Django -Lab 1a

Django – Djongo –MongoDB

# Environment Setup

1. Type “Anaconda Navigator” in search bar (Windows Button).
2. Till redirect you to Anaconda Navigator and there “Launch Cmd.exe Prompt”
3. Go to the folder where you want the Application to be
4. In the terminal, to create a virtual environment with conda, run the following command:

**conda create --name newEnv django**

**(base) D:\ML\_DL\_NLP\Django>conda create --name newEnv django**

Here, we created an environment called “newEnv” with the latest version of Django. Proceed with the default packages when prompted (i.e. select y)

1. To activate the environmen, run:

**activate newEnv**

**(base) D:\ML\_DL\_NLP\Django>activate newEnv**

Note: When an environment is activated, anything installed with pip or conda will only be installed for the given environment

1. If you get a django not intalled error, run the following command to install Django in this new environment

**pip install django**

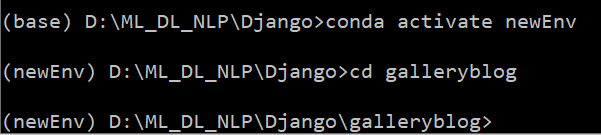
**(newEnv) D:\ML\_DL\_NLP\Django>pip install django**

# Create project

1. Create a folder called “django”. Use this folder to store all your Django projects.
2. Navigate to the “django” folder [Where you have created] in command-line using cd command and open the folder in atom’s project panel.

This is an example shown below:

Use your folder location using cd command

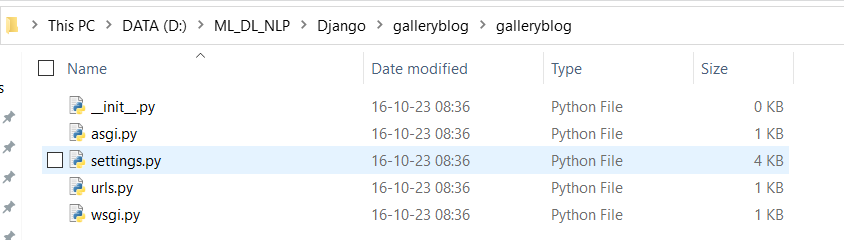


1. Using Django-admin, create your first project:

**django-admin startproject <app-name>**

**(newEnv) D:\ML\_DL\_NLP\Django>django-admin startproject galleryblog**

1. You will notice a folder “first\_project” created with the following files:



You have now successfully created a django project.

# Test Run

1. Now **navigate into the galleryblog** folder using cd command and run the following command:

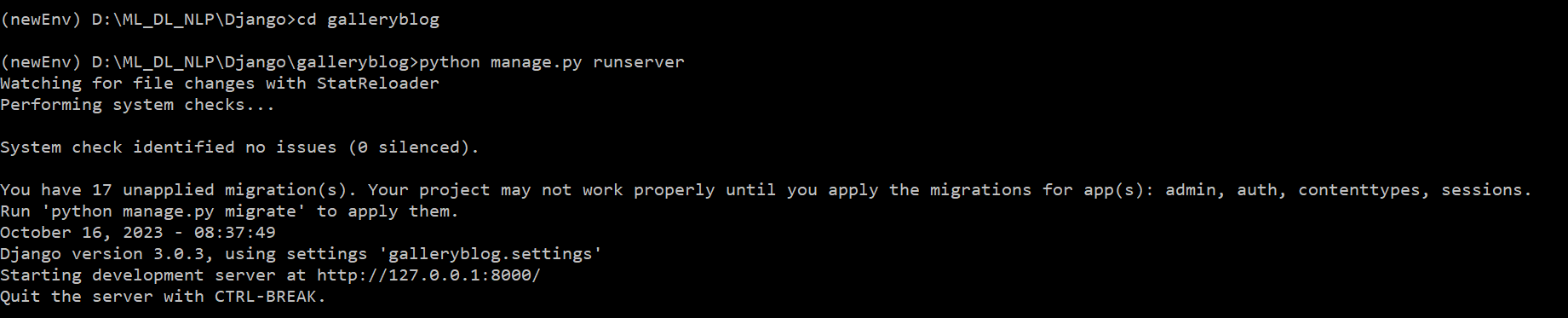
(newEnv) D:\ML\_DL\_NLP\Django>cd galleryblog

**To run the server: Python manage.py runserver**

(newEnv) D:\ML\_DL\_NLP\Django\galleryblog>**python manage.py runserver**

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

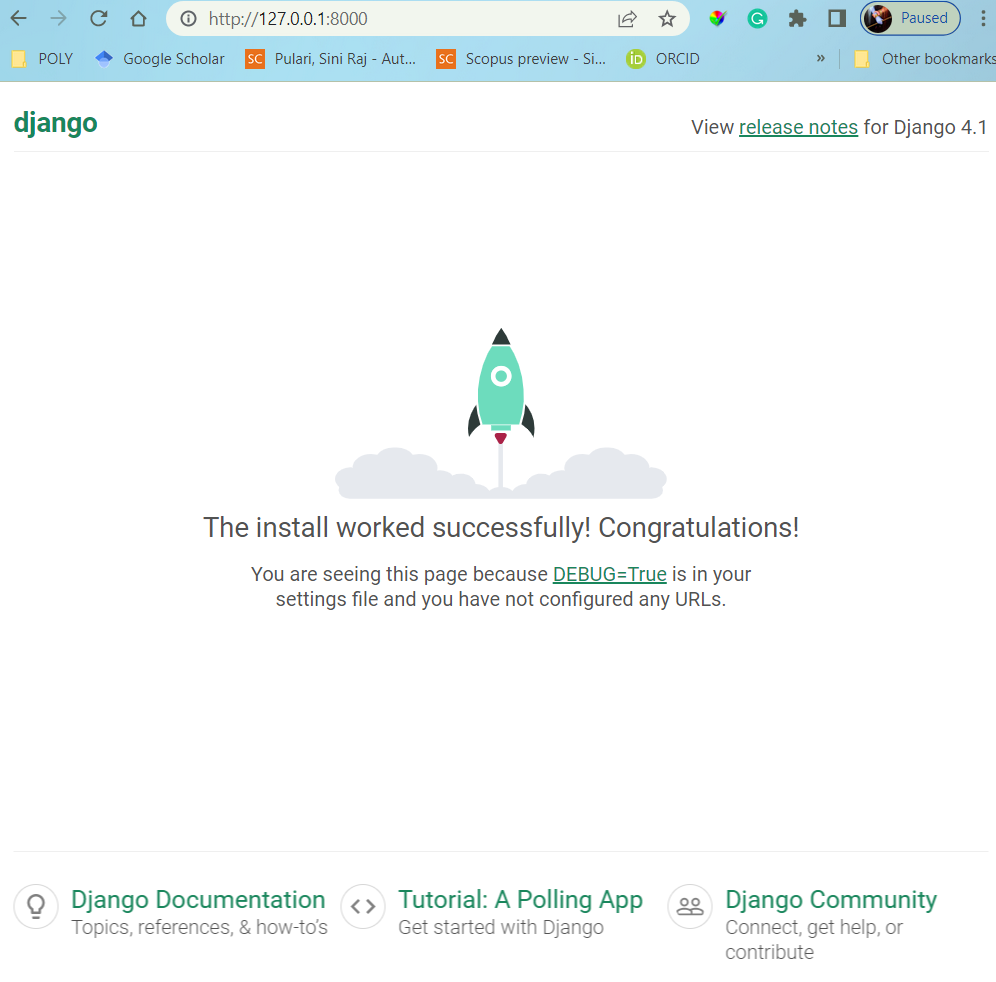
Output:



(Ignore migration setting error messages)

1. To run the project, copy the URL and paste it in your web browser:

**http://127.0.0.1:8000/**



Create a superuser – Django Admin

1. To create a superuser to access the Admin application, run the createsuperuser command in CMD:

**python manage.py createsuperuser**

1. You will be prompted to enter a username and password: enter **admin** and **abcd1234** for username and password respectively.

**(newEnv) D:\ML\_DL\_NLP\Django\galleryblog>python manage.py createsuperuser**

**Username (leave blank to use 'sinik'): admin**

**Email address:** [**admin@gmail.com**](mailto:admin@gmail.com)

**Password:admin**

**Password (again):admin**

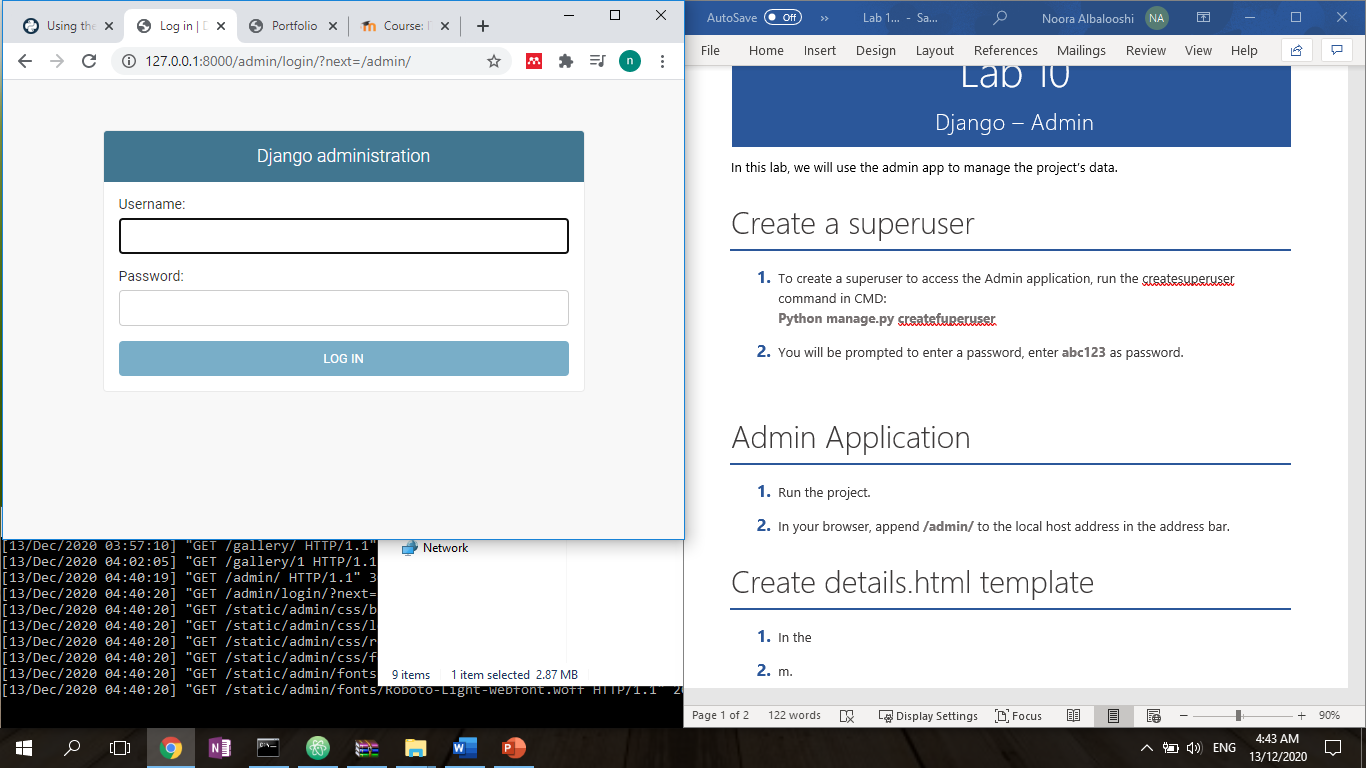
**;;;;;;y**

**Superuser created successfully.**

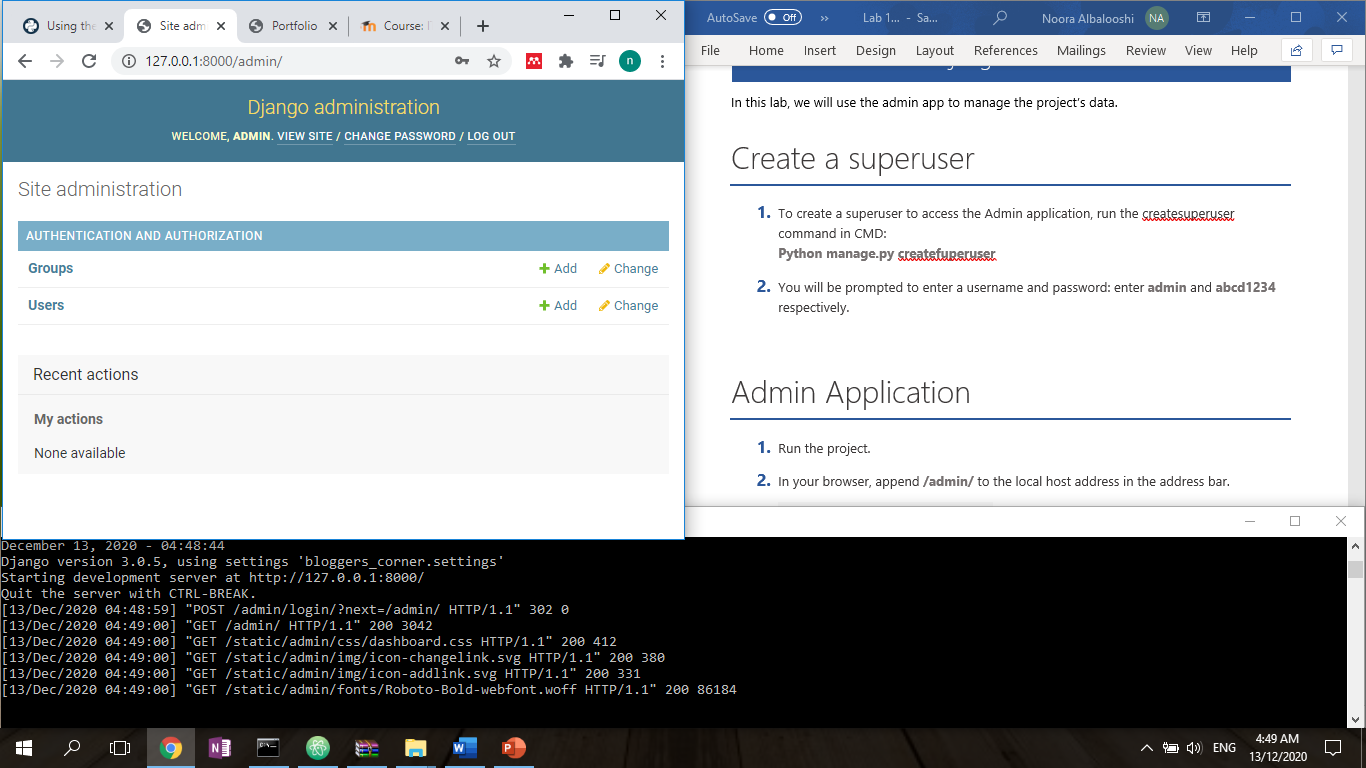
Admin Application

1. Run the project.
2. In your browser, append **/admin/** to the local host address in the address bar.

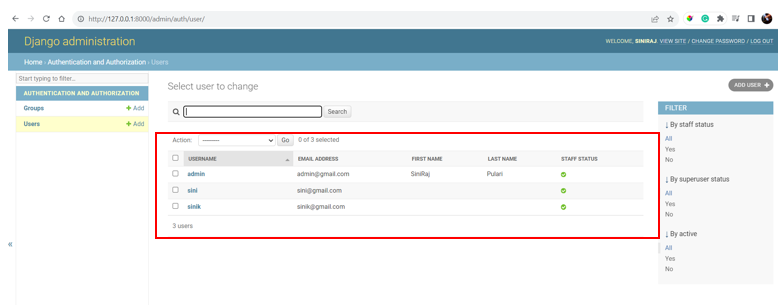
<http://127.0.0.1:8000/admin/>



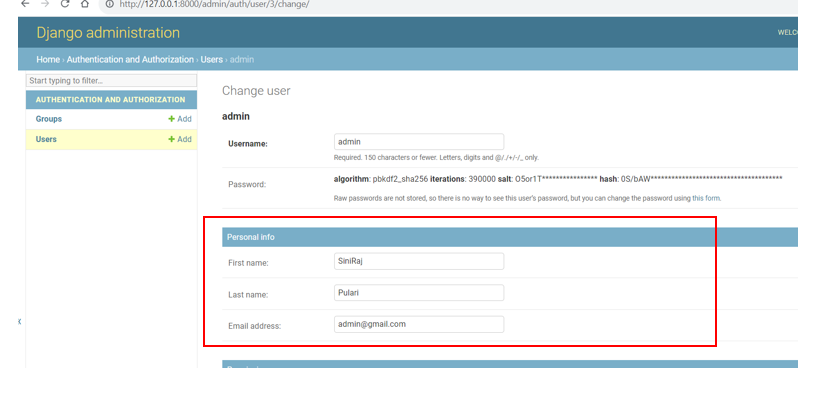
1. Enter the username and passowrd as defined in the previous section.
2. Once logged in, the following details are displayed:



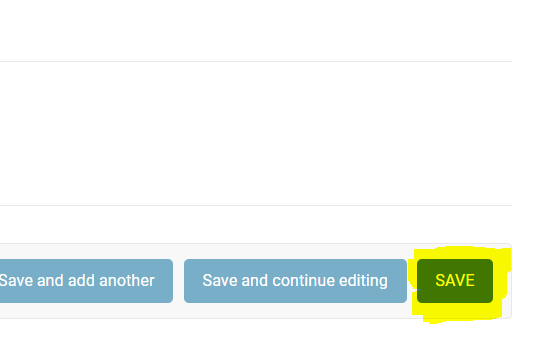
1. Click on users to see a list of registered users.



1. Click on the Admin User



1. Then scroll down and click save

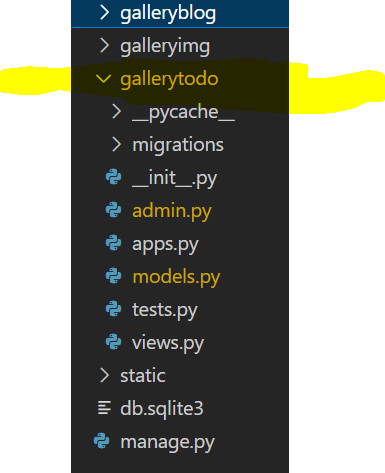


Creating the Applications in Django

1. Now, to create an application within this project, run:

**(newEnv) D:\ML\_DL\_NLP\Django\galleryblog>python manage.py startapp gallerytodo**

“galleryimg” folder will be created within your project. These are your application files.



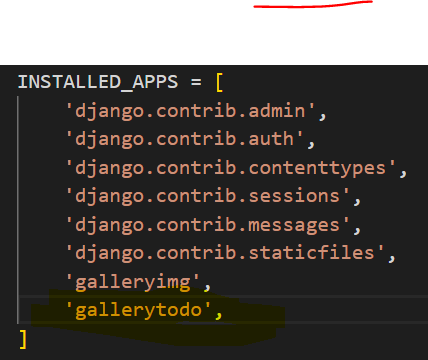
Please note!!!

**Please understand galleryblog is the project! And gallerytodo is the application in Django!**

**Hence whichever name you use doesn’t matter , however, please use the names appropriately while programming the code!**

GalleryBlog –settings.py

1. Add the application ‘gallerytodo’ to the project **galleryblog/settings.py** installed\_app list in settings.py



Settings.py Installed Apps section looks like as follows:

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'galleryimg',

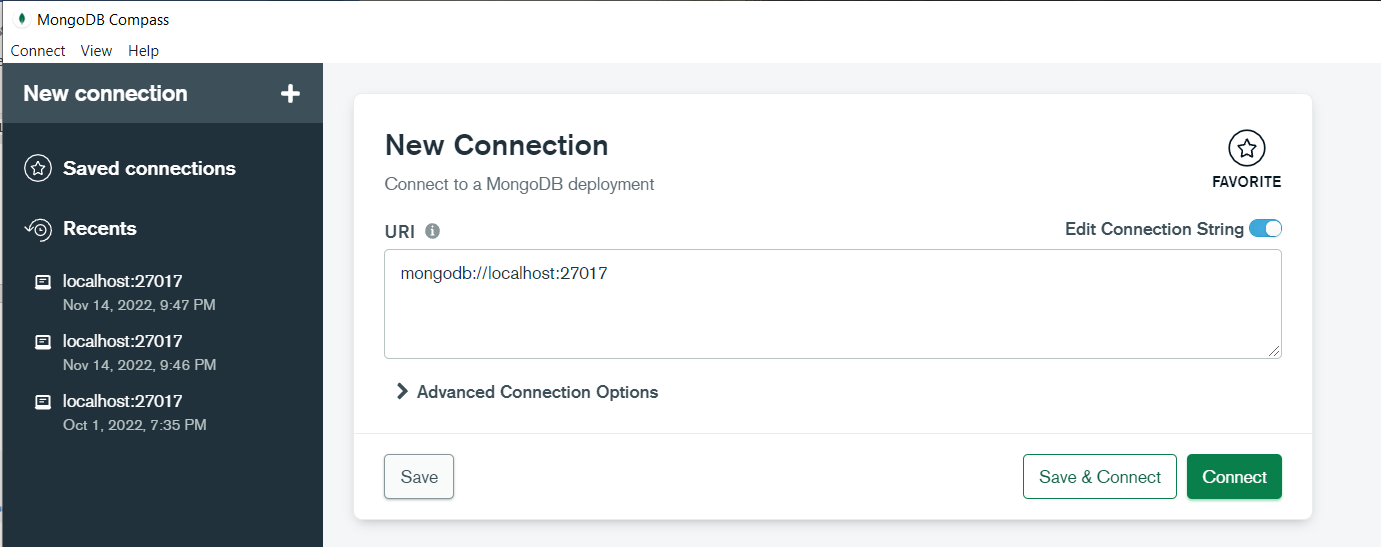
'gallerytodo',

]

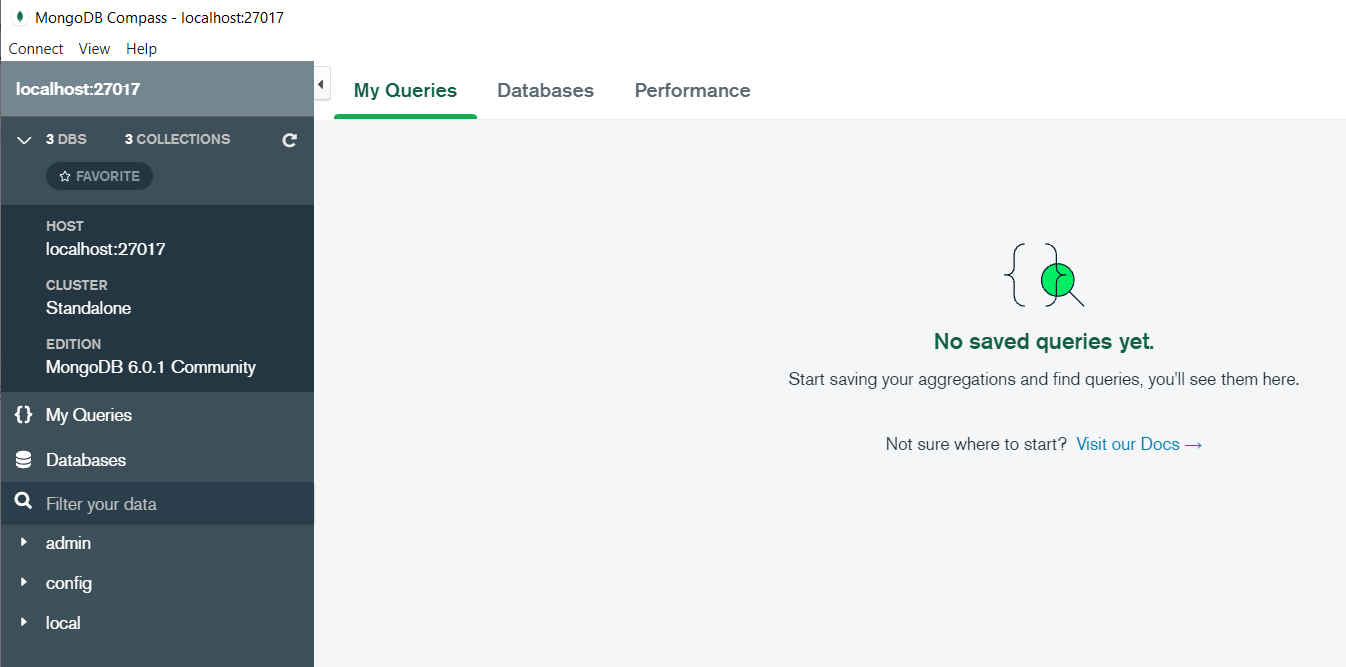
# Database setup - Compass

Skip step 1, 2 and 3 if you already installed MongoDB Compass

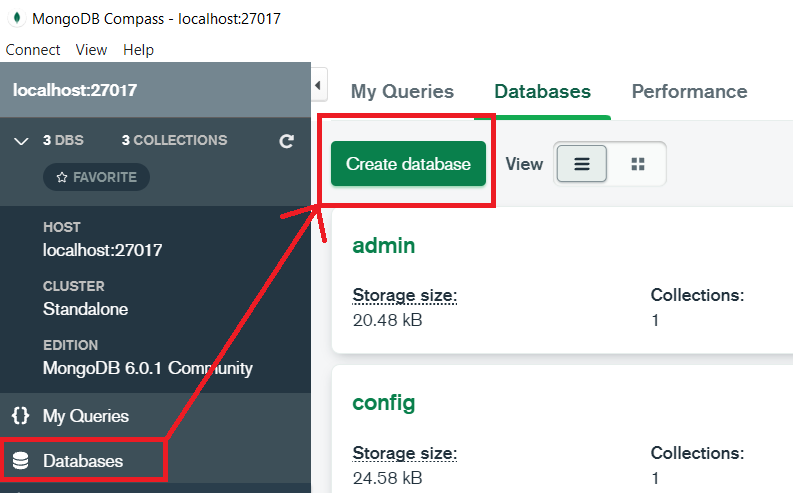
1. Download and install MongoDB Compass. MongoDB Compass provides a graphical user interface to manage your databases. <https://www.mongodb.com/try/download/compass?tck=docs_compass>
2. Run MongoDB Compass application.
3. In the first window, select connect; this allows you to connect to the local mongoDB server.



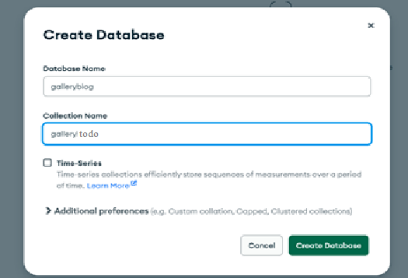
Click Connect and you will see something like given below:



1. Click on Database and it will show Create database



1. Create database called “blog” by clicking on CREATE DATABASE. We will store our applications data in this database.



1. Click on galleryblog database that appears in the list of databases. The following window displays the list of collections in the database.

# Database setup - Django

1. Now update the database information in the galleryblog/settings.py file in your project folder as below:

Code in galleryblog/settings.py

DATABASES = {

    'default': {

        'ENGINE': 'djongo',

        'NAME': 'galleryblog',

    }

}

# Create model

Now we will build our model class for the gallerytodo application. In **gallerytodo/models.py** file, include the following code:

In this class, we have 3 attributes: title, description and image.

* Title is of type CharField and has a maximum length of 50 characters
* Description is of type TextField 100 characters

Check out this link : https://docs.djangoproject.com/en/4.2/topics/db/models/

Next, we will see how Django automatically creates collections for our model class.

from django.db import models

# Create your models here.

class Todo(models.Model):

    task = models.CharField(max\_length=50)

    description = models.CharField(max\_length=100)

# Update Admin.py

1. Go to Admin.py and register the todo

from django.contrib import admin

from .models import \*

# Register your models here.

admin.site.register(Todo)

# Make Migrations and run server

1. Run the following commands

As you have done changes in the models.py

1. (newEnv) D:\ML\_DL\_NLP\Django\galleryblog>**python manage.py makemigrations**

Migrations for 'gallerytodo':

gallerytodo\migrations\0001\_initial.py

- Create model Todo

1. (newEnv) D:\ML\_DL\_NLP\Django\galleryblog>**python manage.py migrate**

Operations to perform:

Apply all migrations: admin, auth, contenttypes, galleryimg, gallerytodo, sessions

Running migrations:

Applying gallerytodo.0001\_initial...This version of djongo does not support "NULL, NOT NULL column validation check" fully. Visit https://nesdis.github.io/djongo/support/

OK

1. (newEnv) D:\ML\_DL\_NLP\Django\galleryblog>**pip install Djongo [ If u have already done no need to repeat!]**

**[ Yes , it is not a mistake, it is Djongo and not django…]**

Djongo is an improvement over PyMongo in that developers need not write lengthy queries. It maps Python objects to MongoDB documents, i.e., Object Document Mapping (ODM). Djongo ensures that only clean data enters the database. By performing integrity checks, applying validations, etc. with Djongo, there is no need to modify the existing Django ORM.

1. (newEnv) D:\ML\_DL\_NLP\Django\galleryblog>**python manage.py runserver**

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

October 19, 2023 - 00:39:45

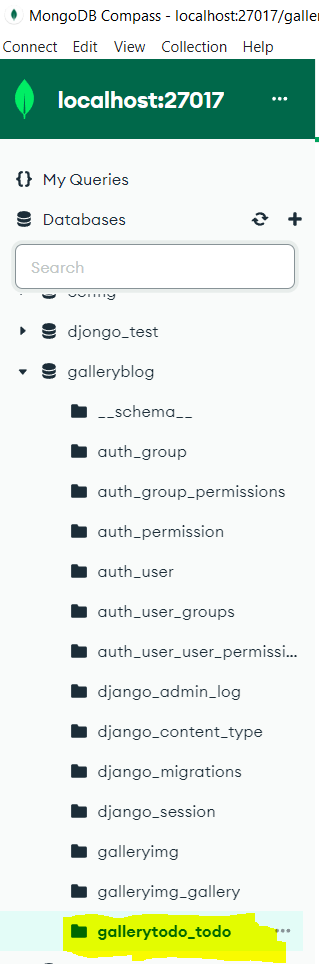
Django version 3.0.3, using settings 'galleryblog.settings'

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

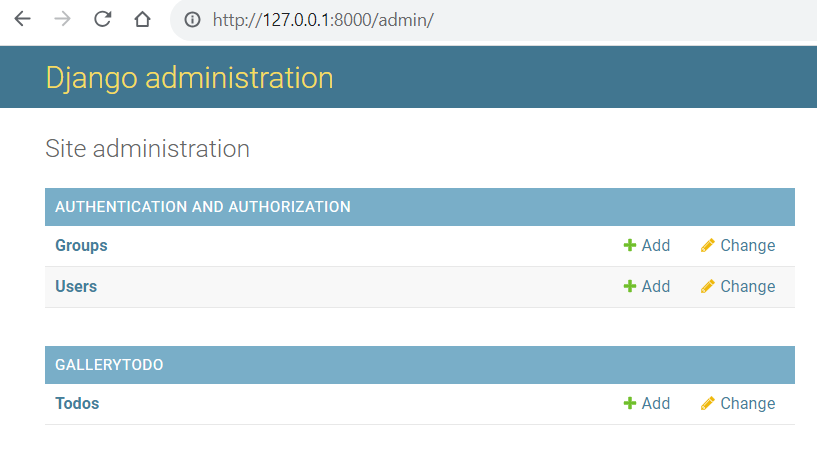
Quit the server with CTRL-BREAK.Observe the output

# MongoDB Compass - Updations

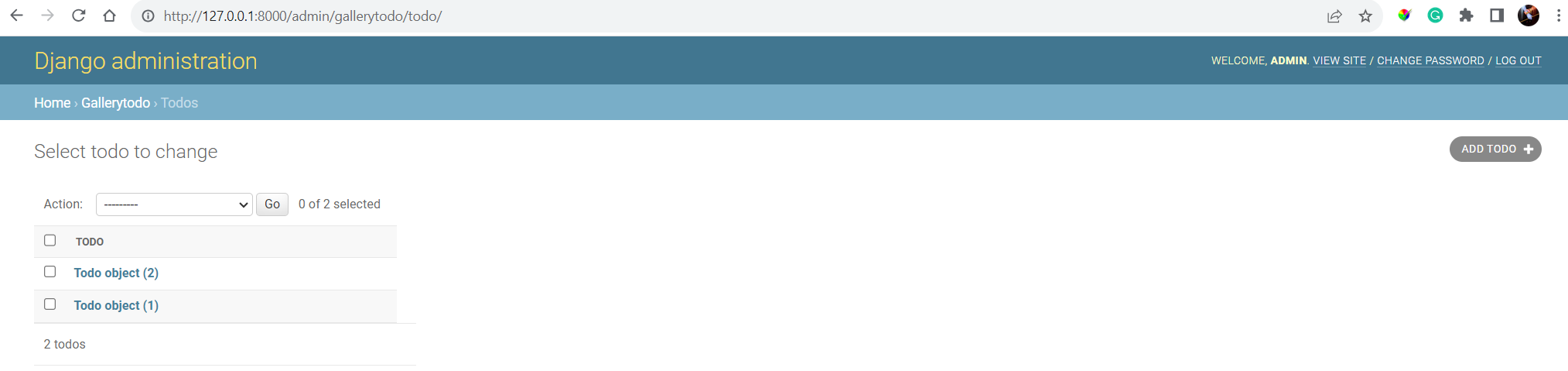
1. Refresh mongoDB Compass to see the updates.



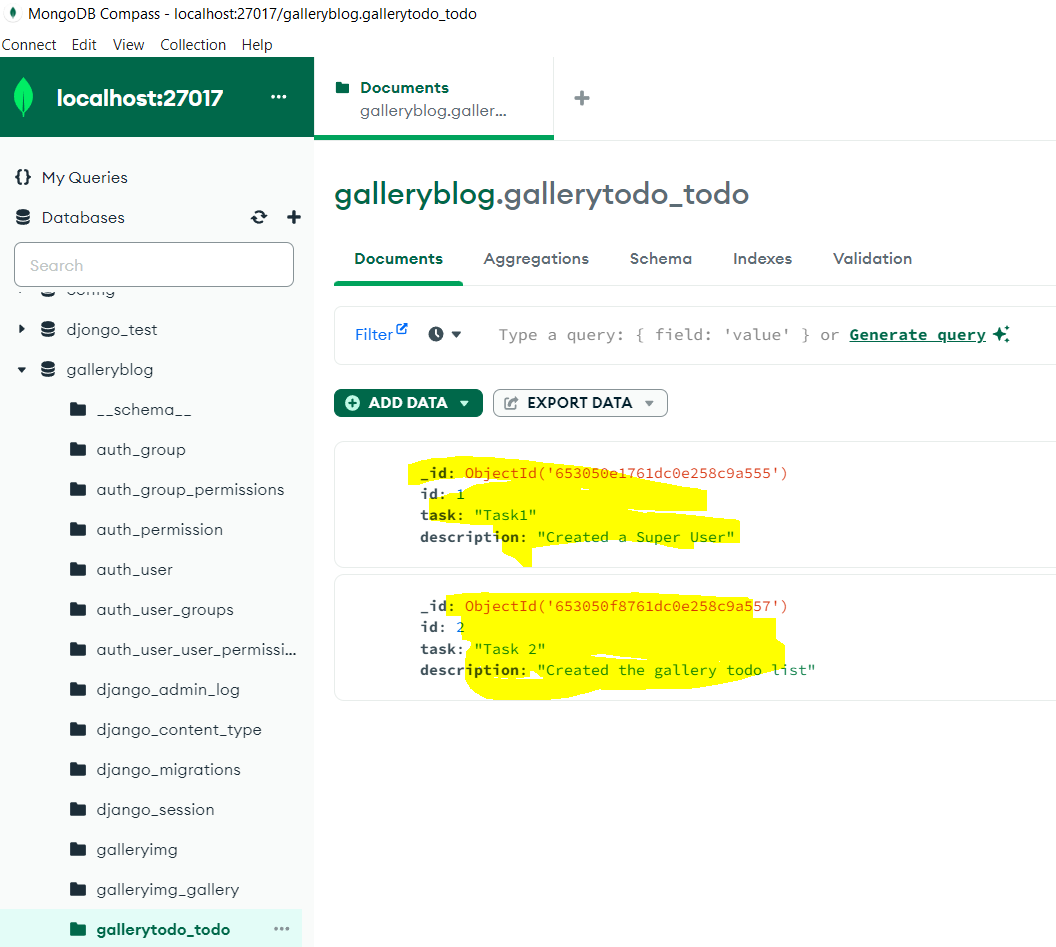
1. Go to <http://127.0.0.1:8000/admin>



1. Click on todo and add two tasks

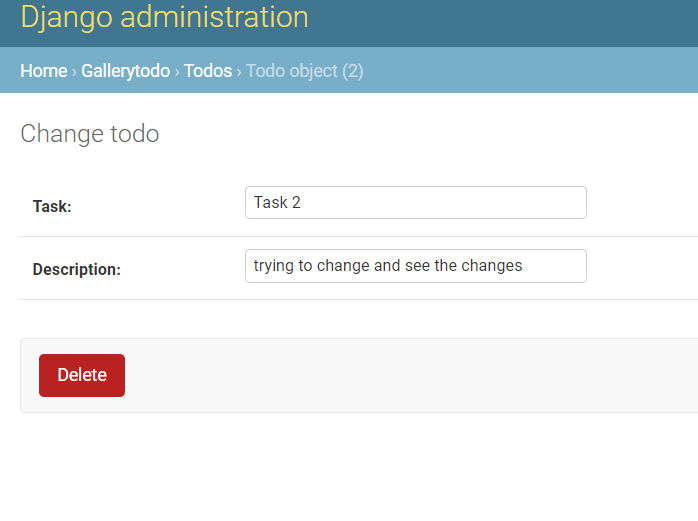


1. Go to MongoDB Compass and refresh!

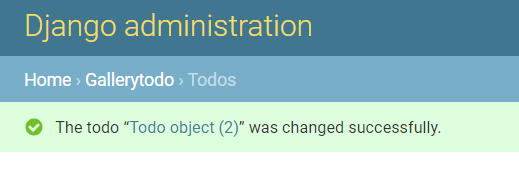


Those items are added in the database

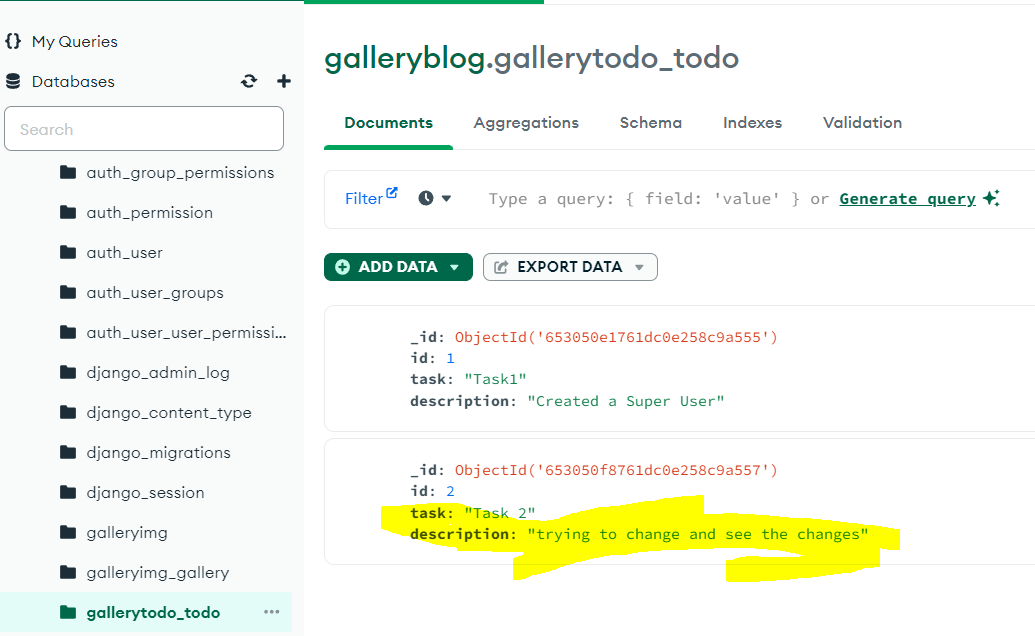
1. Go to Go to <http://127.0.0.1:8000/admin>
2. Update one task there and see the changes in the database



Save the changes in admin



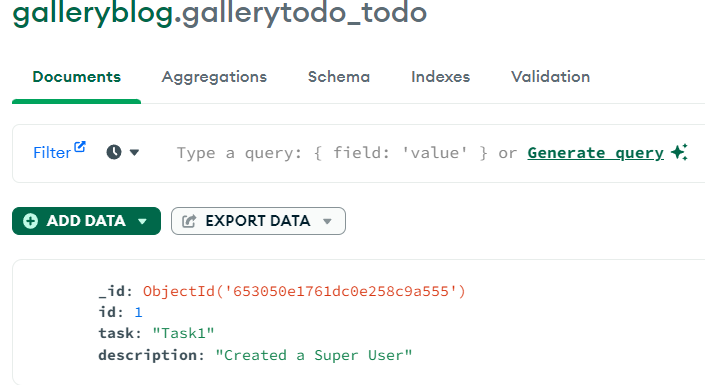
1. Go to MongoDb Compass and check for updates in the TASK 2



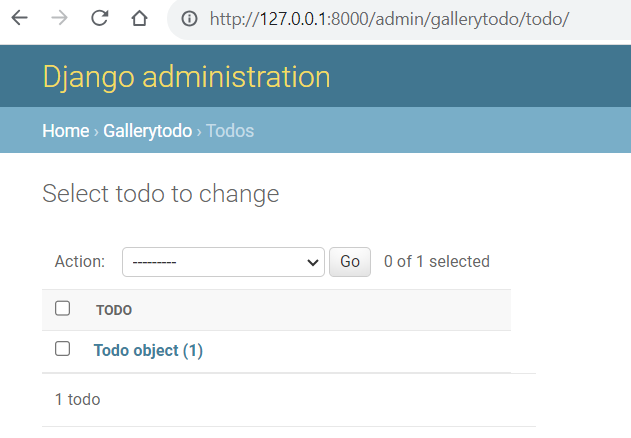
Please note sometimes, you will have to close mongoDB compass and then open to see the changes, sometimes, simple refresh will not work!

1. Delete Task 2 from MongoDB Compass and Go to <http://127.0.0.1:8000/admin> and refresh

Deleted the task2



Refreshing the Admin ,



These are the MONGOCRUD OPERATIONS done from Django Framework

☺