Week 1 – Session 2 -Message board application

Create a directory where we will keep our project. Then open this directory in VS Code

In the VS code terminal type following commands.

Virtual environment setup:

Create virtual environment:

On Windows:

- python -m venv env

On macOS:

- python3 -m venv env

Activate virtual environment:

On Windows:

- env\Scripts\Activate.ps1

On macOS:

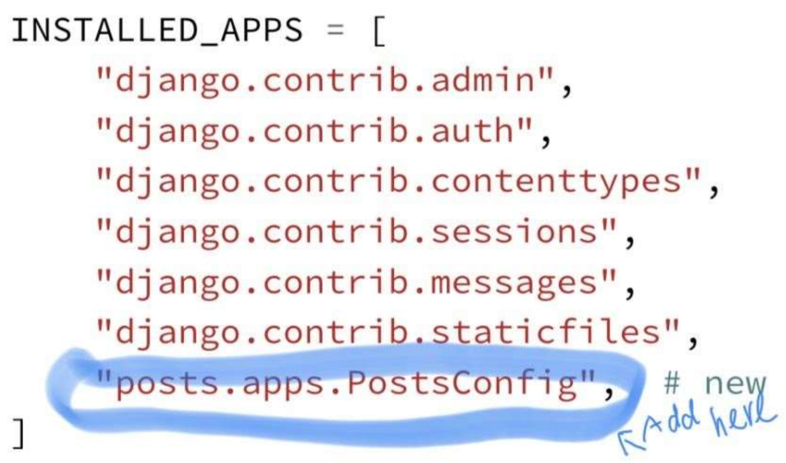
- source env/bin/activate

Project initial setup

In the same terminal, type following commands one by one

- pip install Django
- django-admin startproject messageBoardProject .
- python manage.py startapp posts

Register your 'posts' app in the project's '**settings.py'** file:



```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "posts.apps.PostsConfig",  # new
]
```

- python manage.py migrate

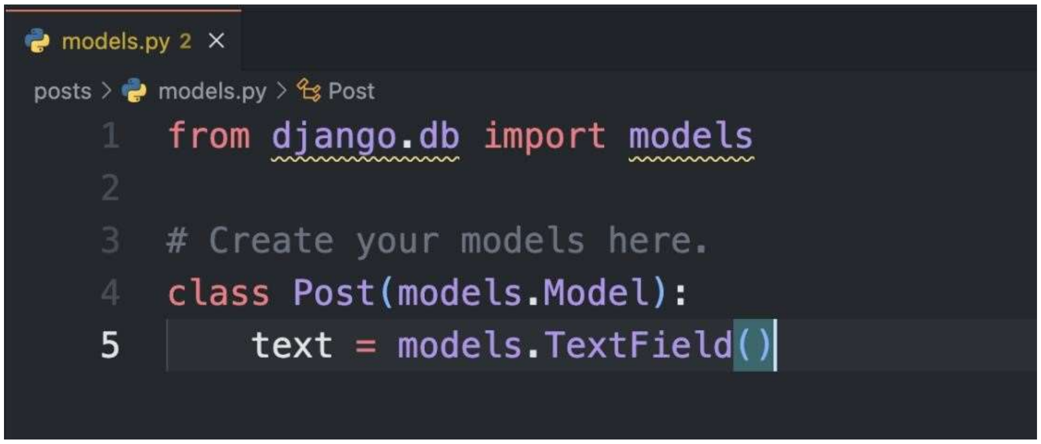
- python manage.py runserver

This will run your project. Now let's make some changes.

First of all, we will create our own model to store information.

- create your first model (https://docs.djangoproject.com/en/4.1/topics/db/models/)
  You may want to know what model fields (https://docs.djangoproject.com/en/4.1/ref/models/fields/) you can use in Django

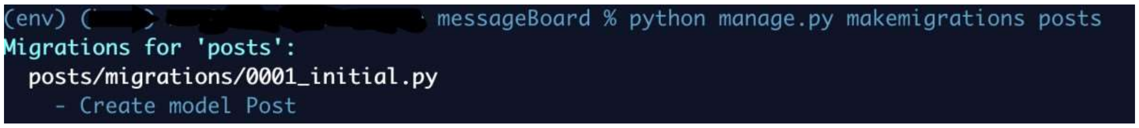
  In "posts/models.py" file, add below code: It defines a Post class with only one text field



```
models.py 2
posts > models.py > Post
from django.db import models

# Create your models here.
class Post(models.Model):
    text = models.TextField()
```

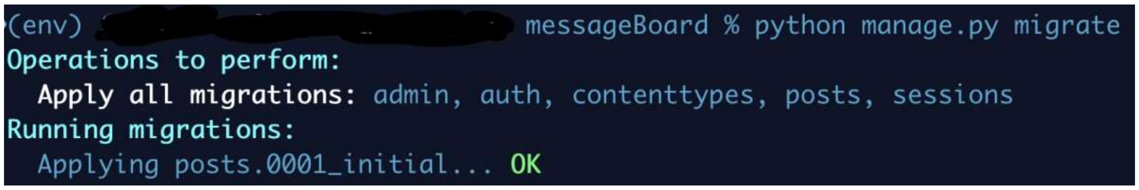
Now we need to tell our ORM(Object Relational Mapper) to make changes in database to add our model. We do that with two commands: **makemigrations** and **migrate**

- Run following commands. To create migrations use makemigrations for our application
  - python manage.py makemigrations posts



```
(env) [redacted] messageBoard % python manage.py makemigrations posts
Migrations for 'posts':
  posts/migrations/0001_initial.py
    - Create model Post
```

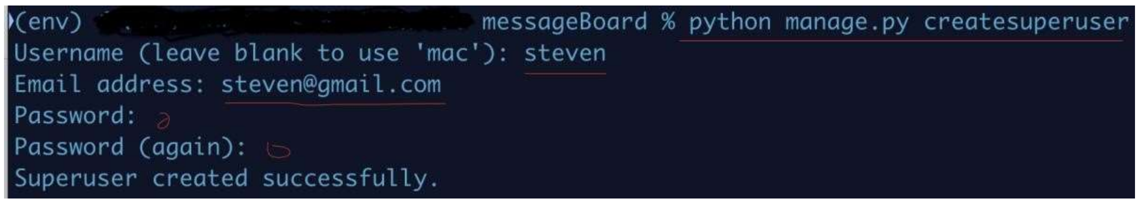
  Migrate command will run or apply those migrations to the database
  - python manage.py migrate



```
(env) [redacted] messageBoard % python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, posts, sessions
Running migrations:
  Applying posts.0001_initial... OK
```

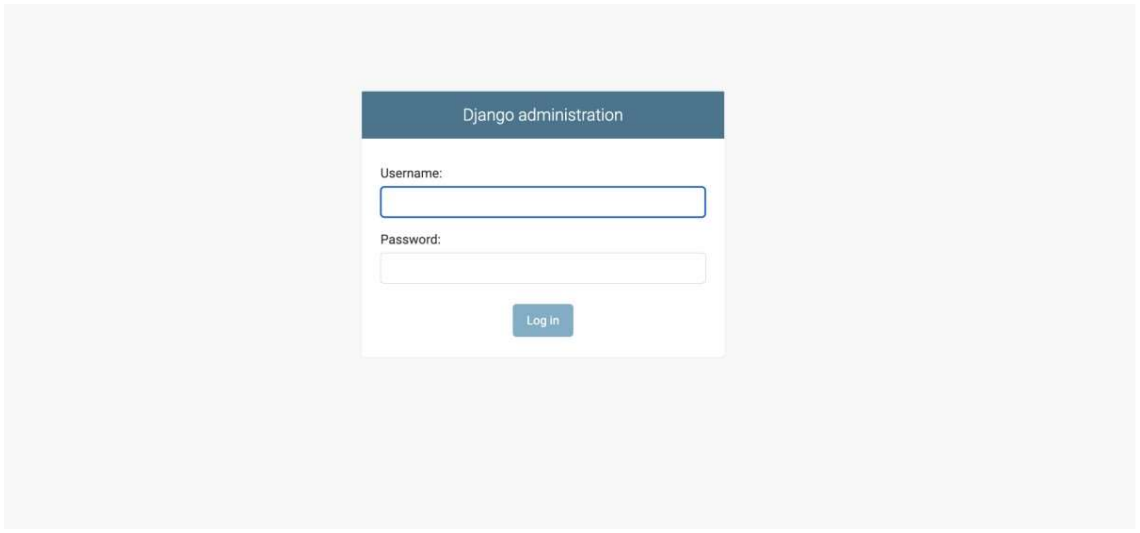
Django’s has a killer feature – robust admin interface which provides a visual way to interact with data. Before we can access it, we need a superuser to login to admin

1) Create a superuser by using the command: python manage.py createsuperuser

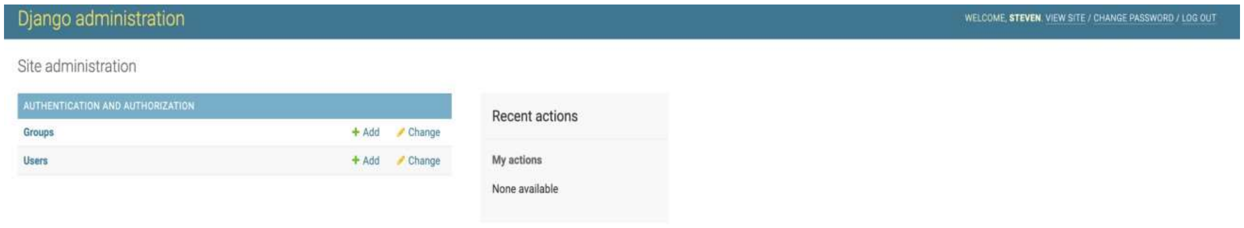


```
(env) [redacted] messageBoard % python manage.py createsuperuser
Username (leave blank to use 'mac'): steven
Email address: steven@gmail.com
Password:
Password (again):
Superuser created successfully.
```

2) Launch the server again if it is not already running by using the command: python manage.py runserver
3) Enter below url in your web browsers such as Chrome, you see: http://127.0.0.1:8000/admin/



4) Login by using superuser name and password you created before, you will see:



Django administration

WELCOME, **STEVEN**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

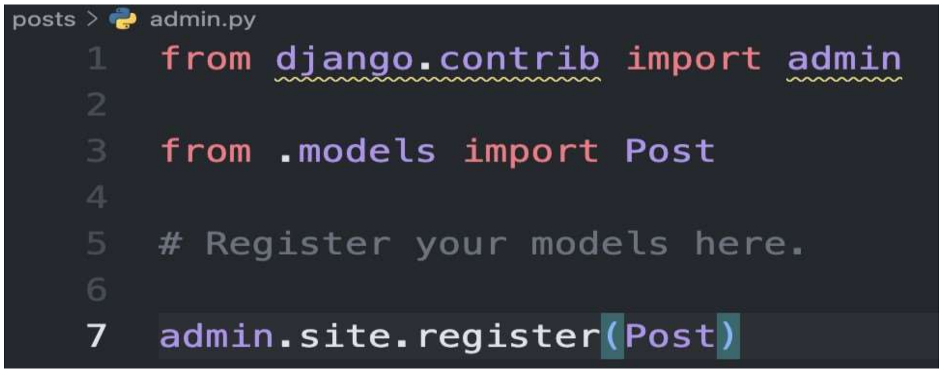
Groups	+ Add	Change
Users	+ Add	Change

Recent actions

My actions

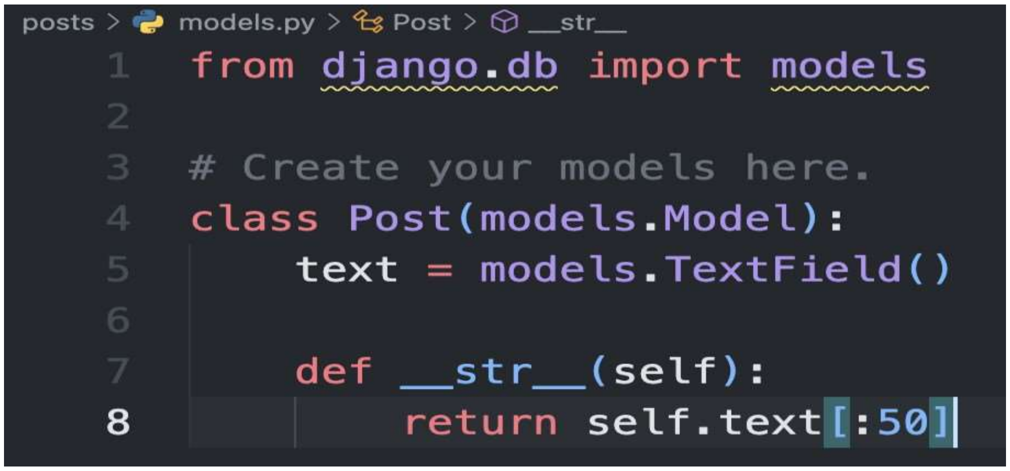
None available

5) We can manage our model in the admin interface (above page) by adding or modifying code in “posts/admin.py” and “posts/models.py”



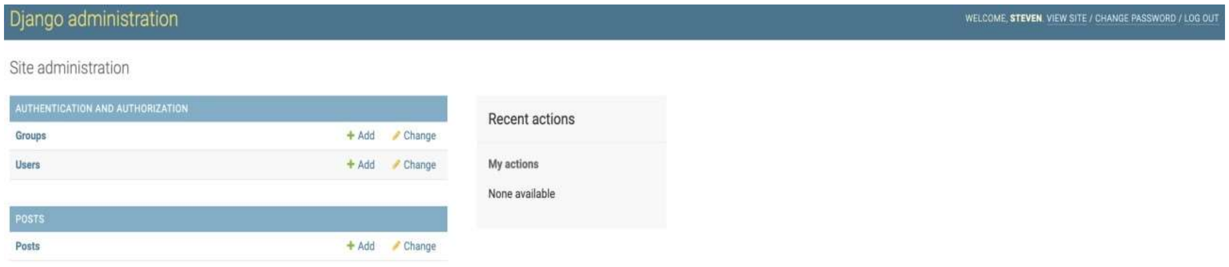
```python
from django.contrib import admin

from .models import Post

# Register your models here.

admin.site.register(Post)
```

We need a user friendly string representation for our objects so lets define a simple __str() function

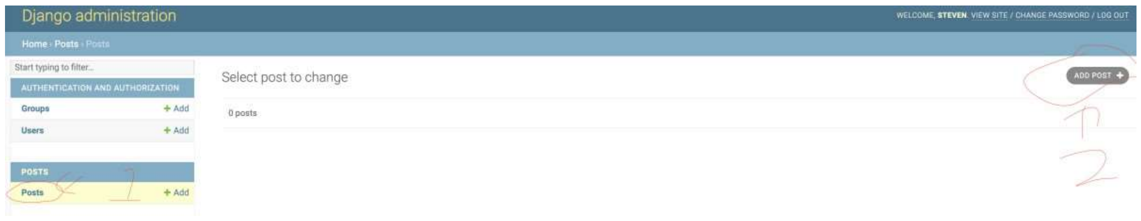


```python
from django.db import models

# Create your models here.
class Post(models.Model):
    text = models.TextField()

    def __str__(self):
        return self.text[:50]
```

6) Now refresh the “admin” page, and you will see:

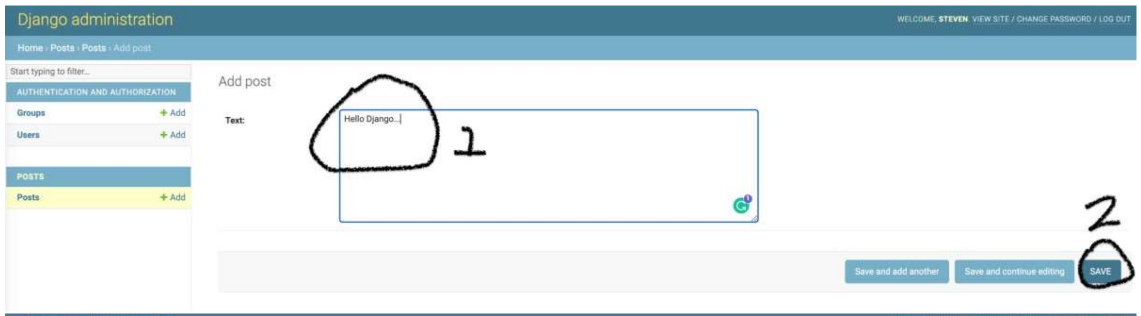


7) To create our first post


Django administration
WELCOME, STEVEN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Posts › Posts
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
POSTS
Posts + Add
1
ADD POST +
2

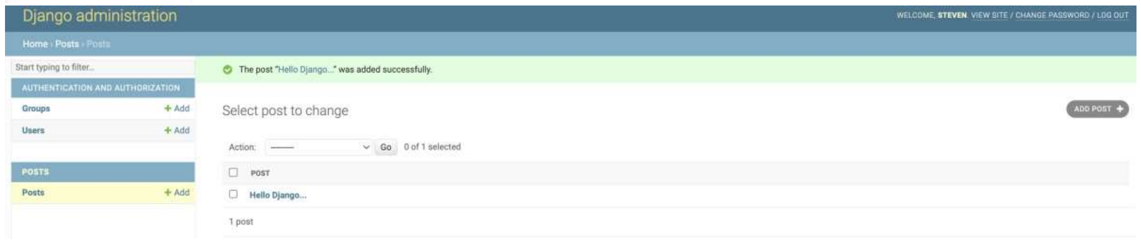
Select post to change

0 posts


Django administration
WELCOME, STEVEN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Posts › Posts › Add post
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
POSTS
Posts + Add
1
2
Save and add another
Save and continue editing
SAVE

Add post

Text: Hello Django...


Django administration
WELCOME, STEVEN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Posts › Posts
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
POSTS
Posts + Add
ADD POST +
Action: Go

The post "Hello Django..." was added successfully.

Select post to change

0 of 1 selected

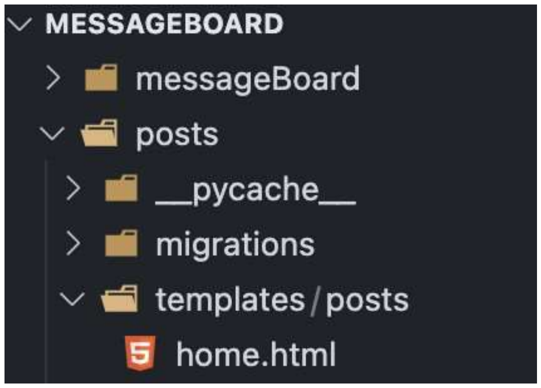
POST
Hello Django...

1 post

Display data (post) from our database

Before following the below steps, please press "Control + C" in terminal to stop the running server

a. Create "templates" folder under 'posts' folder
b. Create 'posts' folder under 'templates' folder. It is convention to store app templates like this. It is good practice and should be followed in all apps
c. Create 'home.html' in 'posts' folder

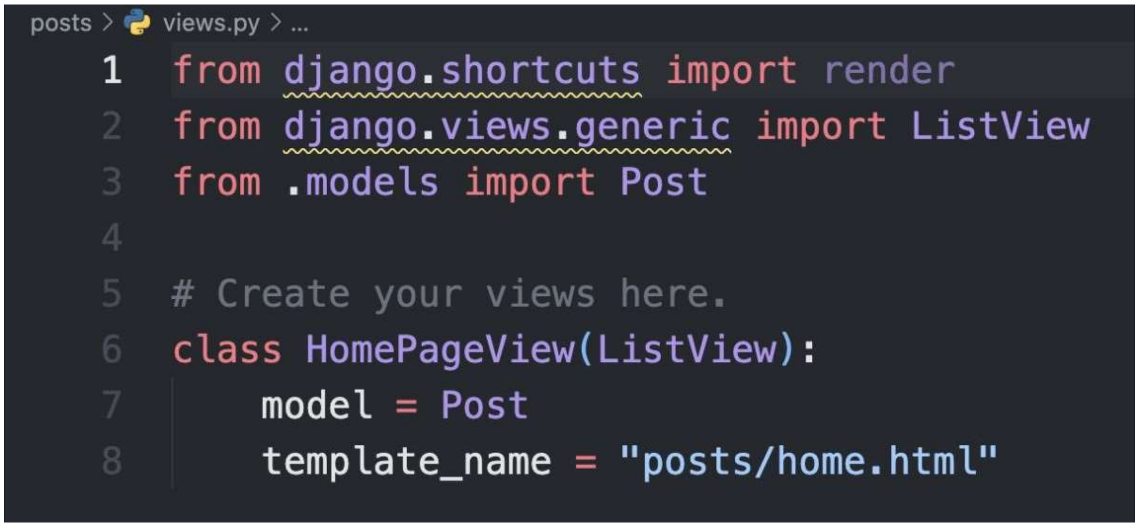


```
MESSAGEBOARD
  messageBoard
  posts
    __pycache__
    migrations
    templates / posts
      home.html
```

d. Create view in "posts/views.py".

ListView . It is one of the generic views that make creating some common view easier. In this instance to display a list of objects. You can find more details here

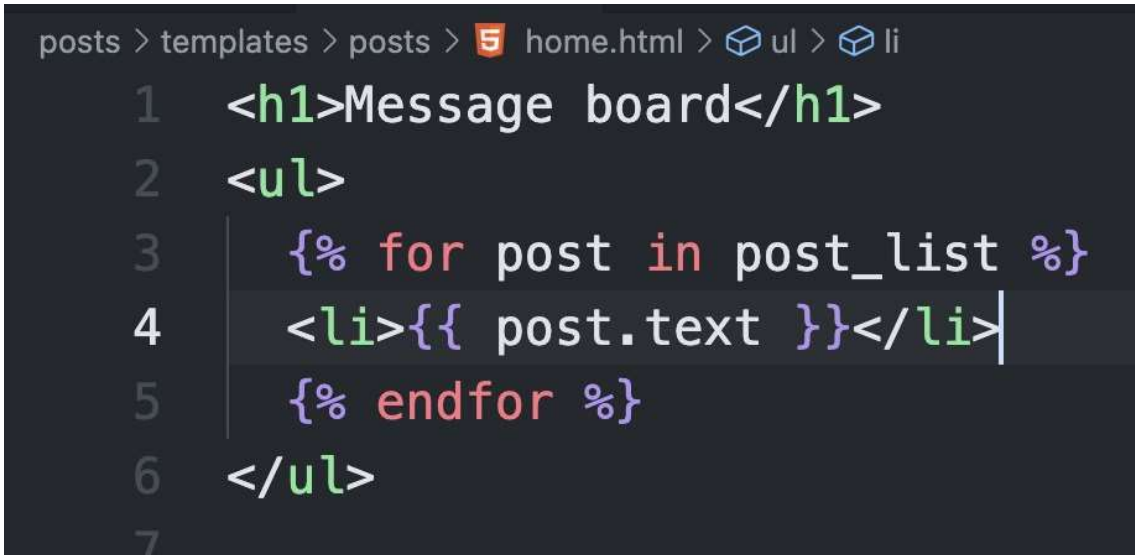
(https://docs.djangoproject.com/en/4.1/ref/class-based-views/genericdisplay/#generic-display-views)


posts > views.py > ...

```python
from django.shortcuts import render
from django.views.generic import ListView
from .models import Post

# Create your views here.
class HomePageView(ListView):
    model = Post
    template_name = "posts/home.html"
```

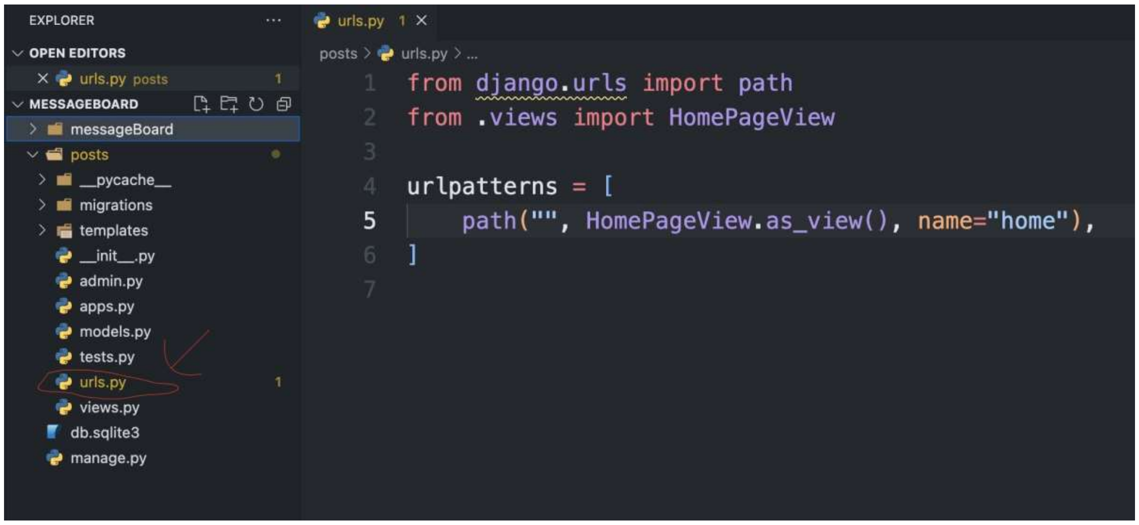
e. Add code in ‘home.html’

Note: ListView will automatically returns to us a context variable called <model>_list, where <model> is our model name. Our model name is post so, here is post_list


posts > templates > posts > home.html > ul > li

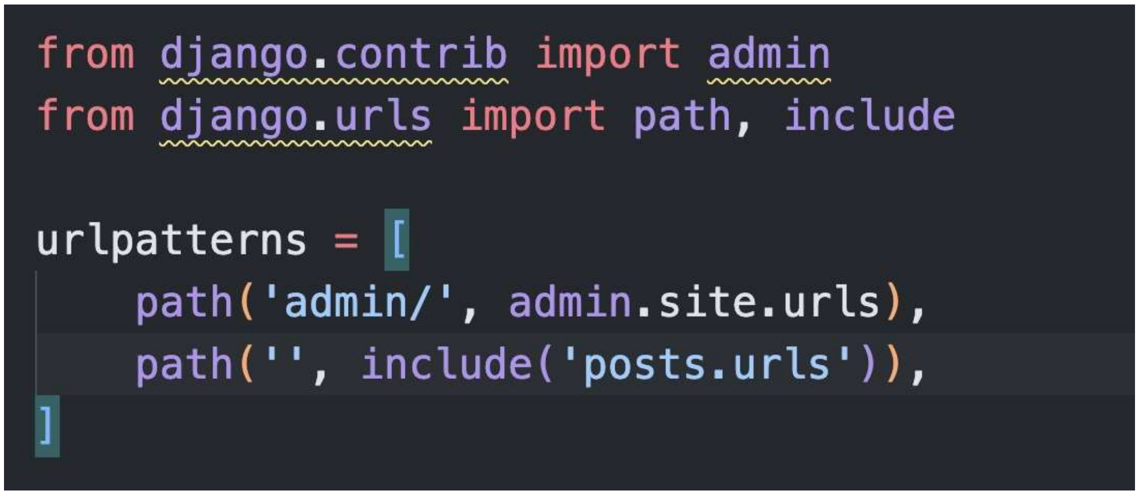
```html
<h1>Message board</h1>
<ul>
  {% for post in post_list %}
  <li>{{ post.text }}</li>
  {% endfor %}
</ul>
```

f. We need to provide user an entry point access the view and we can do that be defining a URL and connecting that with the view. We can define all URLs in urls.py file. Create a ‘urls.py’ under ‘posts’ folder (app folder) and then add the below code:



```python
from django.urls import path
from .views import HomePageView

urlpatterns = [
    path("", HomePageView.as_view(), name="home"),
]
```

g. We need to include this URL configuration in main project. Find 'urls.py' under project(messageBoard) folder and then add code



```python
from django.contrib import admin
from django.urls import path, include

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

h. Launch the server again by running "python manage.py runserver" command in your terminal and visit http://127.0.0.1:8000/, you will see post that you created in admin panel. Take a screenshot and send it to me on teams please.