**Name= Atul Maurya.**

**Class = Btech (CSE)  
Section = “N”**

**Class Roll no: = 21**

**University Roll no: =2118371**

**Sem : =3rd ( 2nd year)**

**Student Id := 21011849**

**Bus Status Mini Project:**

Content:-

* Introduction Of the Bus Status Concept
* What is Django?
* What are the requirements and knowledge for Making the project
* Models
* Admin
* Views
* Urls
* Superuser and its creation.
* Commands:- ( makemigration , migrate …etc)
* Introduction Of the Bus Status Concept:-

The Bus Status Project is based on Web based Projects By using Django, HTML and CSS .

I got the idea to built this project because I saw that the Student Of our University Facing issues in the college for finding their buses that which bus number will go to their destination and When .

So , I thought that What if all can see their Bus Status on their Mobile phone Just by going to the website and *Entering the Details that Where they want to go and the Time Then After Pressing the Search Button they Can get the details That Which Bus Number is ready to go to their desired location.*

And the Bus Driver Who as a admin will registering the information about the bus details by login in the Website.

* What is Django?
* Django is a Python-based web framework that allows you to quickly create efficient web applications. It is also called batteries included framework because Django provides built-in features for everything including Django Admin Interface, default database – SQLlite3, etc.
* When you’re building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use and that too for rapid development.

**Why Django Framework ?**

* Excellent documentation and high scalability.
* Used by Top MNCs and Companies, such as Instagram, Disqus, Spotify, Youtube, Bitbucket, Dropbox, etc. and the list is never-ending.
* Easiest Framework to learn, rapid development and Batteries fully included.
* The last but not least reason to learn Django is  Python has huge library and features such as Web Scrapping, Machine Learning, Image Processing, Scientific Computing, etc. One can integrate it all this with web application and do lots and lots of advance stuff.

**Installation of Django**

* Install python3 if not installed in your system ( according to configuration of your system and OS) from [here](https://www.python.org/downloads/) . Try to download the latest version of python it’s python 3.11.0 this time.

**Note-** Installation of Django in Linux and Mac is similar, here I am showing it in windows for Linux and mac just open terminal in place of command prompt and go through the following commands.

* **Install pip-** Open command prompt and enter following command-

python -m pip install -U pip

* **Install Django-** Install django by giving following command-

pip install django

* What are the requirements and knowledge for Making the project:-

**1. Basic knowledge of the Django framework :-**

* Models
* Admin
* Views
* Urls
* makemigration , migrate
* Superuser and its creation.
* Different Types of Modules in djangoHTTP response , redirect

**2. First-hand experience in web development**

**3. Proficiency in Python**

**4. An understanding of the MySQL and SQL databases**

**5. Efficient testing, debugging codes**

**6. Building complex websites with large data processing requirements**

**7. An understanding and basic knowledge of front-end development**

**8. Verification of code functionality.**

* Models :-

According to Django Models A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you’re storing. Generally, each model maps to a single database table.

The basics:

* Each model is a Python class that subclasses django.db.models.Model.
* Each attribute of the model represents a database field.
* With all of this, Django gives you an automatically-generated database-access API.

Django models are used as a structure to define fields and their types which will be saved in the database. Whatever changes we want to make in the database and want to store them in the database permanently are done using Django Models. A table for a phone in the database

### Create table in database using Django –

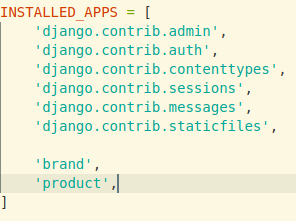
We need to create a new app named **product** so that we can define all properties of phone described in the above-provided image. Open your terminal and run following command:

python manage.py startapp product

Now, our directory will be:-



After creating the app, don’t forget to mention it in account/settings.py under **INSTALLED\_APPS**.



Also, register it with admin by adding following line of code to **product/admin.py**. By registering it with admin, you make sure that admin of the site will be aware that new table schema for database has been prepared.

Like this:---

from django.contrib import admin

# Register your models here.

from product.models import Phone

admin.site.register(Phone)

Now, navigate to **product/models.py**. You will see a file with following lines:

from django.db import models

# Create your models here.

We use a python class to define models which inherit parent class named **Model** defined in **django.db.models** package.

Add the following lines of code to this file:

from django.db import models

# Create your models here.

class Phone(models.Model):

Price = models.IntegerField()

RAM = models.IntegerField()

ROM = models.IntegerField()

Front\_camera = models.IntegerField()

* Admin:-

The Django admin is simply a Django app (a module that allows you to do a especific set of tasks using Django) that provides you with a simple interface to perform CRUD operations on the models that are configured to be shown in the admin website.

It is meant to be a simple way to manage the model instances of your project without the need to spend extra time developing a custom UI to be able to do so.

However, it does provide the developer with a lot of customization options that offer features such as custom permissions, custom form validation and even support for using custom templates for the admin interface.

Register:-

If the default Django admin interface fulfills all your needs, all you need to do to be able to add a specific Model to the admin is to register your model using **django.contrib.admin.site.register**() as follows:

**Admin Site:-**

Each Django admin site is represented by an AdminSite class. In most cases, you don't have to use this class, however, if you happen to want to edit aspects of the admin site that are not associated with a particular model but rather are more general in nature, this is the class that allows you to perform those customizations.

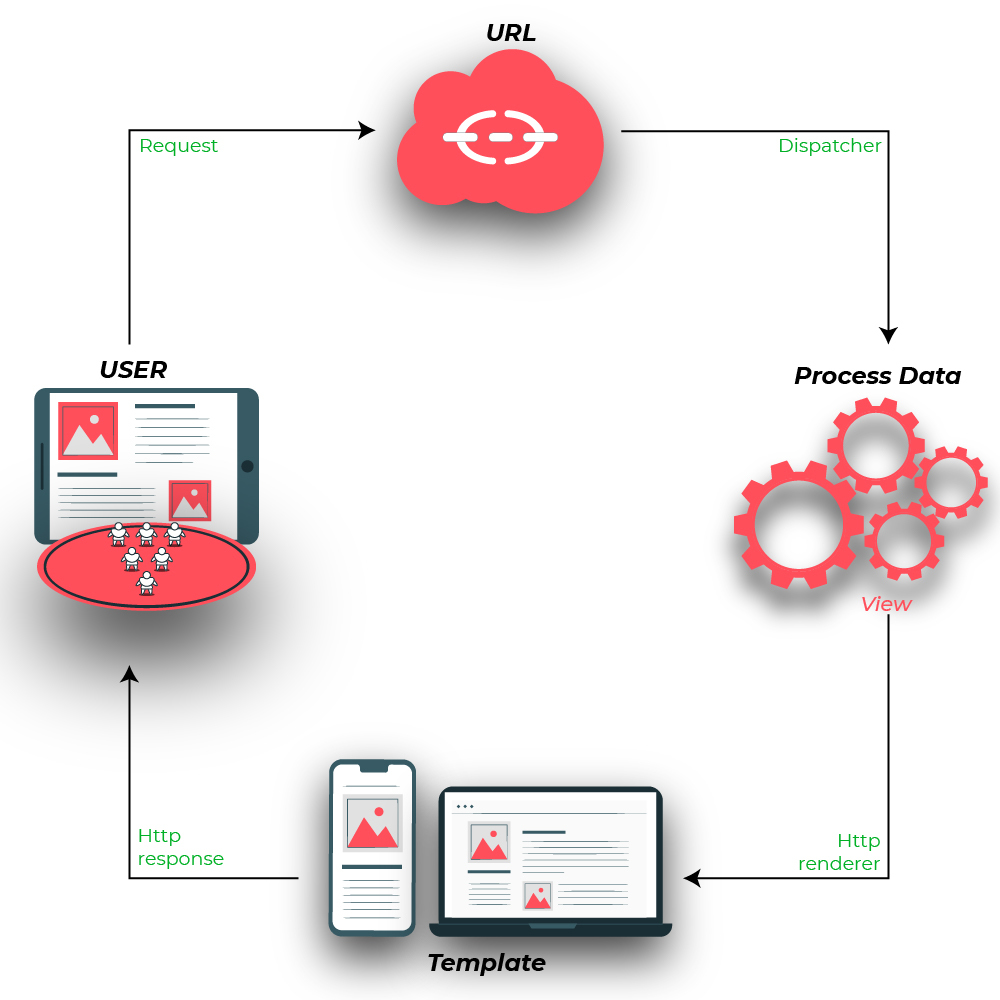
**Views:-**

Django Views are one of the vital participants of [M**V**T Structure of Django](https://www.geeksforgeeks.org/django-project-mvt-structure/). As per Django Documentation, A view function is a Python function that takes a [Web request and returns a Web response](https://www.geeksforgeeks.org/django-request-and-response-cycle-httprequest-and-httpresponse-objects/).

This **response** can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image, anything that a web browser can display.

Django views are part of the user interface — they usually render the HTML/CSS/Javascript in your Template files into what you see in your browser when you render a web page.

Django views roughly correspond to controllers in MVC, and Django templates to views in MVC.)



There are two type of Views:-

* Function based
* Class Based
* Urls:-

A Django template is a text document or a Python string marked-up using the Django template language. Django being a powerful Batteries included framework provides convenience to rendering data in a template. Django templates not only allow passing data from view to template, but also provides some limited features of programming such as variables, for loops, comments, extends, url, etc.  
This article revolves about how to use **url tag** in Templates. url tag Returns an absolute path reference (a URL without the domain name) matching a given view and optional parameters. This is a way to output links without violating the DRY principle by having to hard-code URLs in your templates:

**Syntaxt:-**

**{% url 'some-url-name' v1 v2 %}**

The first argument is a URL pattern name. It can be a quoted literal or any other context variable. Additional arguments are optional and should be space-separated values that will be used as arguments in the URL

###### **{% url 'template1' %}**

**Example:-**

suppose you have a view, **app\_views.client**, whose URLconf takes a client ID (here, client() is a method inside the views file app\_views.py). The URLconf line might look like this:

path('client/<int:id>/', app\_views.client, name='app-views-client')

If this app’s URLconf is included into the project’s URLconf under a path such as this:

path('clients/', include('project\_name.app\_name.urls'))

…then, in a template, you can create a link to this view like this:

{% url 'app-views-client' client.id %}

The template tag will output the string **/clients/client/123/.**

* Superuser and its creation:-

Django provides us Admin Panel for it’s users. So we need not worry about creating a separate Admin page or providing authentication feature as Django provides us that feature. Before using this feature, you must have migrated your project, otherwise the superuser database will not be created.

**How to create superuser in Django?**

For creating superuser, first reach the same directory as that of**manage.py** and run the following command:

python manage.py createsuperuser

Then enter the Username of your choice and press enter.

Username: srishti

Then enter the Email address and press enter.(It can be left blank)

Email address: example@gmail.com

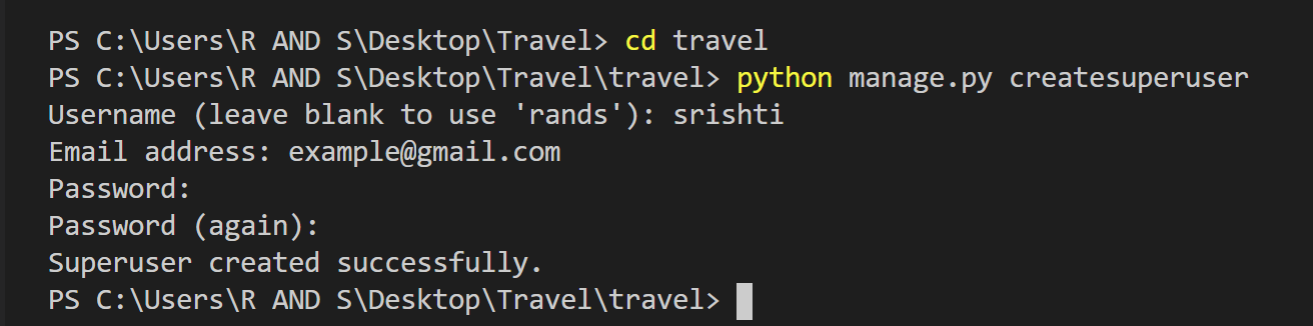
Next, enter the Password in-front of the Password field and press enter.Enter a strong password so as to keep it secure.

Password: \*\*\*\*\*\*

Then again enter the same Password for confirmation.

Password(again): \*\*\*\*\*\*

Superuser created successfully if above fields are entered correctly.





* Commands in Django:-

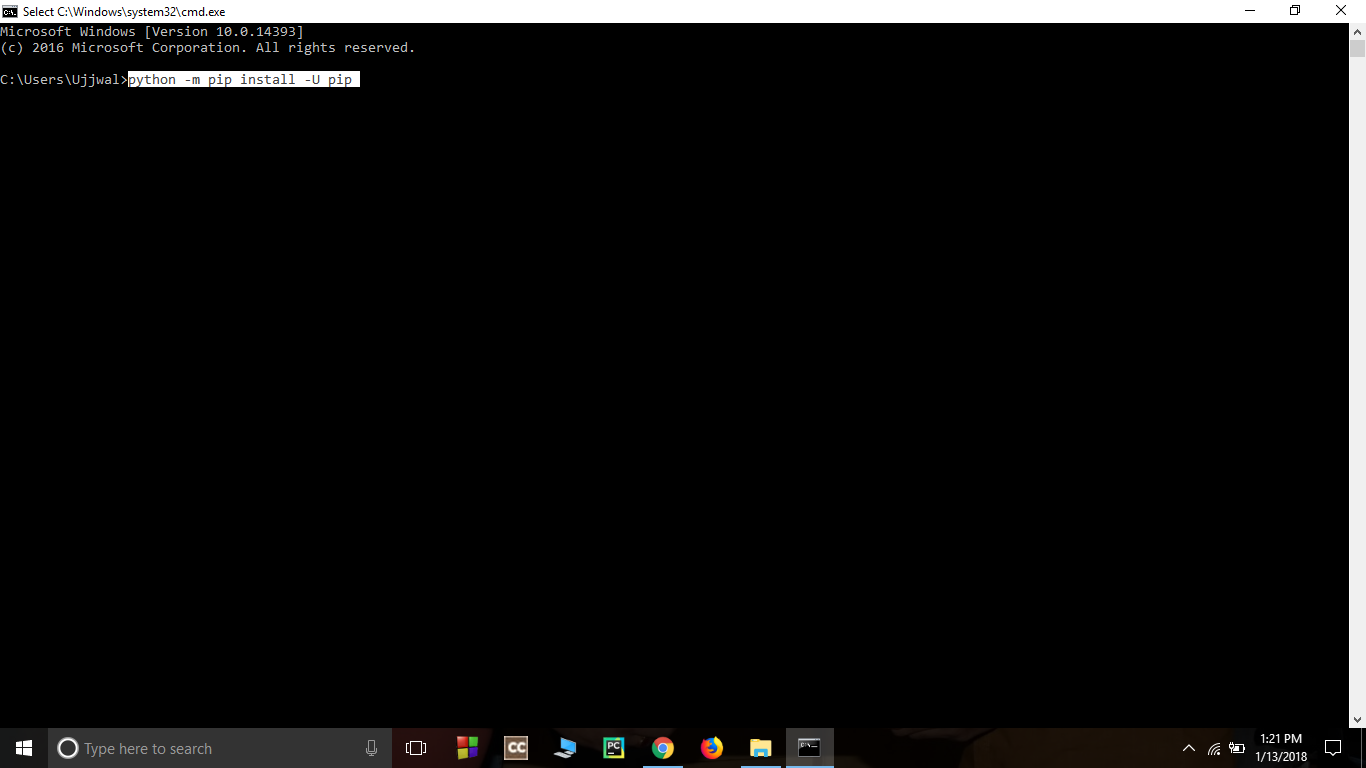
Manage.py in Django is a command-line utility that works similar to the django-admin command. The difference is that it points towards the project’s settings.py file. This manage.py utility provides various commands that you must have while working with Django. Some of the most commonly used commands are –

* python manage.py startapp
* python manage.py makemigrations
* python manage.py migrate
* python manage.py runserver

Interestingly we can create our own Custom Management Commands to fulfill a wide variety of requirements ranging from interacting with our application using the command line to serve as an interface to execute Cron Jobs. We are going to create a custom Management Command which gives us the stats or metrics of new articles published, comments on those articles on a particular day.

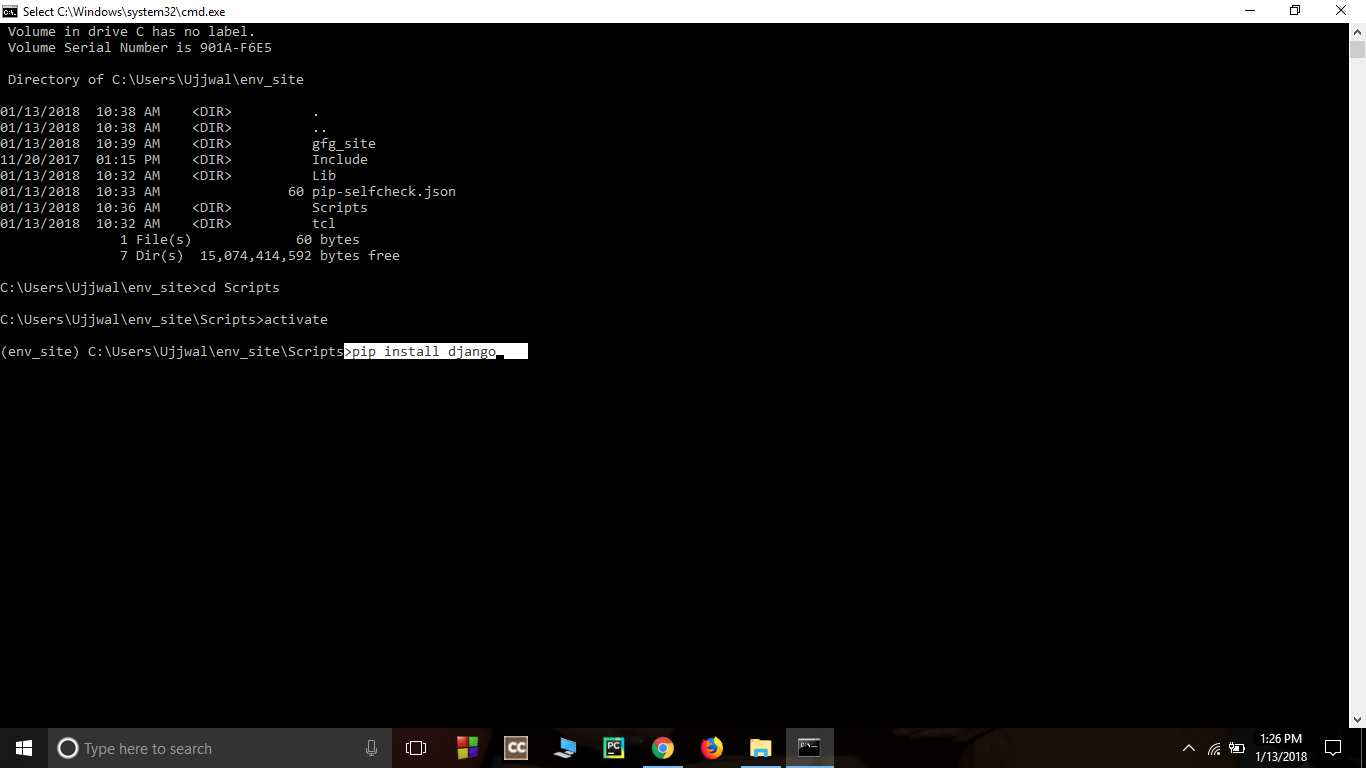
* **Install pip-** Open command prompt and enter following command-

python -m pip install -U pip



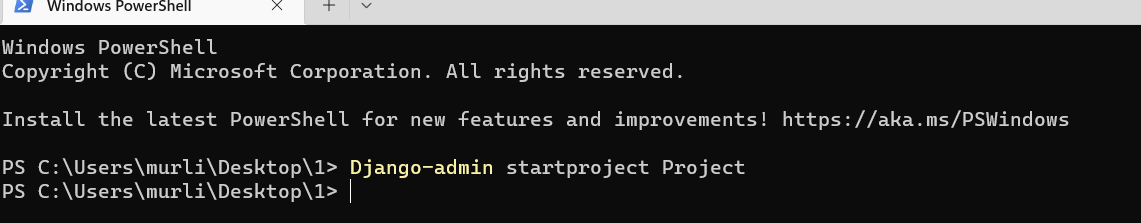
* **Install Django-** Install django by giving following command-

pip install django



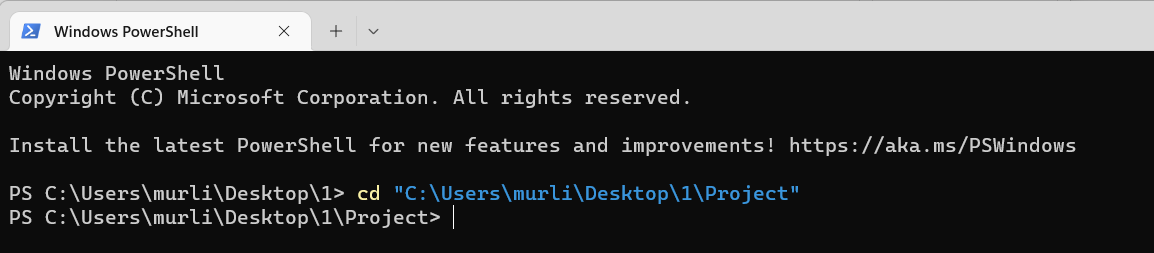
* Start a project by following command-

django-admin startproject Project



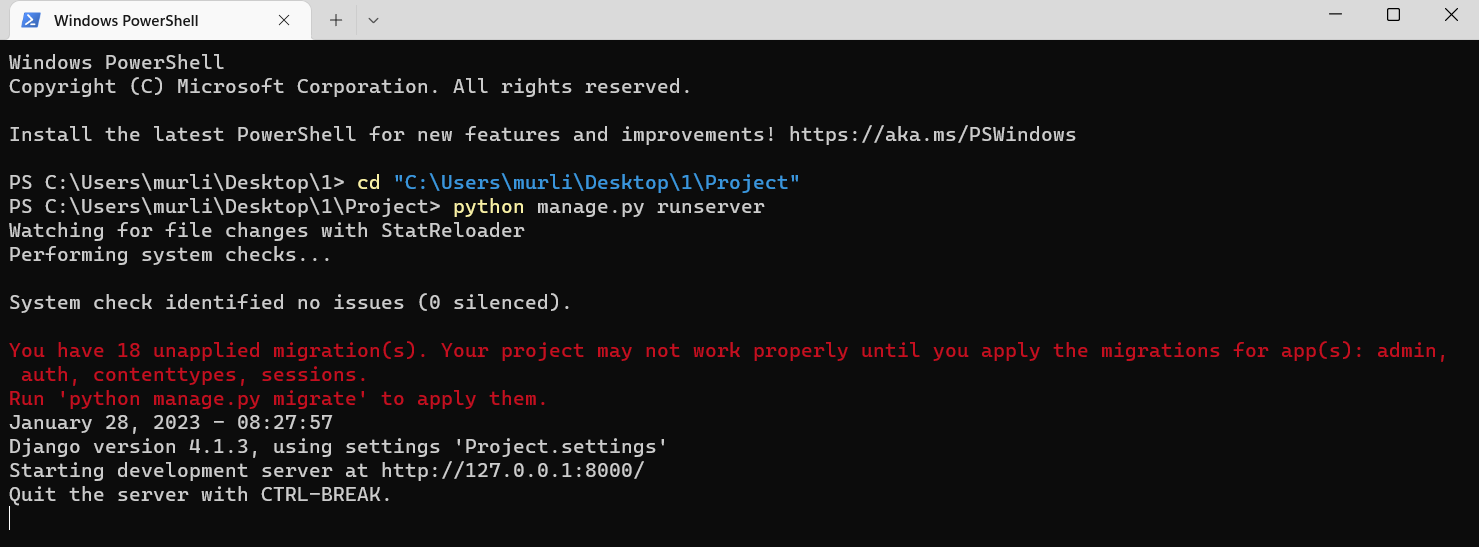
* Change directory to Project

cd Project



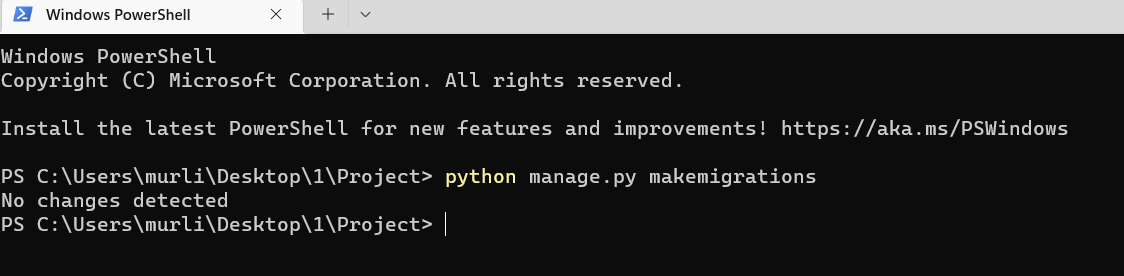
* **Start the server-**Start the server by typing following command in cmd-

python manage.py runserver



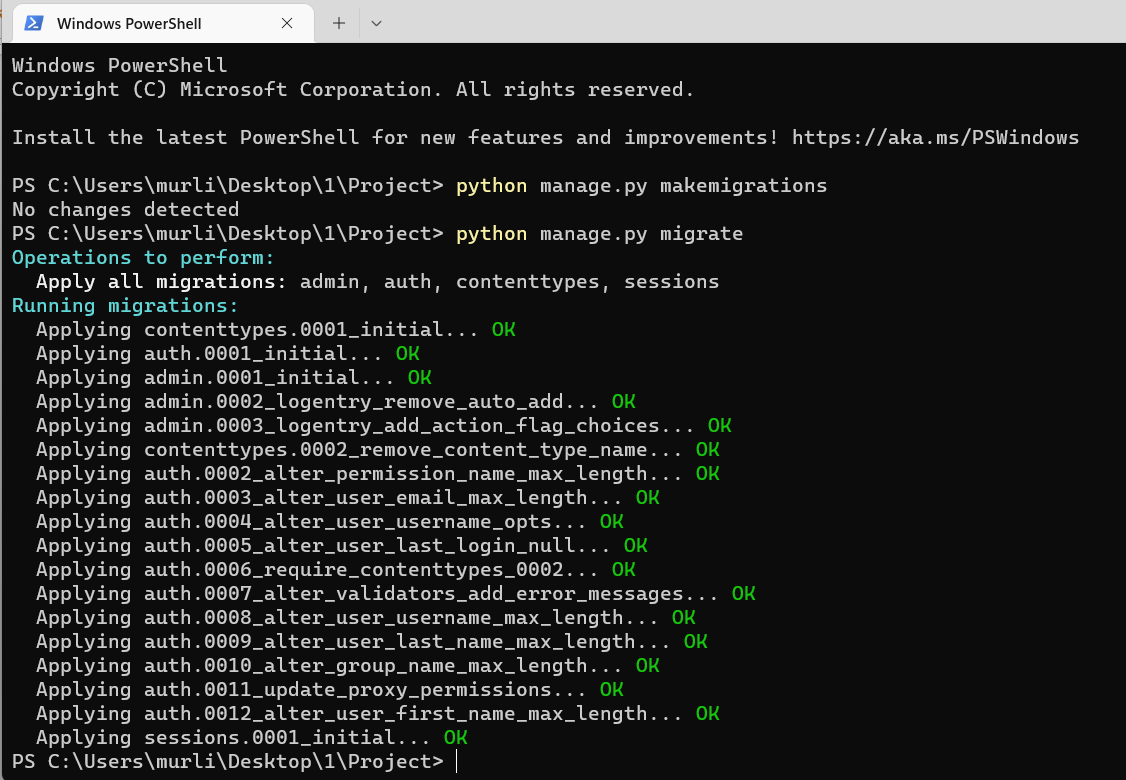
* **makemigrations:-**

**Python manage.py makemigrations**



* **migrate:-**

**Python manage.py migrate**



On Runnig the server the first website opened is:-

