

Full Stack Development using Python

Module 5: Django Web Framework – Part1, 2 & 3

(Model ,Views,Templates,Static CSS)

Django Web Application Fundamentals Exercise

This exercise will guide you through setting up a basic Django project, creating an app, defining a simple model, and displaying data using views and templates.

Objective: Create a Django application that displays a list of "Tasks" on a web page.

Part 1: Project and App Setup

1. Create a Project Folder and Virtual Environment:

- Open your terminal or command prompt.
 - Create a new directory for your project (e.g., my_todo_app) and navigate into it.

✓ `mkdir my_todo_app`

✓ `cd my_todo_app`

```
D:\College_Demos>mkdir my_todo_app
```

```
D:\College_Demos>cd my_todo_app
```

- Create and activate a virtual environment named venv within this folder.

✓ `python -m venv venv`

```
D:\College_Demos\my_todo_app>python -m venv venv
```

On Windows

✓ `venv\Scripts\activate`

```
D:\College_Demos\my_todo_app>venv\Scripts\activate
```

- Install Django in your active virtual environment.

✓ `pip install Django`

```
(venv) D:\College_Demos\my_todo_app>pip install Django
```

2. Create a Django Project:

- Inside my_todo_app, create a new Django project named todo_project. Ensure the project is created in the current directory (use a dot . at the end of the command).

✓ `django-admin startproject todo_project .`

```
(venv) D:\College_Demos\my_todo_app>django-admin startproject todo_project .
```

Create a Django App:

Create a new Django app within your todo_project named tasks.

✓ `python manage.py startapp tasks`

```
(venv) D:\College_Demos\my_todo_app>python manage.py startapp tasks
```

3. Register the App:

- Open todo_project/settings.py and add 'tasks' to your INSTALLED_APPS list.

```
'DIRS': [os.path.join(BASE_DIR, 'templates')], # Modify this line
```

- **Open todo_project/settings.py** in your text editor.
- **Find the TEMPLATES setting.** It will look something like this:

```
TEMPLATES = [
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [], # This is the line you need to modify
    'APP_DIRS': True,
    'OPTIONS': {
        # ...
    },
},
]
```

Modify the DIRS list to include the path to your templates directory.

You should use os.path.join and BASE_DIR to make the path platform-independent and robust.

First, ensure you have import os at the top of your settings.py if it's not already there.

Then, change the DIRS line to:

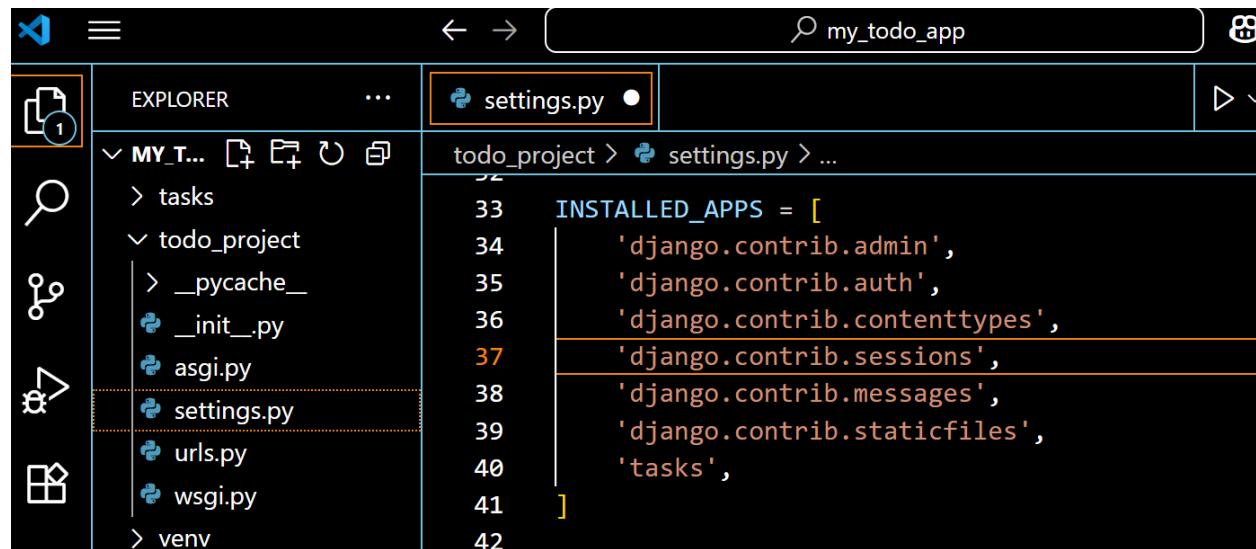
import os # Make sure this line is at the top of settings.py

```
# ... other settings ...
TEMPLATES = [
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [os.path.join(BASE_DIR, 'templates')], # Modify this line
    'APP_DIRS': True,
    'OPTIONS': {
        # ...
    },
},
]
```

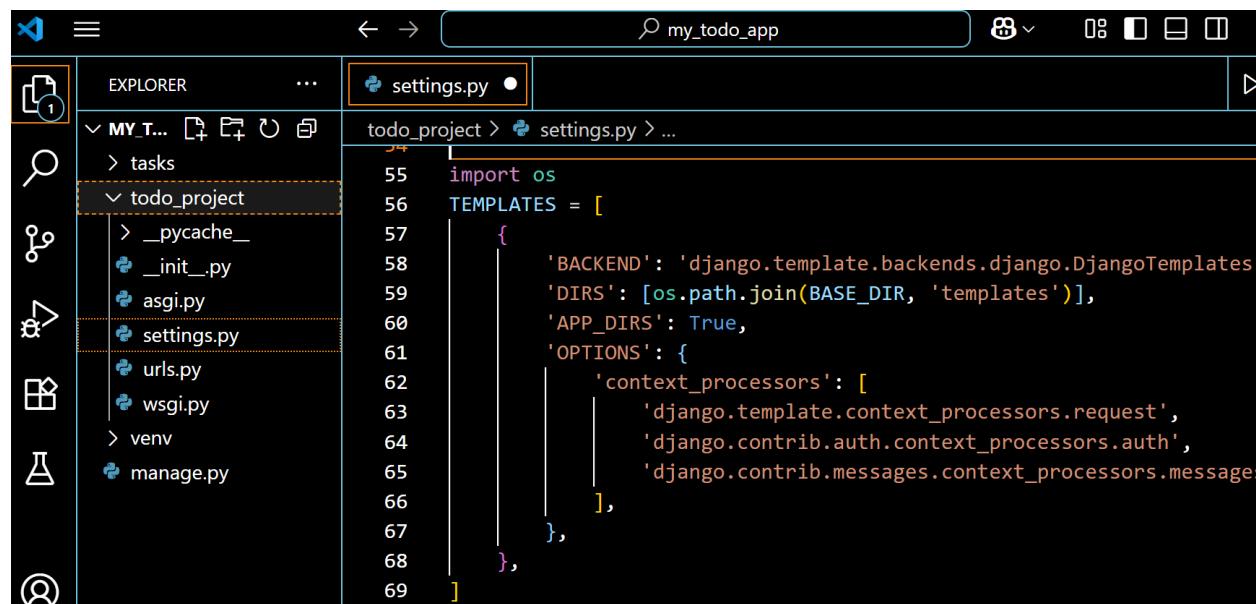
```
# ... rest of your settings ...
```

Open the VSCode Editor:

```
(venv) D:\College_Demos\my_todo_app>code .
```



```
EXPLORER      ...    settings.py ●
todo_project > settings.py > ...
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'tasks',
41 ]
```



```
EXPLORER      ...    settings.py ●
todo_project > settings.py > ...
55 import os
56 TEMPLATES = [
57     {
58         'BACKEND': 'django.template.backends.django.DjangoTemplates',
59         'DIRS': [os.path.join(BASE_DIR, 'templates')],
60         'APP_DIRS': True,
61         'OPTIONS': {
62             'context_processors': [
63                 'django.template.context_processors.request',
64                 'django.contrib.auth.context_processors.auth',
65                 'django.contrib.messages.context_processors.message',
66             ],
67         },
68     },
69 ]
```

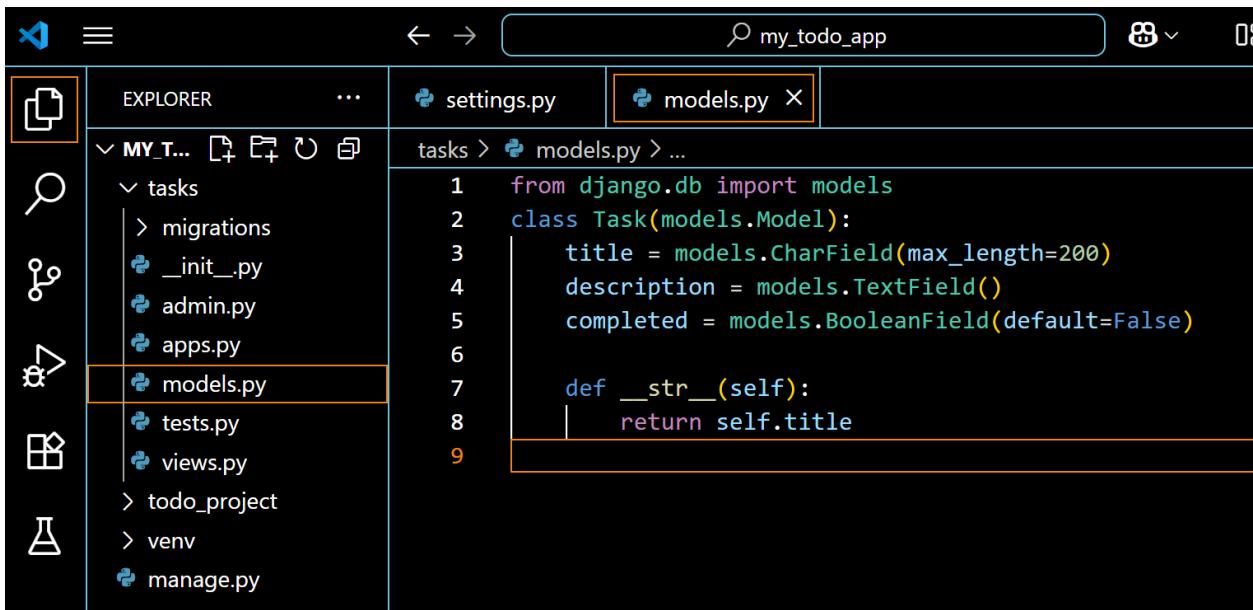
Part 2: Define a Model

1. Define the Task Model:

- Open tasks/models.py.
- Create a simple Django model named Task with the following fields:
 - title (CharField, max_length=200)
 - description (TextField)
 - completed (BooleanField, default=False)
- Add a __str__ method to return the task's title.

```
# tasks/models.py
from django.db import models
class Task(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    completed = models.BooleanField(default=False)

    def __str__(self):
        return self.title
```



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

- File Explorer:** On the left, it shows the project structure under "MY_T...". The "tasks" folder is expanded, showing "migrations", "__init__.py", "admin.py", "apps.py", and "models.py". "models.py" is currently selected and highlighted with an orange border.
- Code Editor:** The main pane displays the contents of "models.py". The code is as follows:

```
1  from django.db import models
2  class Task(models.Model):
3      title = models.CharField(max_length=200)
4      description = models.TextField()
5      completed = models.BooleanField(default=False)
6
7      def __str__(self):
8          return self.title
9
```

2. Create and Apply Migrations:

- In your terminal (from the todo_project directory), run the following commands to create and apply migrations for your new Task model.

✓ *python manage.py makemigrations tasks*
✓ *python manage.py migrate*

```
(venv) D:\College_Demos\my_todo_app>python manage.py makemigrations tasks
```

```
(venv) D:\College_Demos\my_todo_app>python manage.py migrate
```

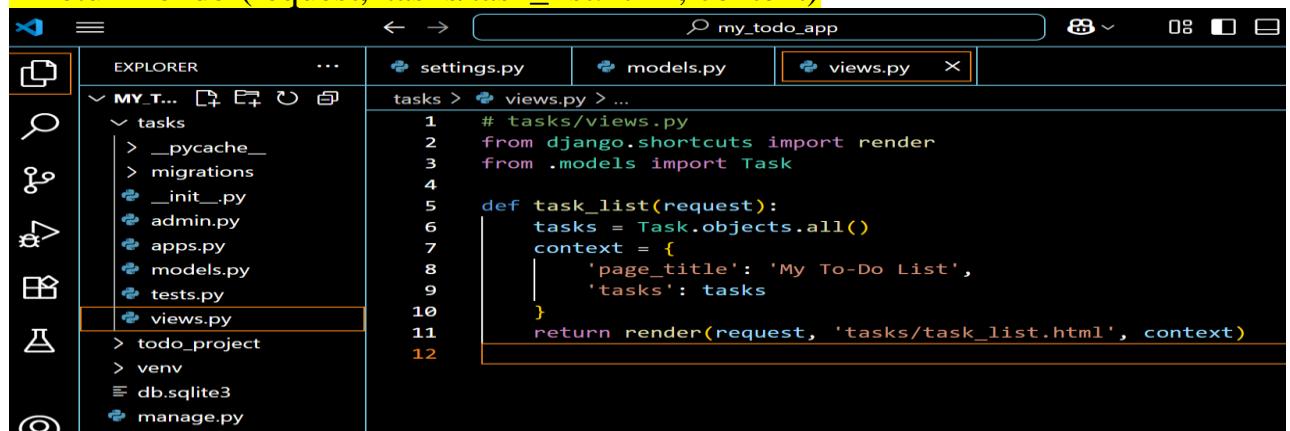
Part 3: Create a View and URLs

1. Create a Task List View:

- Open tasks/views.py.
- Create a function-based view task_list that fetches all Task objects from the database and passes them to a template.

```
# tasks/views.py
from django.shortcuts import render
from .models import Task
```

```
def task_list(request):
    tasks = Task.objects.all()
    context = {
        'page_title': 'My To-Do List',
        'tasks': tasks
    }
    return render(request, 'tasks/task_list.html', context)
```

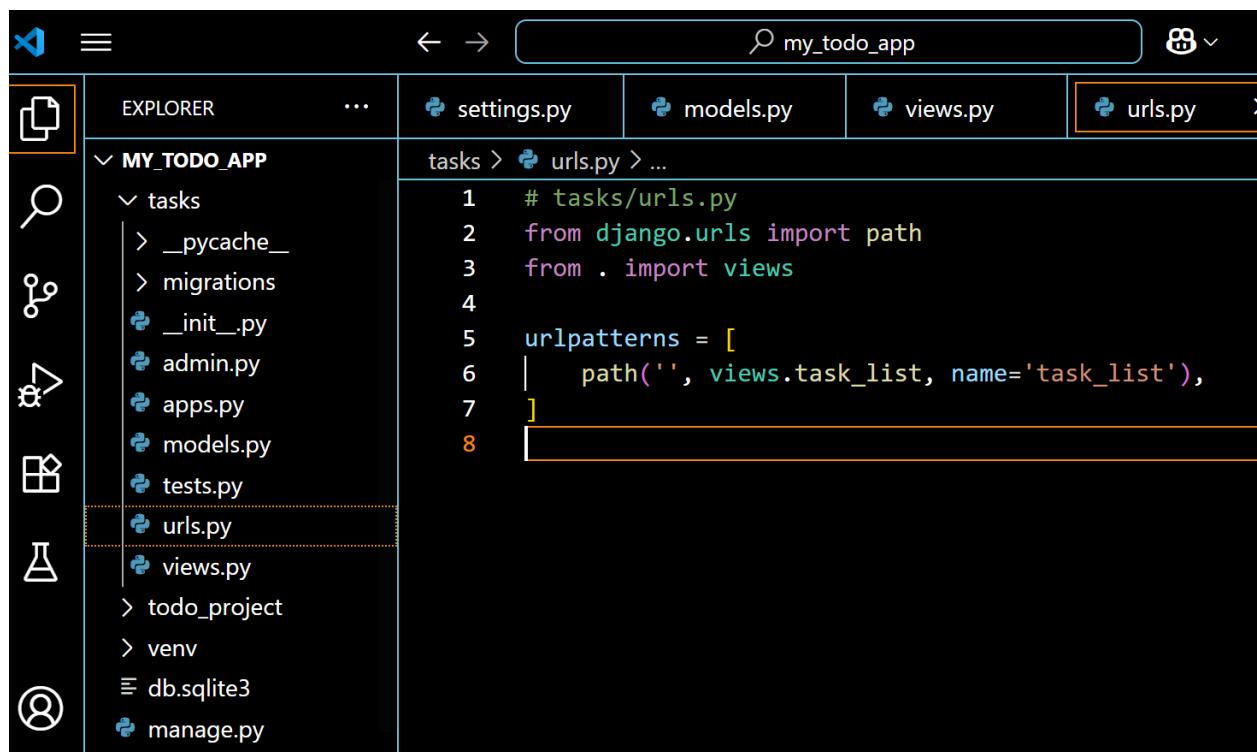


2. Define App URLs:

- Inside the tasks directory, **create a new file named urls.py**.
- Define a URL pattern that maps the root path ("") to your task_list view.

```
# tasks/urls.py
from django.urls import path
from . import views
```

```
urlpatterns = [
    path("", views.task_list, name='task_list'),
]
```



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

- File Explorer:** On the left, it shows the project structure under "MY_TODO_APP". The "tasks" folder is expanded, showing files like __init__.py, admin.py, apps.py, models.py, tests.py, and urls.py. "urls.py" is currently selected and highlighted with a dotted border.
- Editor:** The main area displays the content of "urls.py". The code is as follows:

```
# tasks/urls.py
from django.urls import path
from . import views

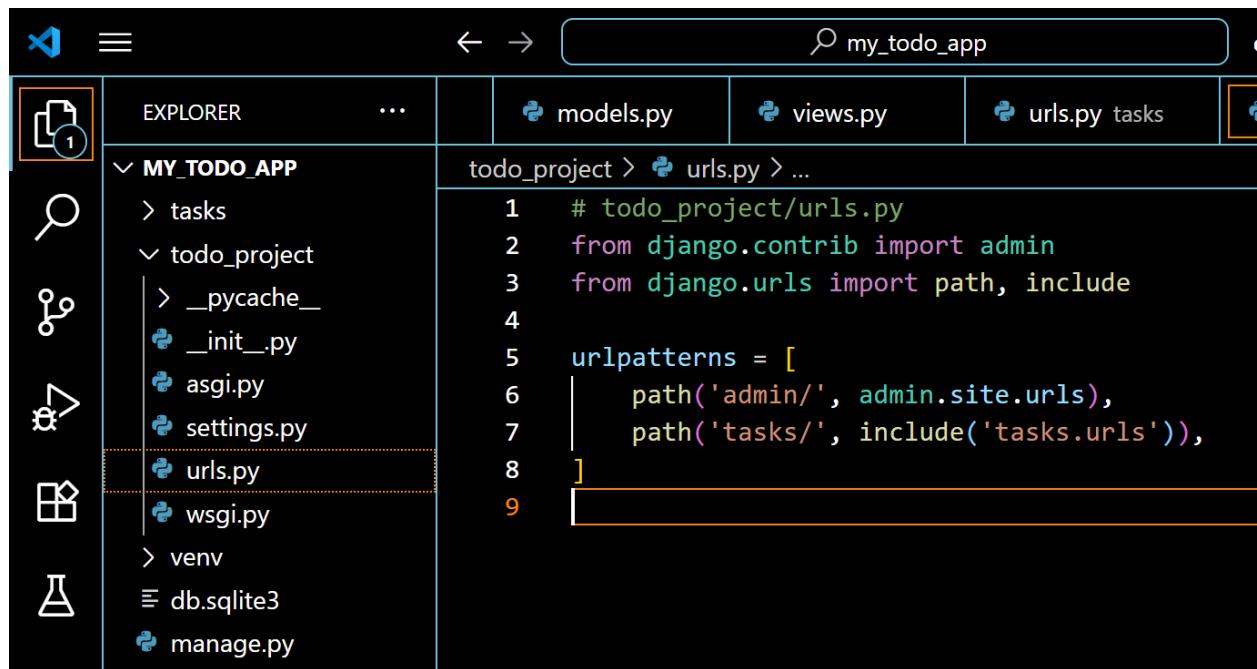
urlpatterns = [
    path("", views.task_list, name='task_list'),
]
```

3. Include App URLs in Project URLs:

- Open todo_project/urls.py (the project-level urls.py).
- Import include and add a path that delegates to your tasks.urls. For example, all URLs under /tasks/ will be handled by your tasks app.

```
# todo_project/urls.py
from django.contrib import admin
from django.urls import path, include
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('tasks/', include('tasks.urls')),
]
```



```
todo_project > urls.py > ...
1 # todo_project/urls.py
2 from django.contrib import admin
3 from django.urls import path, include
4
5 urlpatterns = [
6     path('admin/', admin.site.urls),
7     path('tasks/', include('tasks.urls')),
8 ]
9 
```

Part 4: Create a Template

1. Create Templates Directory:

- In your project root (todo_project), create a new folder named templates.
- Inside templates, create another folder named tasks.

```
(venv) D:\College_Demos\my_todo_app>mkdir templates
```

```
(venv) D:\College_Demos\my_todo_app>mkdir templates\tasks
```

2. Create Task List Template:

- Inside todo_project/templates/tasks/, create a new file named task_list.html.
- Add basic HTML structure and use Django Template Language (DTL) to display the page_title and iterate over the tasks list. For each task, display its title, description, and completed status.

#task_list.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ page_title }}</title>
</head>
<body>
    <h1>{{ page_title }}</h1>
    {% if tasks %}
        <ul>
            {% for task in tasks %}
                <li>
                    <h3>{{ task.title }}</h3>
                    <p>{{ task.description }}</p>
                    {% if task.completed %}
                        <p>Status: Completed</p>
                    {% else %}
```

```

<p>Status: Pending</p>
    {% endif %}
</li>
    {% endfor %}
</ul>
    {% else %}
        <p>No tasks added yet.</p>
    {% endif %}
</body>
</html>

```

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

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** my_todo_app
- Explorer Sidebar:**
 - MY_TODO_APP
 - > tasks
 - ✓ templates\tasks
 - task_list.html
 - > todo_project
 - > __pycache__
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
 - > venv
 - db.sqlite3
 - manage.py
- Code Editor Area:**

```

templates > tasks > task_list.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>{{ page_title }}</title>
7  </head>
8  <body>
9      <h1>{{ page_title }}</h1>
10     {% if tasks %}
11         <ul>
12             {% for task in tasks %}
13                 <li>
14                     <h3>{{ task.title }}</h3>
15                     <p>{{ task.description }}</p>
16                     {% if task.completed %}
17                         <p>Status: Completed</p>
18                     {% else %}
19                         <p>Status: Pending</p>
20                     {% endif %}
21                 </li>
22             {% endfor %}
23         </ul>
24     {% else %}
25         <p>No tasks added yet.</p>
26     {% endif %}
27 </body>
28 </html>

```

Part 5: Run the Development Server

1. Run the Server:

- In your terminal (from the todo_project directory), run the Django development server.

✓ **python manage.py runserver**

```
(venv) D:\College_Demos\my_todo_app>python manage.py runserver
```

2. View Your Application:

- Open your web browser and go to <http://127.0.0.1:8000/tasks/>.
- You should see your "My To-Do List" page.



Bonus (Optional): Add Data via Admin Panel

1. Create a Superuser:

- Create an administrative user for your Django project.

✓ **python manage.py createsuperuser**

```
(venv) D:\College_Demos\my_todo_app>python manage.py createsuperuser
Username (leave blank to use 'dell'): shubhali
Email address: shubhali11@gmail.com
Password:
Password (again):
Superuser created successfully.
```

- Follow the prompts to set up a username, email (optional), and password.

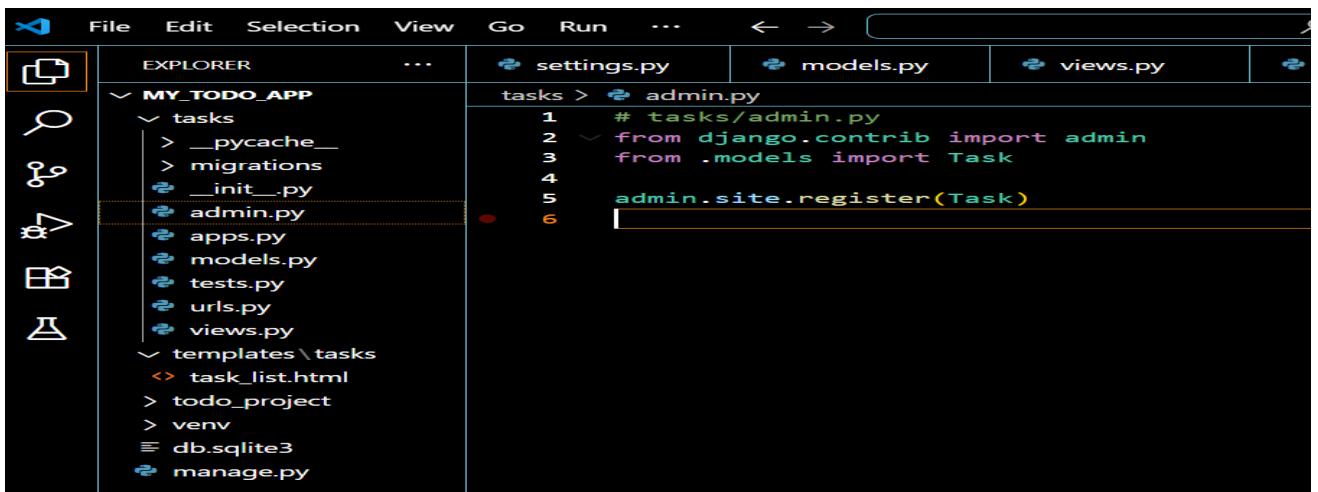
2. Register Model in Admin:

- o Open tasks/admin.py and register your Task model so you can manage tasks through the Django admin interface.

```
# tasks/admin.py
```

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

```
admin.site.register(Task)
```



The screenshot shows a code editor interface with a dark theme. On the left is an Explorer sidebar showing the project structure of 'MY_TODO_APP' with files like settings.py, models.py, views.py, admin.py, apps.py, migrations, __init__.py, and various templates and URLs. The main editor area displays the contents of 'tasks/admin.py'. The registration code is highlighted in yellow:

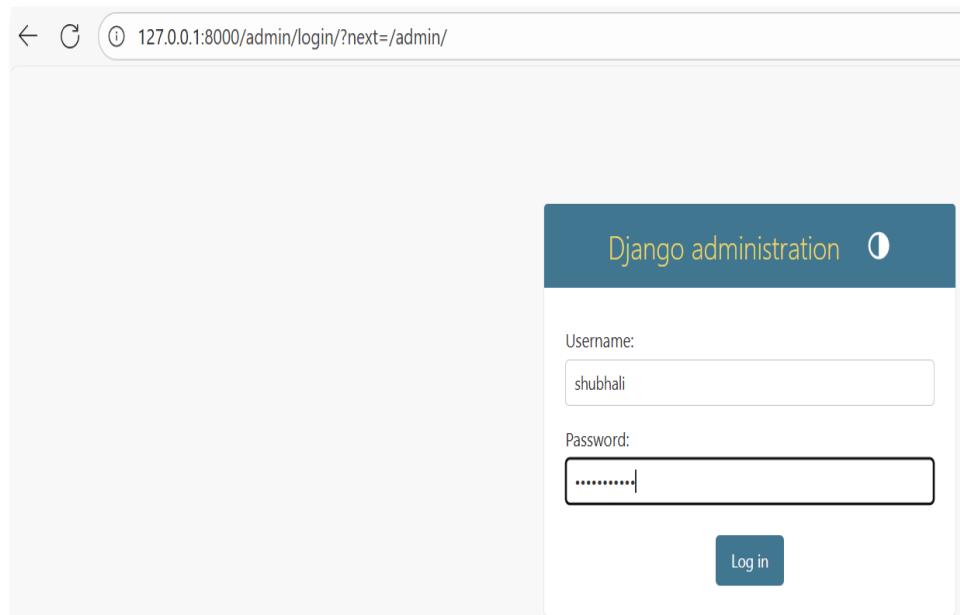
```
# tasks/admin.py  
from django.contrib import admin  
from .models import Task  
  
admin.site.register(Task)
```

3. Access Admin Panel:

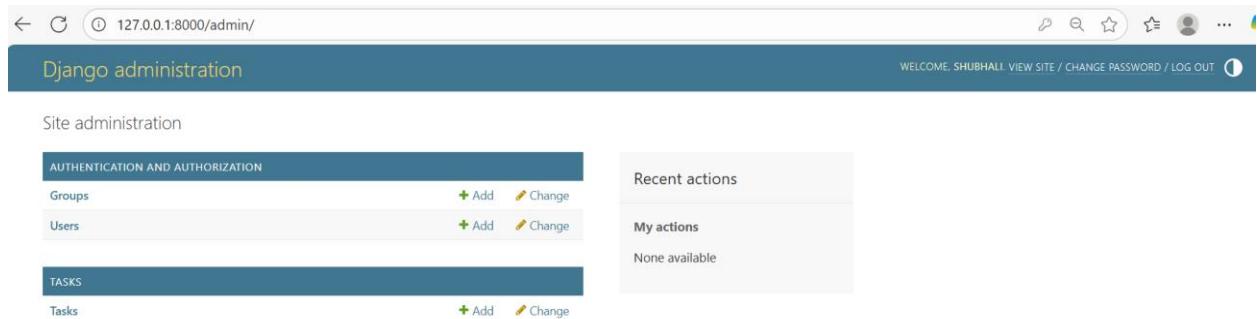
1. Ensure your development server is running (python manage.py runserver).

```
(venv) D:\College_Demos\my_todo_app>python manage.py runserver
```

- Open <http://127.0.0.1:8000/admin/> in your browser. Log in with the superuser credentials you just created.



- You should now see "Tasks" under your "Tasks" app. Click "Add" next to "Tasks" to create new tasks. These tasks will then appear on your <http://127.0.0.1:8000/tasks/> page.



Django administration

Home > Tasks > Tasks > Add task

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add
Users	+ Add

TASKS	
Tasks	+ Add

«

Add task

Title:

Description:

Completed

[SAVE](#) [Save and add another](#) [Save and continue editing](#)

Django administration

WELCOME, SHUBHALI. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Tasks > Tasks

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add
Users	+ Add

TASKS	
Tasks	+ Add

The task "Complete Assignments" was added successfully.

Select task to change [ADD TASK +](#)

Action: Go 0 of 1 selected

<input type="checkbox"/> TASK
<input type="checkbox"/> Complete Assignments

1 task

← ⌍ ⓘ 127.0.0.1:8000/tasks/

My To-Do List

- **Complete Assignments**

Complete Assignments on Html.
Status: Pending

Part 6: Add Static CSS

The goal here is to create a basic CSS file and apply some styles to your task_list.html page.

1. Create Static Files Directories:

- In your project's root directory (my_todo_app, where manage.py is located), create a new folder named static.
- Inside the static folder, create another folder named css.

```
(venv) D:\College_Demos\my_todo_app>mkdir static
```

```
(venv) D:\College_Demos\my_todo_app>mkdir static\css
```

Your directory structure should now look something like this:

```
my_todo_app/
    todo_project/
        __init__.py
        asgi.py
        settings.py <-- You will edit this
        urls.py     <-- You will edit this
        wsgi.py
    tasks/
        migrations/
        __init__.py
        admin.py    <-- You might have edited this
        apps.py
        models.py   <-- You edited this
        tests.py
        urls.py    <-- You created this
        views.py   <-- You edited this
    templates/
        tasks/
            task_list.html <-- You will edit this
        static/      <-- NEW FOLDER
            css/       <-- NEW FOLDER
                style.css <-- NEW FILE
    manage.py
    venv/
```

2. Create your CSS File:

- o Inside my_todo_app/static/css/, create a new file named style.css.
- o Add some basic CSS rules to it. For example:

```
/* my_todo_app/static/css/style.css */  
body {  
    font-family: Arial, sans-serif;  
    line-height: 1.6;  
    margin: 20px;  
    background-color: #f4f4f4;  
    color: #333;  
}  
h1 {  
    color: #336699;  
    border-bottom: 2px solid #336699;  
    padding-bottom: 10px;  
}  
ul {  
    list-style: none;  
    padding: 0;  
}  
li {  
    background-color: #fff;  
    border: 1px solid #ddd;  
    margin-bottom: 10px;  
    padding: 15px;  
    border-radius: 5px;  
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}  
h3 {  
    color: #555;  
    margin-top: 0;  
}  
  
p {  
    margin-bottom: 5px;  
}
```

```
1  body {  
2      font-family: Arial, sans-serif;  
3      line-height: 1.6;  
4      margin: 20px;  
5      background-color: #f4f4f4;  
6      color: #333;  
7  }  
8  h1 {  
9      color: #336699;  
10     border-bottom: 2px solid #336699;  
11     padding-bottom: 10px;  
12  }  
13  ul {  
14      list-style: none;  
15      padding: 0;  
16  }  
17  li {  
18      background-color: #fff;  
19      border: 1px solid #ddd;  
20      margin-bottom: 10px;  
21      padding: 15px;  
22      border-radius: 5px;  
23      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
24  }  
25  h3 {  
26      color: #555;  
27      margin-top: 0;  
28  }  
29  p {  
30      margin-bottom: 5px;  
31  }  
32
```

3. Configure settings.py for Static Files:

- Open todo_project/settings.py.
- You should already have STATIC_URL = 'static/' defined (it's usually there by default).
- Below STATIC_URL, add STATICFILES_DIRS. This list tells Django where to look for static files outside of individual app static folders.

```
# todo_project/settings.py
```

```
import os # Make sure this is at the top of your file
```

```
# ... (other settings) ...
```

```
# Static files (CSS, JavaScript, Images)
```

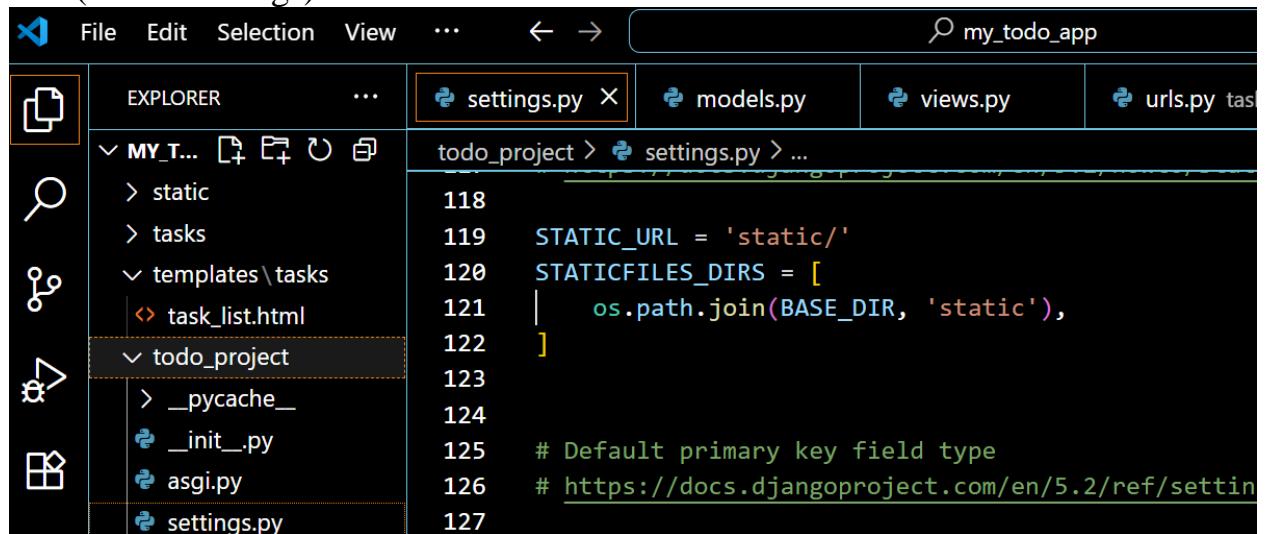
```
# https://docs.djangoproject.com/en/5.0/howto/static-files/
```

```
STATIC_URL = 'static/'
```

```
# Add this line to tell Django where your project-wide static files are
```

```
STATICFILES_DIRS = [  
    os.path.join(BASE_DIR, 'static'),  
]
```

```
# ... (rest of settings) ...
```



```
File Edit Selection View ... ← → ⌂ my_todo_app  
EXPLORER ... settings.py X models.py views.py urls.py tasks  
MY_T... D E U ⌂ todo_project > settings.py > ...  
> static  
> tasks  
< templates\tasks  
< task_list.html  
< todo_project  
> _pycache_  
__init__.py  
asgi.py  
settings.py  
118  
119     STATIC_URL = 'static/'  
120     STATICFILES_DIRS = [  
121         os.path.join(BASE_DIR, 'static'),  
122     ]  
123  
124  
125     # Default primary key field type  
126     # https://docs.djangoproject.com/en/5.2/ref/setting
```

4. Link the CSS in your Template:

- Open todo_project/templates/tasks/task_list.html.
- At the very top of the file, add { % load static % }. This tag makes the static template tag available.
- In the <head> section of your HTML, add a <link> tag to reference your style.css file using the { % static % } template tag.

```
#task_list.html
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>{{ page_title }}</title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body>
    <h1>{{ page_title }}</h1>
    {% if tasks %}
        <ul>
            {% for task in tasks %}
                <li>
                    <h3>{{ task.title }}</h3>
                    <p>{{ task.description }}</p>
                    {% if task.completed %}
                        <p>Status: Completed</p>
                    {% else %}
                        <p>Status: Pending</p>
                    {% endif %}
                </li>
            {% endfor %}
        </ul>
    {% else %}
        <p>No tasks added yet.</p>
    {% endif %}
</body>
</html>
```

5. Run the Development Server and Verify:

- Ensure your development server is running:

```
python manage.py runserver
```

```
(venv) D:\College_Demos\my_todo_app>python manage.py runserver
```

- Open your browser and navigate to <http://127.0.0.1:8000/tasks/>.

