMAND

- `python manage.py runserver`

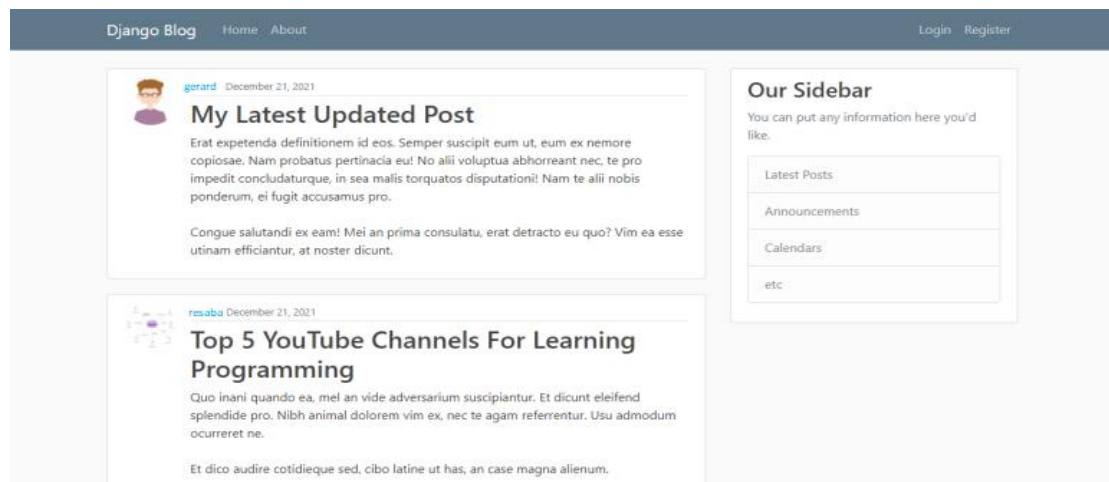


```
PS C:\Users\Asus\Desktop\django_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

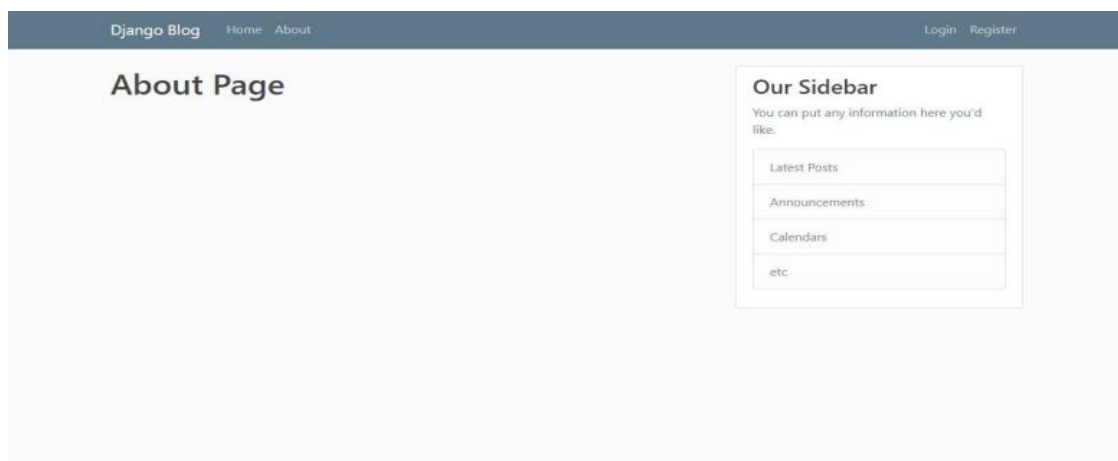
System check identified no issues (0 silenced).
December 22, 2021 - 11:26:53
Django version 4.0, using settings 'django_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[]
```

DJANGO BLOG OUTPUT

- **Home Page**



- **About Page**



➤ Register Page

Django Blog

Home

About

Login

Register

Join Today

Username*

Required: 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

etc

Django Blog

Home

About

Login

Register

Required: 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

Sign Up

Already Have An Account? [Sign In](#)

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

etc

➤ Login Page

Django Blog

Home

About

Login

Register

Log In

Username*

Password*

Login

Forgot Password?

Need An Account? [Sign Up Now](#)

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

etc

➤ Forgot Password Page

[Django Blog](#) [Home](#) [About](#) [Login](#) [Register](#)

Reset Password

Email*

Our Sidebar

You can put any information here you'd like.

Latest Posts


Announcements

Calendars

etc

➤ Log In User Home Page

[Django Blog](#) [Home](#) [About](#) [New Post](#) [Profile](#) [Logout](#)




[gerard](#) December 21, 2021

My Latest Updated Post

Erat expetenda definitionem id eos. Semper suscipit eum ut, eum ex nemore copiosae. Nam probatus pertinacia eu! No alii voluptua abhorreant nec, te pro impedit concludaturque, in sea malis torquatos disputationi! Nam te alii nobis ponderum, ei fugit accusamus pro.

Congue salutandi ex eam! Mei an prima consulatu, erat detracto eu quo? Vim ea esse utinam efficiantur, at noster dicunt.



[resaba](#) December 21, 2021

Top 5 YouTube Channels For Learning Programming

Quo inani quando ea, mel an vide adversarium suscipiantur. Et dicunt eleifend splendide pro. Nibh animal dolore vim ex, nec te agam referrentur. Usu admodum occurreret ne.

Et dico audire cotidieque sed, cibo latine ut has, an case magna alienum.

Our Sidebar

You can put any information here you'd like.

Latest Posts


Announcements

Calendars

etc

➤ Profile Page

[Django Blog](#) [Home](#) [About](#) [New Post](#) [Profile](#) [Logout](#)



[gerard](#)

Profile Info

Username*

Required. 150 characters or fewer. Letters, digits and @/+/+/-/_ only.

Email*

Image*

Currently: [profile_pics/HardwareHaus_Context_Diagram.JPG](#)
Change:
 No file chosen

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

etc

➤ New Post Page

[Django Blog](#) [Home](#) [About](#) [New Post](#) [Profile](#) [Logout](#)

Blog Post

Title*

Content*

Post

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements


Calendars

etc

➤ User Posts Page

[Django Blog](#) [Home](#) [About](#) [New Post](#) [Profile](#) [Logout](#)


Posts by gerard (15)

[gerard](#) December 21, 2021

My Latest Updated Post

Erat expetenda definitionem id eos. Semper suscipit eum ut, eum ex nemore copiosae. Nam probatus pertinacia eu! No alii voluptua abhorreant nec, te pro impedit concludaturque, in sea malis torquatos disputationi! Nam te alii nobis ponderum, ei fugit accusamus pro.

Congue salutandi ex eam! Mei an prima consulatu, erat detracto eu quo? Vim ea esse utinam efficiantur, at noster dicunt.

[resaba](#) December 21, 2021

Top 10 Python Tips and Tricks

Pro minim iuvaret ad. No nam ornatus principes euripidis, at sale vituperatoribus eos, eros regione scripserit id mea. Has ne inermis nostrum, quo tantas melius dissentias at! Ut vim tibi que omnesque. An mel modo ponderum, eum at probo appetere imperdiet? Natum quaeque intellegebat per ex. Cu viris clita sit?

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

etc

➤ Logout Page

[Django Blog](#) [Home](#) [About](#) [Login](#) [Register](#)

You have been logged out

[Log In Again](#)

Our Sidebar

You can put any information here you'd like.


Latest Posts

Announcements

Calendars

etc

➤ Admin Login Page



The image shows the Django administration login page. It features a central white box with a blue header that says "Django administration". Below the header, there are two input fields: "Username:" and "Password:". A blue "Log in" button is positioned below the password field. The entire login box is centered on a light gray background.

➤ Admin Home Page (Logged In)



The image shows the Django administration home page after logging in. The page has a dark blue header with the text "Django administration" on the left and "WELCOME, JANO. VIEW SITE / CHANGE PASSWORD / LOG OUT" on the right. Below the header, the page is divided into two main sections. On the left, under the heading "Site administration", there are three categories: "AUTHENTICATION AND AUTHORIZATION" (containing "Groups" and "Users"), "BLOG" (containing "Posts"), and "USERS" (containing "Profiles"). Each item has a green plus icon for "Add" and a yellow pencil icon for "Change". On the right, under the heading "Recent actions", there is a section titled "My actions" which lists three items: "john profile" (Profile), "ana user" (User), and "john user" (User). Each item has a green plus icon.