**Core Django**

Create virtual environment:

1. pip install virtualenvwrapper-win
2. mkvirtualenv envname
3. workon envname
4. For deactivate the virtual env just type : **deactivate**
5. Delete virtual env. : **rmvirtualenv ENVNAME**
6. Update pip version **python -m pip install --upgrade pip**

OR

1. **pip install virtualenv**
2. **python -m venv envname**
3. **.\envname\Scripts\activate.bat**
4. **deactivate**

Install Django In public Directery

1. pip install django
2. py -m django --version
3. django-admin startproject firstproject
4. go in firstproject dir.(cd firstproject)
5. Python manage.py startapp appname
6. Python manage.py makemigrations
7. Python manage.py migrate
8. Python mange.py runserver

**Fake Migrations**

If no migrations apply in database then run this command:

* **python manage.py migrate --fake products zero**
* Here products in table name

**Starting with github project**

**Clone the repo.:**

$ git clone https://github.com/realpython/image-of-the-day.git

$ cd image-of-the-day/

$ git checkout tags/start\_here\_py3

**Some Git-Hub Command**

1. It will print all your remotes' fetch/push URLs(Pritnt all repo.) : **git remote -v**
2. it will show the URL or the remote repo : **git remote show origin**
3. remove origin from git repository **: git remote rm origin**
4. Add Heroku github url : **heroku git:remote -a yourapp**
5. To reset last commit : **git reset HEAD~**
6. **Fetch all branch :** git fetch –all
7. **Get logs** : git log –oneline

**GitHub Stash command**

1. **Help command :** git stash –help
2. **Apply stash :** git stash -u
3. **See all stash list :** git stash list
4. **Fetch and delete(last stash) stash :** git stash pop
5. **Fetch stash but not delete last stash** : git stash apply

**Vim command**

* **To add toke in URL :** vim .git/config
* **Login throughout token and username in github :**

https://user\_name:token@github.com/kp-bitcoding/Flask-crud.git

* **Insert mode** : press **i** and **Enter**
* **Exit Without save data :** Esc :q!
* **Exit with save data :** Esc :wq!

**Install all modules:**

**pip install -r requirements.txt**

**UnInstall all modules:**

**pip uninstall -r requirements.txt -y**

**Snippet****Install python(pydev) in vs code..**

**urls.py:**

from django.contrib import admin

from django.urls import path

from django.urls.conf import include

from django.views.generic.base import RedirectView

urlpatterns = [

    path('admin/', admin.site.urls),

    path('home/', include('social.urls')),

    path('',RedirectView.as\_view(url='home/')),

]

**Views.py**

from django.urls import path

from social import views

urlpatterns = [

    path('',views.home),

]

**HttpResponse**

**Views.py:**

from django.shortcuts import render

from django.http.response import HttpResponse

from django.http import request

def home(request):

    return HttpResponse("Hello Pradip")

**context**

**views.py:**

from django.shortcuts import render

# from django.http.response import HttpResponse

from django.http import request

def home(request):

    fname = {"name":"pradip","surname":"kachhadiya"}

    return render(request,'home.html',fname)

**home.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Home page</title>

</head>

<body>

    <h2>hello {{name}}</h2>

    <h2>hello {{surname}}</h2>

</body>

</html>

**Template outside application**

**Settings.py:**

TEMPLATES\_DIR = os.path.join(BASE\_DIR,'templates')

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [TEMPLATES\_DIR],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

**Dj/urls.py:**

from django.contrib import admin

from django.urls import path

from django.urls.conf import include

from django.views.generic.base import RedirectView

urlpatterns = [

    path('admin/', admin.site.urls),

    path('social/', include('social.urls')),

    path('course/', include('course.urls')),

    path('',RedirectView.as\_view(url='social/')),

]

**course/urls.py:**

from django.urls import path

from course import views

urlpatterns = [

    path('index/',views.home),

]

**Views.py:**

from django.shortcuts import render

def home(req):

    return render(req,'course/home.html')

**social/urls.py:**

from django.urls import path

from social import views

urlpatterns = [

    path('home/',views.home),

]

**Views.py:**

from django.shortcuts import render

# from django.http.response import HttpResponse

from django.http import request

def home(request):

    return render(request,'social/index.html')

**Static File**

**Settings.py:**

STATIC\_DIR = os.path.join(BASE\_DIR,'static')

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [STATIC\_DIR]

**Home.html:**

<!DOCTYPE html>

{% load static %}

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="{% static 'course/style.css' %}"/>

    <title>home</title>

</head>

<body>

    <h2>hello word</h2>

</body>

</html>

**Url shortcutt**

**urls.py:**

 path('about/',views.about,name="ab"),

**home.html:**

 <a href="{% url 'ab' %}">about</a>

**Include method (Joint Another html)**

**Home.html:**

{% include 'course/best.html' %}

{% include 'course/best.html' with p="hello" %}

**ORM(OBJECT RELATION MODEL)**

1. Makemigration :: It’s creat sql query
2. Migrate :: It’s execute sql query

**INSTALL SQL DATABASE**

* Load db

1. **GET DATA FROME DATABASE**

**Views.py:**

def student(req):

    st = Student.objects.all()

    return render(req,'course/home.html',{"st":st})

**models.py:**

from django.db import models

class Student(models.Model):

    name = models.CharField(max\_length=20)

    roll\_no = models.IntegerField()

    email = models.EmailField(max\_length=20)

    stupass = models.CharField(max\_length=20)

    comment = models.CharField(max\_length=10)

    def \_\_str\_\_(self):

        return "%s" %(self.name)

1. **Register the Model**

**Admin.py:**

admin.site.register(Student,StudentAdmin)

**OR (BEST-PRACTICE)**

@admin.register(Student,StudentAdmin)

1. **ADD ADITIONAL FETURES**

**Admin.py:**

from django.contrib import admin

from course.models import Student

class StudentAdmin(admin.ModelAdmin):

    list\_display= ["name","email","id"]

    search\_fields=["name","id"]

    list\_filter = ["name","email"]

list\_display\_links = ('name', 'eamil')

admin.site.register(Student,StudentAdmin)

**DJANGO FORM**

**Forms.py:**

from django import forms

class StudentRegistration(forms.Form):

    name  = forms.CharField()

    roll\_no = forms.CharField()

**views.py:**

from course.forms import StudentRegistration

from django.http import request

def show(request):

    fm = StudentRegistration()

    return render(request,"course/home.html",{"fm":fm})

**urls.py:**

from django.urls import path

from course import views

urlpatterns = [

    path('about/',views.show,name="ab"),

]

**Home.html:**

{% extends 'base.html' %}

{% block title %}home page{% endblock %}

{% block content %}

<form action="">

<h1>{{fm.as\_p}}</h1>

<input type="submit" value="press">

</form>

{% endblock %}

* **0r  Fm.as\_table**

**Form Validation**

**Forms.py:**

from django import forms

class StudentRegistration(forms.Form):

    name  = forms.CharField()

    roll\_no = forms.CharField()

**views.py:**

from django.shortcuts import render

from course.forms import StudentRegistration

from django.http import request

def show(request):

    if request.method == "POST":

        fm=StudentRegistration(request.POST)

        if fm.is\_valid():

            print(fm.cleaned\_data['name'])

            fm = StudentRegistration()

        print("post")

    else:

        print("get")

        fm = StudentRegistration()

    return render(request,'course/home.html',{"fm":fm})

**home.html:**

{% extends 'base.html' %}

{% block title %}home page{% endblock %}

{% block content %}

<form action=""  method="POST">

{% csrf\_token %}

<h1>{{fm.as\_p}}</h1>

<input type="submit" value="press">

</form>

{% endblock %}

**FORM FIELD**

**Forms.py:**

from django import forms

class StudentRegistration(forms.Form):

    name  = forms.CharField(max\_length=5,min\_length=3)

    roll\_no = forms.CharField()

**check box:**

 agree = forms.BooleanField(label="I agree")

**Integer field:**

 roll = forms.IntegerField(max\_value=15)

**validators:**

from django import forms

from django.core import validators

class StudentRegistration(forms.Form):

    name  = forms.CharField(validators=[validators.MaxLengthValidator(4)])

    roll\_no = forms.CharField(validators=[validators.MinLengthValidator(5,'enter true evalue')])

**Httpresponsredirect**

**Views.py:**

from django.shortcuts import render

from course.forms import StudentRegistration

from django.http.response import HttpResponseRedirect

def show(request):

    if request.method == "POST":

        fm=StudentRegistration(request.POST)

        if fm.is\_valid():

            name = fm.cleaned\_data['name']

            password = fm.cleaned\_data['password']

            re\_pass = fm.cleaned\_data['re\_enter\_pass']

            print(re\_pass)

            # if password != re\_pass:

            #     raise ValueError("Not match pass...")

            return HttpResponseRedirect("/course/success")

    else:

        print("get")

        fm = StudentRegistration()

    return render(request,'course/home.html',{"fm":fm})

def success(request):

    return render(request,"course/success.html")

**Clean Data(Compaire two field)**

from django import forms

from django.core import validators

class StudentRegistration(forms.Form):

    name  = forms.CharField(validators=[validators.MaxLengthValidator(4)])

    roll\_no = forms.CharField(validators=[validators.MinLengthValidator(5,'enter true evalue')])

    password = forms.CharField(widget=forms.PasswordInput)

    re\_enter\_pass = forms.CharField(widget=forms.PasswordInput)

    def clean(self):

        cleaned\_data = super().clean()

        pass\_w = self.cleaned\_data['password']

        re\_pass = self.cleaned\_data['re\_enter\_pass']

        if pass\_w != re\_pass:

            raise forms.ValidationError("Passwod no match")

**Error Massage**

<form action=""  method="POST" novalidate>

**Forms.py:**

name  = forms.CharField(error\_messages={'required':"enter name"})

**style error msg:**

<style>

    .errorlist{

        list-style-type: none;

        color: red;

    }

**Save Form Data in Database**

st = Student.objects.create(name=nm,roll\_no=roll,password=ps,email=em)

st.save()

**OR**

st = Student(name=nm,roll\_no=roll,password=ps,email=em)

st.save()

**Use Model Form**

**Forms.py:**

from django import forms

from course.models import Student

class Studentregistration(forms.ModelForm):

    class Meta:

        model = Student

        fields = "\_\_all\_\_"

exclude = [‘name’] name sivay nu badhu dekhashe

* baki badhu same as abov normal form

**Use Extra Features In ModelForm:**

labels = {'name':"Username"}

error\_messages = {

            'name':{'required':"Enter name"},

            'password':{'required':"Enter password"}

            }

widgets = {

           'password':forms.PasswordInput(attrs={'placeholder':"Password"})

           }

**Dynemic Urls**

**Three Types :**

1. **<str>  Passing the string**
2. **<int:pk>  Passing the integer key**
3. **<slug>  Passing any value like(\_,-,pradip\_kachhadiya etc)**

**EX.**

**Urls.py:**

from django.urls import path

from django.views.generic.base import RedirectView

from college import views

urlpatterns = [

    path('home/',views.home,name="home"),

    path('show/<int:pk>/',views.show,name="show"),

    path('',RedirectView.as\_view(url = 'home/')),

]

**Views.py:**

from django.shortcuts import render

def home(request):

     return render(request,"course/home.html")

def show(request,pk):

    return render(request,"course/show.html",{"pk":pk})

**Fatch data from database:**

* **St = Student.objects.create(…….)**

**St.save()**

* **St = Student.objects.get(pk=pk)**

**St.delete()**

**Messages In Django**

**Views.py:**

* **From Django.contrib import messages**
* **Fm.save() ni niche lakhavu**
* **messages.add\_message(request,messages.SUCCESS, ‘Your account has been created’)**

**Register.html:**

* **</form> ni niche lakhavu**
* **{% if messages %}**
* **{% for message in messages%}**
* **<span class=“success” >{{message}}</span> (Give css using class)**
* **{% endfor %}**
* **{% endif %}**

**Sign-Up Page(User Registration)**

**Forms.py:**

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

class Studentregistrtion(UserCreationForm):

    class Meta:

        model = User

        fields = ["username","first\_name","last\_name","email"]

        labels = {'email':'Email'}

**Views.py:**

from django.shortcuts import render

from roll.forms import Studentregistrtion

def sign\_up(request):

    if request.method=="POST":

        fm = Studentregistrtion(request.POST)

        if fm.is\_valid():

            fm.save()

    else:

        fm = Studentregistrtion()

    return render(request,'show.html',{"fm":fm})

**home.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>show</title>

</head>

<body>

    <form action="" method="POST">

        {% csrf\_token %}

    <h1>{{fm.as\_p}}</h1>

    <button>Submit</button>

</form>

</body>

</html>

**Login Page(User login)**

**Note :- If any error occurred for this login page code…please follow logout page code(See this code login page below)..They include all login page code.**

**Urls.py:**

from django.urls import path

from roll import views

from django.views.generic.base import RedirectView

urlpatterns = [

        path('sign\_up/',views.sign\_up),

        path('login/',views.login),

        path('profile/',views.profile),

        path('',RedirectView.as\_view(url='sign\_up/')),

]

**Views.py:**

from django.shortcuts import render

from roll.forms import Studentregistrtion

from django.contrib.auth.forms import AuthenticationForm

from django.contrib.auth import authenticate

from django.http.response import HttpResponseRedirect

def login(request):

    if request.method == "POST":

        fm = AuthenticationForm(request=request,data=request.POST)

        if fm.is\_valid():

            uname = fm.cleaned\_data["username"]

            pw = fm.cleaned\_data["password"]

            user = authenticate(username=uname,passsword=pw)

            if user is not None:

                login(request,user)

            return HttpResponseRedirect("/roll/profile/")

    else:

        fm = AuthenticationForm()

    return render(request, "login.html",{"fm":fm})

def profile(request):

    return render(request,"profile.html")

**login.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Login</title>

</head>

<body>

    <form action="" method="POST">

        {% csrf\_token %}

    {{fm.as\_p}}

    <input type="submit" value="Submit">

    </form>

    <a href="/roll/sign\_up/">Signup</a>

</body>

</html>

**As it is create profile.html…..**

**Logout Page**

**Urls.py:**

from django.urls import path

from . import views

from django.views.generic.base import RedirectView

urlpatterns = [

        path('sign\_up/',views.sign\_up),

        path('login\_user/',views.login\_user),

        path('logout\_user/',views.logout\_user),

        path('profile/',views.profile),

        path('changepassword/',views.changepassword,name="cp"),

        path('',RedirectView.as\_view(url='sign\_up/')),

]

**Views.py:**

from django.shortcuts import render

from .forms import SignUpForm

from django.contrib.auth.forms import AuthenticationForm, PasswordChangeForm

from django.contrib.auth import authenticate, login,logout,\

    update\_session\_auth\_hash

from django.http.response import HttpResponseRedirect

from django.contrib import messages

def sign\_up(request):

    if request.method=="POST":

        fm = SignUpForm(request.POST)

        if fm.is\_valid():

            fm.save()

            return HttpResponseRedirect("/roll/login\_user/")

    else:

        fm = SignUpForm()

    return render(request,'signup.html',{"fm":fm})

def login\_user(request):

    if not request.user.is\_authenticated:

        if request.method == "POST":

            fm = AuthenticationForm(request=request,data=request.POST)

            if fm.is\_valid():

                uname = fm.cleaned\_data['username']

                upw = fm.cleaned\_data['password']

                user = authenticate(request=request,username=uname,password=upw)

                if user is not None:

                    login(request,user)

                    return HttpResponseRedirect("/roll/profile/")

                else:

                    print("Locha he bhai isme")

        else:

            fm = AuthenticationForm()

        return render(request, "login.html",{"fm":fm})

    else:

        return HttpResponseRedirect("/roll/profile/")

def profile(request):

    if request.user.is\_authenticated:

        return render(request,"profile.html",{"us":request.user})

    else:

        return HttpResponseRedirect("/roll/login\_user/")

def logout\_user(request):

    logout(request)

    return HttpResponseRedirect("/roll/login\_user/")

**profile.py:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Profile</title>

    <style>

        .success{

            color: green;

        }

    </style>

</head>

<body>

    {% if messages %}

    {% for message in messages %}

       <span class="success">{{message}}</span>

    {% endfor %}

    {% endif %}

    <h2>I am {{us}}</h2>

    <a href="/roll/logout\_user">Logout</a>

    <a href="{% url 'cp' %}">Change Password</a>

</body>

</html>

**Change Password**

**Urls.py:**

* See above urls.py…..that is same

**Views.py:**

def changepassword(request):

    if request.user.is\_authenticated:

        if request.method == "POST":

            fm = PasswordChangeForm(user=request.user,data=request.POST)

            if fm.is\_valid():

                fm.save()

                update\_session\_auth\_hash(request,fm.user)

                messages.success(request,"You changed password Successfully ")

                return HttpResponseRedirect("/roll/profile/")

        else:

            fm = PasswordChangeForm(user=request.user)

    else:

        return HttpResponseRedirect("/roll/login\_user/")

    return render(request,"changepass.html",{"fm":fm})

**changepassword.html :**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>changepass</title>

</head>

<body>

    <h1>Change Password</h1>

    <form action="" method="POST">

        {% csrf\_token %}

        {{fm.as\_p}}

        <input type="Submit" value="Submit">

    </form>

</body>

</html>

**Profile.html:**

* **See above code…..**

**SetPassword Form**

**Views.py:**

* Same as above code but here change only write SetPasswordForm instead of PasswordChange Form.

**Use :: Not required input old password.**

**Profile(User Profile)**

**Profile.py:**

from django.contrib.auth.forms import UserCreationForm, UserChangeForm

class UserProfileForm(UserChangeForm):

    password = None

    class Meta:

        model = User

        fields = ["username","first\_name","last\_name","email","last\_login","date\_joined"]

        labels = {'email':'Email'}

**views.py:**

from .forms import SignUpForm,UserProfileForm

def profile(request):

    if request.user.is\_authenticated:

        if request.method=="POST":

            fm = UserProfileForm(request.POST,instance=request.user)

            if fm.is\_valid():

                fm.save()

                messages.success(request,"Profile Updated!!!")

                return HttpResponseRedirect("/roll/profile/")

        else:

            fm = UserProfileForm(instance=request.user)

        return render(request,"profile.html",{"us":request.user,"fm":fm})

    else:

        return HttpResponseRedirect("/roll/login\_user/")

**Profile.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Profile</title>

    <style>

        .success{

            color: green;

        }

    </style>

</head>

<body>

    {% if messages %}

    {% for message in messages %}

       <span class="success">{{message}}</span>

    {% endfor %}

    {% endif %}

    <form action="" method="POST">

        {% csrf\_token %}

        {{fm.as\_p}}

        <input type="submit" value="Submit">

    </form>

    <h2>I am {{us}}</h2><br><br>

    <a href="/roll/logout\_user">Logout</a><br><br>

    <a href="{% url 'cp' %}">Change Password</a>

</body>

</html>

**Admin Profile**

**Form.py:**

class AdminProfileForm(UserChangeForm):

    password = None

    class Meta:

        model = User

        fields = "\_\_all\_\_"

**views.py:**

from .forms import SignUpForm,UserProfileForm,AdminProfileForm

def profile(request):

    if request.user.is\_authenticated:

        if request.method=="POST":

            if request.user.is\_superuser:

                fm = AdminProfileForm(request.POST,instance=request.user)

            else:

                fm = UserProfileForm(request.POST,instance=request.user)

            if fm.is\_valid():

                fm.save()

                messages.success(request,"Profile Updated!!!")

                return HttpResponseRedirect("/roll/profile/")

        else:

            if request.user.is\_superuser==True:

                fm = AdminProfileForm(instance=request.user)

            else:

                fm = UserProfileForm(instance=request.user)

                return HttpResponseRedirect("/roll/login\_user/")

        return render(request,"profile.html",{"us":request.user,"fm":fm})

    else:

        return HttpResponseRedirect("/roll/login\_user/")

**profile.html:**

* same as it is above part….And do extra below

 {% if user.is\_superuser %}

    <a href="/admin">Admin</a><br><br>

    {% endif %}

**User Dashbord**

**Urls.py:**

      path('userdashbord/',views.userdashbord),

**views.py:**

def userdashbord(request):

    if request.user.is\_authenticated:

        return render(request,"dash.html",{"nm":request.user.username})

    else:

        return HttpResponseRedirect("/roll/login\_user")

**Give a permission in admin panel:**

**Make Group in admin pannel**

**Views.py:**

def signup(req):

    if req.method == "POST":

        form = SignupForm(req.POST)

        if form.is\_valid():

            user = form.save()

            group = Group.objects.get(name='Auther')

            user.groups.add(group)

            messages.success(req,"Successfully Register!!!!")

            messages.warning(req,"Please login!!!")

            return HttpResponseRedirect("/blog/user\_login")

    else:

        form = SignupForm()

    return render(req,"signup.html",{"fm":form})

**dash.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>dash</title>

</head>

<body>

    <h1>hello {{nm}}</h1>

    {% if perms.roll.delete\_modal %}

    <input type="button" value="delete">

    {% else %}

    <h1>no delete</h1>

    {% endif %}

    {% if perms.roll.add\_modal %}

    <input type="button" value="add">

    {% else %}

    <h1>no add</h1>

    {% endif %}

    <a href="/roll/logout\_user">Logout</a>

</body>

</html>

**COOKIES IN DJANGO**

**Urls.py:**

from django.urls import path

from django.views.generic.base import RedirectView

from . import views

urlpatterns = [

    path('set/', views.set\_cook),

    path('get/', views.get\_cook),

    path('del/', views.del\_cook),

]

**Views.py:**

from django.shortcuts import render

from datetime import datetime, timedelta

# Create your views here.

def set\_cook(req):

    pi = render(req,"set.html")

    pi.set\_cookie("name","pradip",max\_age=60)

#expires=datetime.utcnow()+timedelta(days=1)

    return pi

def get\_cook(req):

    #pi.COOKIES["name"]        #this return error when cookies are not in session

    pi = req.COOKIES.get("name","guest")   #this return none value when cookies are not in session

    return render(req,"get.html",{"pi":pi})

def del\_cook(req):

    pi = render(req,"del.html")

    pi.delete\_cookie("name")

    return pi

* **Here guest is default value and max age is cookies’s life time….**
* **Make set.html,get.html,del.html…..**

**Signed cookies**

* **Here content is hidden(encrypted)**

**Views.py:**

from django.shortcuts import render

from datetime import datetime, timedelta

def set\_cook(req):

    pi = render(req,"set.html")

    pi.set\_signed\_cookie("name","kachadiya",salt="nm",expires=datetime.utcnow()+timedelta(days=1))

    return pi

def get\_cook(req):

    pi = req.get\_signed\_cookie("name","guest",salt="nm")

    return render(req,"get.html",{"pi":pi})

def del\_cook(req):

    pi = render(req,"del.html")

    pi.delete\_cookie("name")

    return pi

**Session in django**

**Views.py:**

from django.shortcuts import render

def set\_session(req):

    req.session["name"]="pradip"

    req.session["sname"]="kachhadiya"

    return render(req,"set.html")

def get\_session(req):

    # name = req.session["name"]

    name = req.session.get("name",'guest')

    sname = req.session.get("sname",'guest')

    return render(req,"get.html",{"nm":name})

def del\_session(req):

    if 'name' in req.session:

        del req.session["name"]

    return render(req,"del.html")

**Other Method**

**views.py:**

from django.shortcuts import render

def set\_session(req):

    req.session["name"]="pradip"

    req.session["sname"]="kachhadiya"

    return render(req,"set.html")

def get\_session(req):

    # name = req.session["name"]

    name = req.session.get("name",'guest')

    sname = req.session.get("sname",'guest')

    key = req.session.keys()

    item = req.session.items()

    # df = req.session.setdefault("age","12")

    return render(req,"get.html",{"nm":name,"key":key,"sn":sname,"item":item})

def del\_session(req):

    req.session.flush()

    return render(req,"del.html")

**get.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    getttttt <br>{{nm}}<br>{{sn}}

    {% for key in key %}

    <h4>{{key}}</h4><hr>

    {% endfor %}

    {% for key,value in item %}

    <h4>{{key}}---{{value}}</h4><hr>

    {% endfor %}

</body>

</html>

**Expiry date clear(delete)**

**Views.py:**

from django.shortcuts import render

def set\_session(req):

    req.session["name"]="pradip"

    req.session.set\_expiry(10)

    return render(req,"set.html")

def get\_session(req):

    # name = req.session["name"]

    name = req.session.get("name",'guest')

    return render(req,"get.html",{"nm":name})

def del\_session(req):

    req.session.flush()

    req.session.clear\_expired()

    return render(req,"del.html")

**get.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    getttttt <br>{{nm}}<br>

    {{request.session.get\_session\_cookie\_age}}<br>

    {{request.session.get\_expiry\_age}}<br>

    {{request.session.get\_expiry\_date}}<br>

    {{request.session.get\_expire\_at\_browser\_close}}

</body>

</html>

**Check Cookies enable or not**

**Testing on cookies :**

**Views.py:**

from django.shortcuts import render

def set\_test(req):

    req.session.set\_test\_cookie()

    return render(req,'set.html')

def get\_test(req):

    req.session.test\_cookie\_worked()

    return render(req,'get.html')

def del\_test(req):

    req.session.delete\_test\_cookie()

    return render(req,'del.html')

**get.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

 <h1>  {{request.session.test\_cookie\_worked}}</h1>

</body>

</html>

**Cookies settings**

**Setting.py:**

SESSION\_COOKIE\_AGE = 400             #Chage cookies age time

SESSION\_COOKIE\_NAME = "sessionname"   #Change session name

SESSION\_COOKIE\_PATH = '/home'         #Change cookies path

SESSION\_COOKIE\_HTTPONLY = True   #Not acsses cookies by using javascript

SESSION\_COOKIE\_SECURE = True    #Use only HTTPS ===>>Prevent packets tarnsfer or hijecking

SESSION\_ENGINE = 'django.contrib.session.backends.file'   #Save the session in file base not save in database

SESSION\_EXPIRE\_AT\_BROWSER\_CLOSE = True   #Use in When browser close at that session is close

SESSION\_FILE\_PATH = '##'     #Give specific file path to save session

SESSION\_SAVE\_EVERY\_REQUEST = True   #Every time update session when page refersh

**Modified cookies**

**Views.py:**

def get\_session(req):

    # name = req.session["name"]

    if 'name' in req.session:

        name = req.session.get("name")

        req.session.modified = True

        return render(req,"get.html",{"nm":name})

    else:

        return HttpResponse("Your session has been expired!!!!")

**File Based cookies**

**Setting.py:**

SESSION\_ENGINE = 'django.contrib.sessions.backends.file'

SESSION\_FILE\_PATH = os.path.join(BASE\_DIR,'session')

* **Create session folder in current project….**

**Page Counter Project Using Cookies**

**Views.py:**

def home(req):

    pi = req.session.get('counter',0)

    pi += 1

    req.session['counter'] = pi

    return render(req,"home.html",{'c':pi})

**Home.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="refresh" content="0.5">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <style>

        .counter{

            background-color: rgb(231, 169, 200);

            font-weight: bold;

            font-style: italic;

            text-align: center;

            font-size: 5vw;

            text-shadow: 3px 4px 2px rgb(98, 6, 190);

            color: white;

            margin-top: 20% ;

}

    </style>

    <title>Document</title>

</head>

<body class="counter">

            Page Counte :: {{c}}

</body>

</html>

**Cache in django**

* **There are three types:**

1. Per site cache
2. Per view cache
3. Template fragment caching
4. **Per site cache**

**[A] . Store cache in databse::**

**Setting.py:**

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.cache.UpdateCacheMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.cache.FetchFromCacheMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

TIME\_ZONE = 'Asia/Kolkata'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = False

STATIC\_URL = '/static/'

CACHE\_MIDDLEWARE\_SECONDS = 10

CACHES = {

    'default':{

        'BACKEND' : 'django.core.cache.backends.db.DatabaseCache',

        'LOCATION' : 'enroll\_cache',

    }

}

**Run in terminal:-** **python manage.py createcachetable**

**Urls.py:**

from django.urls import path

from . import views

urlpatterns = [

    path('set/', views.set\_cache),

**views.py:**

from django.shortcuts import render

def set\_cache(req):

    return render(req,"set.html")

**set.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="refresh" content="0.5">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>hello1</h1>

    <h1>hello2</h1>

    <h1>hello3</h1>

    <h1>helo4</h1>

</body>

</html>

**[B] . Store cache in file::**

**Setting.py:**

* **Make the cache folder in root directry……**

STATIC\_URL = '/static/'

CACHE\_MIDDLEWARE\_SECONDS = 20

CACHES = {

    'default':{

        'BACKEND' : 'django.core.cache.backends.filebased.FileBasedCache',

        'LOCATION' : str(r'C:\Users\Pradip\Documents\Django project\dj\cache'),

    }

}

* **Baki badhu as it is above…..implimentetion……**

**[C] . Store cache in local memory**

**Setting.py:**

STATIC\_URL = '/static/'

CACHE\_MIDDLEWARE\_SECONDS = 20

CACHES = {

    'default':{

        'BACKEND' : 'django.core.cache.backends.locmem.LocMemCache',

        'LOCATION' :'unique-snowflake',

    }

}

1. **Per view cache**
2. **Store cache in database**
3. **Store cache in file**
4. **Store cache in localmemory**

**NOTE::Here use only database cache storing**

**Setting.py:**

 # 'django.middleware.cache.UpdateCacheMiddleware',

    'django.middleware.common.CommonMiddleware',

    # 'django.middleware.cache.FetchFromCacheMiddleware',

CACHE\_MIDDLEWARE\_SECONDS = 20

CACHES = {

    'default':{

        'BACKEND' : 'django.core.cache.backends.db.DatabaseCache',

        'LOCATION' : 'enroll\_cache',

    }

}

**Two type cache declair:**

1. **Views base cache**
2. **Urls base cache**
3. **Views base cache:**

**Views.py:**

from django.shortcuts import render

from django.views.decorators.cache import cache\_page

@cache\_page(30)

def set\_cache\_v(req):

    return render(req,"set.html")

def get\_cache\_v(req):

    return render(req,"get.html")

1. **Urlbase cache:**

**urls.py:**

from . import views

from django.views.decorators.cache import cache\_page

urlpatterns = [

    path('set/', views.set\_cache\_v),

    path('get/', cache\_page(20)(views.get\_cache\_v)),

    path('get2/', views.get\_cache\_v),

**[3]. Template Fragment cache**

** store cache in database or file or localmemory…..**

**NOTE : Here use only store cache in database…**

**Setting.py:**

CACHE\_MIDDLEWARE\_SECONDS = 20

CACHES = {

    'default':{

        'BACKEND' : 'django.core.cache.backends.db.DatabaseCache',

        'LOCATION' : 'enroll\_cache',

    }

}

**Urls.py:**

from django.urls import path

from . import views

urlpatterns = [

    path('set/', views.set\_cache\_v),]

**views.py:**

from django.shortcuts import render

from django.views.decorators.cache import cache\_page

def set\_cache\_v(req):

    return render(req,"set.html")

**set.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="refresh" content="0.5">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>set</title>

{% load cache %}

</head>

<body>

    <h1>hello1</h1>

    <h1>hello2</h1>

    {% cache 20 set %}

    <h1>hello3</h1>

    <h1>helo4</h1>

    {% endcache set %}

</body>

</html>

**Low level cache API**

**Urls.py:**

from django.urls import path

from . import views

urlpatterns = [

    path('set/', views.set\_cache\_v), ]

**views.py:**

from django.shortcuts import render

from django.views.decorators.cache import cache\_page

from django.core.cache import cache

def set\_cache\_v(req):

    ch = cache.get('planet',"no data")

    if ch == "no data":

        cache.set('planet','mars',20)

        ch = cache.get('planet')

    return render(req,"set.html",{"pn":ch})

**OR**

def set\_cache\_v(req):

    ch = cache.get\_or\_set('planet',"moon",10,version=2)

    return render(req,"set.html",{"pn":ch})

**not compulsory **

def set\_cache\_v(req):

    data = {"name":'pradip','roll':2001}

    cache.set\_many(data,20)

    sv = cache.get\_many(data)

    return render(req,"set.html",{"pn":sv})

**delete cache:**

 def set\_cache\_v(req):

    cache.delete('planet')

    return render(req,"set.html")

**Decrease and Increase:**

def set\_cache\_v(req):

      cache.set("roll",5,600)

    dv = cache.decr('roll',delta=1)

    print(dv)

=================================================================

    dv = cache.incr('roll',delta=2)

    print(dv)

    return render(req,"set.html")

**Clear cache:**

def set\_cache\_v(req):

    cache.set("roll",5,600)

    cache.clear()

    return render(req,"set.html")

**Signals in django**

**Step :**

1. **\_\_init\_\_.py**
2. **App.py**
3. **Signals.py**

**Login,Logout,Loginfail signals**

**[1]. Settings.py:**

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'blog'

]

**[2] \_\_init\_\_.py:**

default\_app\_config = 'blog.apps.BlogConfig'

**OR**

**Settings.py:**

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'blog.apps.BlogConfig'

]

**Here noneed to register app name in \_\_init\_\_.py…….**

**[3]. apps.py:**

from django.apps import AppConfig

class BlogConfig(AppConfig):

    name = 'blog'

    def ready(self):

        import blog.signals

**[4]. Signals.py:**

from django.contrib.auth.models import User

from django.contrib.auth.signals import user\_logged\_in, user\_logged\_out,\

    user\_login\_failed

from django.dispatch.dispatcher import receiver

@receiver(user\_logged\_in,sender=User)

def login\_user(sender,request,user,\*\*kwargs):

    print("====================  LOGIN USER  ==========================")

    print(sender)

    print(request)

    print(user.password)

    print(kwargs)

    print(f"kwargs:  {kwargs}")

# user\_logged\_in.connect(login\_user,sender=User)

@receiver(user\_logged\_out,sender=User)

def logout\_user(sender,request,user,\*\*kwargs):

    print("====================  LOGOUT USER  ==========================")

    print(sender)

    print(request)

    print(user.password)

    print(kwargs)

    print(f"kwargs:  {kwargs}")

# user\_logged\_out.connect(logout\_user,sender=User)

@receiver(user\_login\_failed)

def login\_fail(sender,request,credentials,\*\*kwargs):

    print("====================  LOGIN FAILED  ==========================")

    print(sender)

    print(request)

    print(credentials)

    print(kwargs)

    print(f"kwargs:  {kwargs}")

# user\_login\_failed.connect(login\_fail,sender=User)

**Output in terminal:**

**1). Login fail:**

**==================== LOGIN FAILED ==========================**

**django.contrib.auth**

**<WSGIRequest: POST '/admin/login/?next=/admin/'>**

**{'username': 'admin', 'password': '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'}**

**{'signal': <django.dispatch.dispatcher.Signal object at 0x000001DE097F46A0>}**

**kwargs: {'signal': <django.dispatch.dispatcher.Signal object at 0x000001DE097F46A0>}**

**[04/Aug/2020 00:11:18] "POST /admin/login/?next=/admin/ HTTP/1.1" 200 2074**

**2). Login:**

**==================== LOGIN USER ==========================**

**<class 'django.contrib.auth.models.User'>**

**<WSGIRequest: POST '/admin/login/?next=/admin/'>**

**pbkdf2\_sha256$180000$7jrMRFt1QVq4$614eNq2aiQd3OwZfA+EJWTLNChnEtAgG9uoYneLmAVI=**

**{'signal': <django.dispatch.dispatcher.Signal object at 0x000001DE097E8BE0>}**

**kwargs: {'signal': <django.dispatch.dispatcher.Signal object at 0x000001DE097E8BE0>}**

**[04/Aug/2020 00:11:24] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0**

**[04/Aug/2020 00:11:24] "GET /admin/ HTTP/1.1" 200 3042**

**3). Logout:**

**==================== LOGOUT USER ==========================**

**<class 'django.contrib.auth.models.User'>**

**<WSGIRequest: GET '/admin/logout/'>**

**pbkdf2\_sha256$180000$7jrMRFt1QVq4$614eNq2aiQd3OwZfA+EJWTLNChnEtAgG9uoYneLmAVI=**

**{'signal': <django.dispatch.dispatcher.Signal object at 0x000001DE097F45E0>}**

**kwargs: {'signal': <django.dispatch.dispatcher.Signal object at 0x000001DE097F45E0>}**

**[04/Aug/2020 00:11:26] "GET /admin/logout/ HTTP/1.1" 200 1207**

**Built-in Signals**

1. **Pre\_init**
2. **Pre\_save**
3. **Pre\_delete**
4. **Post\_init**
5. **Post\_save**
6. **Post\_delete**

* **Import module:**

from django.contrib.auth.models import User

from django.contrib.auth.signals import user\_logged\_in, user\_logged\_out,\

    user\_login\_failed

from django.dispatch.dispatcher import receiver

from django.db.models.signals import pre\_save, post\_save, pre\_delete,\

    post\_delete, pre\_init, post\_init

from django.core.signals import request\_started, request\_finished,\

    got\_request\_exception

**Pre\_save and post\_save**

**Signals.py:**

@receiver(pre\_save,sender=User)

def at\_beginning\_save(sender,instance,\*\*kwargs):

    print("==================== At Pre Save  ==========================")

    print(sender)

    print(instance)

    print(f"kwargs:  {kwargs}")

# pre\_save.connect(at\_beginning\_save,sender=User)

@receiver(post\_save,sender=User)

def at\_ending\_save(sender,instance,created,\*\*kwargs):

    print("==================== Post Save  ==========================")

    if created:

        print("=======================New Created=====================")

        print(sender)

        print(instance)

        print(created)

        print(f"kwargs:  {kwargs}")

    else:

        print("=======================Updated=====================")

        print(sender)

        print(instance)

        print(created)

        print(f"kwargs:  {kwargs}")

# post\_save.connect(at\_ending\_save,sender=User)

**================================================**

**My Cheatshit (Skip This Area)**

**[1]. college/\_\_init\_\_.py:**

default\_app\_config = 'college.apps.CollegeConfig'

1. **Jyare koi new user register karshe tyare only one time (send signal) \_\_init\_\_.py call thase and then te college.apps.CollegeConfig ne call karshe.**

**[2]. apps.py:**

from django.apps import AppConfig

class CollegeConfig(AppConfig):

    name = 'college'

    def ready(self):

        import college.mysignal

1. **ready method only one time mysignal.py ne call karshe**

**[3]. mysignal.py:**

from django.dispatch.dispatcher import receiver

from django.db.models.signals import post\_save

from django.contrib.auth.models import User

from college.models import Profile

@receiver(post\_save,sender=User)

def save\_profile(sender,instance,created,\*\*kw):

    if created:

        Profile.objects.create(user=instance)

1. **And then mysignal user ni signal ne receive karshe**
2. **@receiver(post\_save,sender=user):**
   1. **Post\_save : Te sender ni detail ne save karshe.**
   2. **Sender : Te user table ne catch karshe(singnal sender).**
3. **Instance : Tema newly created user no object aavshe.(all user detail).**
4. **Created : Te jyare new user hashe to true raheshe otherwise update hashe to false raheshe.**
5. if created:
6. Profile.objects.create(user=instance)
7. **Jo newly created hashe to teni profile banshee.**

**===========================================**

**Pre\_delete and post\_delete**

@receiver(pre\_delete,sender=User)

def at\_beginning\_delete(sender,instance,\*\*kwargs):

    print("==================== At Pre Delete  ==========================")

    print(sender)

    print(instance)

    print(f"kwargs:  {kwargs}")

# pre\_delete.connect(at\_beginning\_delete,sender=User)

@receiver(post\_delete,sender=User)

def at\_ending\_delete(sender,instance,\*\*kwargs):

    print("==================== At Post Delete  ==========================")

    print(sender)

    print(instance)

    print(f"kwargs:  {kwargs}")

# post\_delete.connect(at\_ending\_delete,sender=User)

**Pre\_init and post\_init**

@receiver(pre\_init,sender=User)

def at\_beginning\_init(sender,\*args,\*\*kwargs):

    print("==================== At Pre Inits  ==========================")

    print(sender)

    print(args)

    print(f"kwargs:  {kwargs}")

# pre\_init.connect(at\_beginning\_init,sender=User)

@receiver(post\_init,sender=User)

def at\_ending\_init(sender,\*args,\*\*kwargs):

    print("==================== At Post Inits  ==========================")

    print(sender)

    print(args)

    print(f"kwargs:  {kwargs}")

# post\_init.connect(at\_ending\_init,sender=User)

**Pre\_request and finish\_request**

@receiver(request\_started)

def at\_beginning\_request(sender,environ,\*\*kwargs):

    print("==================== At Pre Request ==========================")

    print(sender)

    print(environ)

    print(f"kwargs:  {kwargs}")

# request\_started.connect(at\_beginning\_request,sender=User)

@receiver(request\_finished)

def at\_ending\_request(sender,\*\*kwargs):

    print("==================== At Request Finishing ==========================")

    print(sender)

    print(f"kwargs:  {kwargs}")

# request\_finished.connect(at\_ending\_request,sender=User)

**Request Exception**

**When exception occurred in views.py…….**

@receiver(got\_request\_exception)

def at\_request\_exception(sender,request,\*\*kwargs):

    print("==================== At Request Exception ==========================")

    print(sender)

    print(request)

    print(f"kwargs:  {kwargs}")

# got\_request\_exception.connect(at\_request\_exception,sender=User)

**Track Client IP**

**Use in login signal….Views ma use kari shakay**

**Ip = request.META.get(‘REMOTE\_ADDR’)**

**request.session[‘ip’] = ip**

**Total Visiter(IP Address Treacker)**

**Views.py:**

    def get\_client\_ip(request):

        x\_forwarded\_for = request.META.get('HTTP\_X\_FORWARDED\_FOR')

        if x\_forwarded\_for:

            ip = x\_forwarded\_for.split(',')[-1].strip()

        elif request.META.get('HTTP\_X\_REAL\_IP'):

            ip = request.META.get('HTTP\_X\_REAL\_IP')

        else:

            ip = request.META.get('REMOTE\_ADDR')

        return ip

    ip = get\_client\_ip(request)

    v\_c = Visiter\_ip.objects.all()

    v\_count = v\_c.count()

    v\_ip = Visiter\_ip.objects.values\_list("visit\_ip",flat=True)

    if ip in v\_ip:

        pass

    else:

        v\_save = Visiter\_ip.objects.create(visit\_ip=ip)

        v\_save.save()

**login count using cache and signal**

**signal.py:**

from django.dispatch.dispatcher import receiver

from django.contrib.auth.signals import user\_logged\_in

from django.contrib.auth.models import User

from django.core.cache import cache

@receiver(user\_logged\_in,sender=User)

def login\_success(sender,request,user,\*\*kwargs):

    ct = cache.get('count',0,version=user.pk)

    newcount = ct+1

    cache.set('count',newcount,60\*60\*24,version=user.pk)

**views.py: Follow blog project**

def dashboard(req):

    if req.user.is\_authenticated:

        post = Post.objects.all()

        user = req.user

        full\_name = user.get\_full\_name()

        gps = user.groups.all()

        ct = cache.get('count',version=user.pk)

        return render(req,"dashboard.html",{"post":post,"fnm":full\_name,"gps":gps,"ct":ct})

    else:

        return HttpResponseRedirect("/blog/user\_login")

**dashboard.html:**

<button  class="btn btn-danger border mt-3">

Login Time :  <h2 class="badge badge-dark bg-dark">{{ct}}</h2>

</button>

</div>

**Custom signal**

**Signal.py:**

notification = Signal(providing\_args=["request","user"])

@receiver(notification)

def show\_notification(sender,\*\*kwargs):

    print(sender)

    print(kwargs)

    print("notification")

**view.py:**

def home(request): signals.notification.send(sender=None,request=request,user=["pradip","kachhadiya"])

return Httpresponse(“Hello Word”)

**Middleware in django**

1. **Function base :**

**middleware.py:**

def my\_middleware(get\_response):

    print("One time intialization")

    def my\_function(request):

        print("This is before view")

        response = get\_response(request)

        print("This is after view")

        return response

    return my\_function

**settings.py:**

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

    'blog.middleware.my\_middleware',]

**Views.py:**

from django.shortcuts import render

from django.http.response import HttpResponse

def home(request):

    print("I am real view")

    return HttpResponse("This is home view")

**urls.py:**

from django.contrib import admin

from django.urls import path

from blog import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('',views.home),

    path('exc/',views.exc),

]

1. **Class base middleware:**

**Middleware.py:**

from django.http.response import HttpResponse

class my\_brother\_middleware:

    def \_\_init\_\_(self,get\_response):

        self.get\_response = get\_response

        print("One time Intialization brother")

    def \_\_call\_\_(self,request):

        print("This is before brother views")

        response = self.get\_response(request)

        print("This is after brother views")

        return response

class my\_father\_middleware:

    def \_\_init\_\_(self,get\_response):

        self.get\_response = get\_response

        print("One time Intialization father")

    def \_\_call\_\_(self,request):

        print("This is before father views")

        # response = self.get\_response(request)

        response = HttpResponse("hII You Are great programmer")

        print("This is after father views")

        return response

class my\_mother\_middleware:

    def \_\_init\_\_(self,get\_response):

        print("One time Intialization mother ")

    def \_\_call\_\_(self,request):

        print("This is before mother views")

        response = self.get\_response(request)

        print("This is after mother views")

        return response

**settings.py: Here order is effect in middleware.py**

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

    # 'blog.middleware.my\_middleware',

    'blog.middleware.my\_brother\_middleware',

    'blog.middleware.my\_father\_middleware',

    'blog.middleware.my\_mother\_middleware',]

* **Views.py and urls.py is same above (func. Base views)**

**Middleware hook**

**There are three type hooks:**

**Middleware.py:**

class Myprocess:

    def \_\_init\_\_(self,get\_response):

        self.get\_response = get\_response

    def \_\_call\_\_(self,request):

        response = self.get\_response(request)

        return response

    def process\_view(request,\*args,\*\*kwargs):

        print("This is Process view - before view")

        # return HttpResponse("This is before view")

        return None

class Myexception:

    def \_\_init\_\_(self,get\_response):

        self.get\_response = get\_response

    def \_\_call\_\_(self,request):

        response = self.get\_response(request)

        return response

    def process\_exception(self,request,exception):

        msg = exception

        return HttpResponse(msg)

class Mytemplate:

    def \_\_init\_\_(self,get\_response):

        self.get\_response = get\_response

    def \_\_call\_\_(self,request):

        response = self.get\_response(request)

        return response

    def process\_template\_response(self,request,response):

        print("This is duplicate template")

        response.context\_data["name"] = "Kachhadiya"

        return response

**settings.py:**

**‘**MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

    # 'blog.middleware.my\_middleware',

    # 'blog.middleware.my\_brother\_middleware',

    # 'blog.middleware.my\_father\_middleware',

    # 'blog.middleware.my\_mother\_middleware',

    'blog.middleware.Myprocess',

    'blog.middleware.Myexception',

'blog.middleware.Mytemplate',

]

**Views.py:**

def exc(request):

    print("My exception")

    a = 10/0

    return HttpResponse("None")

def user\_info(request):

    print("I am user info view")

    context = {"name":"Pradip"}

    return TemplateResponse(request,"user.html",context)

**user.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>user</title>

</head>

<body>

    <h1>hello {{name}}</h1>

</body>

</html>

**Page under construction Project**

**Views.py:**

from django.shortcuts import render

from django.http.response import HttpResponse

def home(request):

    print("I am real view")

    return HttpResponse("This is home view")

def about(request):

    return HttpResponse("This is about page")

**Middleware.py:**

class Myprocess:

    def \_\_init\_\_(self,get\_response):

        self.get\_response = get\_response

    def \_\_call\_\_(self,request):

        response = self.get\_response(request)

        return response

    def process\_view(request,\*args,\*\*kwargs):

        print("This is duplicate view")

        return HttpResponse("<h1>This Page Is Under Construction!!!</h1>")

        # return None

* **When return is none then all views method are call!!!!**

**QuerySet API in Django**

**Return Queryset**

**Views.py:**

1. **.all():**

from django.shortcuts import render

from .models import School

# Create your views here.

def home(req):

    td = School.objects.all() # see all data raw

    return render(req,"home.html",{"td":td})

1. **.filter():**

from django.shortcuts import render

from .models import School

# Create your views here.

def home(req):

    # td = School.objects.all()

    td = School.objects.filter(marks=98) # only see 98 data raw

    return render(req,"home.html",{"td":td})

1. **.exclude():**

from django.shortcuts import render

from .models import School

# Create your views here.

def home(req):

    td = School.objects.exclude(marks=98) # 98 exclude

    return render(req,"home.html",{"td":td})

1. **.order\_by():**

def home(req):

    td = School.objects.order\_by('marks') #assending order

    td = School.objects.order\_by('-marks') #dessending order

    td = School.objects.order\_by('?')  # randomly generated

    return render(req,"home.html",{"td":td})

1. **.reverse():**

from django.shortcuts import render

from .models import School

# Create your views here.

def home(req):

    td = School.objects.order\_by('id').reverse()[:5] #see reverse 5 raw

    td = School.objects.order\_by('id').reverse() # reverse raw

    return render(req,"home.html",{"td":td})

1. **.value() and values\_list:**

from django.shortcuts import render

from .models import School

# Create your views here.

def home(req):

    #They return Dictionary

    td = School.objects.values("name","roll")   #Specific raw selection

#They return tuple

    td = School.objects.values\_list("name","roll",named=True)

    return render(req,"home.html",{"td":td})

1. **Union and Intersection and diffrence:**

* **Te two data table ne metch karshe and queryset mujab output malshe**

 td1 = School.objects.values\_list("name","roll",named=True)

    td2 = College.objects.values\_list("name","roll",named=True)

    data = td2.union(td1)

   td1 = School.objects.values\_list("name","roll",named=True)

    td2 = College.objects.values\_list("name","roll",named=True)

    data = td2.intersection(td1)

# te same data hashe tene decline karshe

td1 = School.objects.values\_list("name","roll",named=True)

    td2 = College.objects.values\_list("name","roll",named=True)

    data = td2.difference(td1)

1. **and,or operator and Q Object:**

from django.db.models.query\_utils import Q

td = School.objects.filter(Q(id=1) | Q(id=2))

    td = School.objects.filter(Q(id=1) & Q(roll=2))

    return render(req,"home.html",{"td":td})

**Not returned querysets**

1. **.get():**

def home(req):

    td = School.objects.get(pk=1)

    return render(req,"home.html",{"td":td})

1. **Latest():**

def home(req):

td = School.objects.latest("cr\_date")

return render(req,"home.html",{"td":td})

1. **Earliest():**

    td = School.objects.earliest("cr\_date")

1. **.exists():**

def home(req):

 td\_d = School.objects.all()

    td = School.objects.get(pk=1)

    print(td\_d.filter(pk=td.pk).exists())

return render(req,"home.html",{"td":td})

1. **.create():**

sd = School.objects.create(name="kp",surname="kachh..",roll=20,marks=97)

1. **.get\_or\_cerate():**

td = School.objects.get\_or\_create(name="kp",surname="kachh..",roll=20,marks=97)

1. **.update():**

**If .filter() then use this:**

td = School.objects.filter(pk=1).update(name="pradip1",roll=2004)

1. **update\_fields:**

**If .get() then use this:**

 user.save(update\_fields=['email','otp','change\_email'])

1. **.delete():**

td = School.objects.get(pk=1).delete()

# td = School.objects.all().delete()

1. **.count():**

   td = School.objects.all()

    print(td.count())

**Q Field Lookups**

def home(req):

    td = School.objects.all()

    td = School.objects.filter(name\_\_exact="pradip")

# Here i means word is not case sensitive

    td = School.objects.filter(name\_\_iexact="Pradip")

    td = School.objects.filter(name\_\_contains="a")

    td = School.objects.filter(name\_\_icontains="A")

# Only see 2001 and 2004

    td = School.objects.filter(roll\_\_in=[2001,2004])

#Greater then and or equal to

    td = School.objects.filter(marks\_\_gt=60)

    td = School.objects.filter(marks\_\_gte=10)

#Less then and or equal to

    td = School.objects.filter(marks\_\_lt=60)

    td = School.objects.filter(marks\_\_lte=60)

    td = School.objects.filter(name\_\_startswith="p")

    td = School.objects.filter(name\_\_istartswith="P")

    td = School.objects.filter(name\_\_endswith="p")

    td = School.objects.filter(name\_\_iendswith="P")

    td = School.objects.filter(id\_\_range=('1','5'))

    return render(req,"home.html",{"td":td})

**home.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Home</title>

</head>

<body>

    <table>

        <thead>

            <tr>

                <th>Id</th>

                <th>Name</th>

                <th>Surname</th>

                <th>Roll</th>

                <th>Marks</th>

                <th>Created Date</th>

            </tr>

        </thead>

        <tbody>

            {% for td in td %}

            <tr>

                <td>{{td.id}}</td>

                <td>{{td.name}}</td>

                <td>{{td.surname}}</td>

                <td>{{td.roll}}</td>

                <td>{{td.marks}}</td>

                <td>{{td.cr\_date}}</td>

            </tr>

           {% endfor %}

        </tbody>

    </table></body></html>

**Chaining Filter()**

**Models.py:**

from django.db import models

class Blog(models.Model):

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

    tagline = models.TextField()

    def \_\_str\_\_(self):

        return self.name

class Author(models.Model):

    name = models.CharField(max\_length=200)

    email = models.EmailField()

    def \_\_str\_\_(self):

        return self.name

class Entry(models.Model):

    blog = models.ForeignKey(Blog, on\_delete=models.CASCADE)

    headline = models.CharField(max\_length=255)

    body\_text = models.TextField()

    pub\_date = models.DateField()

    mod\_date = models.DateField()

    authors = models.ManyToManyField(Author)

    number\_of\_comments = models.IntegerField()

    number\_of\_pingbacks = models.IntegerField()

    rating = models.IntegerField()

    def \_\_str\_\_(self):

        return self.headline

**views.py:**

Entry.objects.filter(pub\_date\_\_year=2006)

Entry.objects.all().filter(pub\_date\_\_year=2006)

Entry.objects.filter(headline\_\_startswith='What').exclude(pub\_date\_\_gte=datetime.date.today()).filter(pub\_date\_\_gte=datetime.date(2005, 1, 30))

Blog.objects.filter(entry\_\_headline\_\_contains='Lennon')

Blog.objects.filter(entry\_\_authors\_\_name='Lennon')

Blog.objects.filter(entry\_\_authors\_\_name\_\_isnull=True)

Blog.objects.filter(entry\_\_headline\_\_contains='Lennon', entry\_\_pub\_date\_\_year=2008)

**F expression**

* Django provides F expressions to allow such comparisons. Instances of F() act as a reference to a model field within a query. These references can then be used in query filters to compare the values of two different fields on the same model instance.

**Views.py:**

from django.db.models import F

Entry.objects.filter(number\_of\_comments\_\_gt=F('number\_of\_pingbacks'))

Entry.objects.filter(rating\_\_lt=F('number\_of\_comments') + F('number\_of\_pingbacks'))

Entry.objects.filter(authors\_\_name=F('blog\_\_name'))

#The F() objects support bitwise operations by .bitand(), .bitor(), .bitxor(), .bitrightshift(), and .bitleftshift(). For example:

F('somefield').bitand(16)

**\_set relationship**

b = Blog.objects.get(id=1)

b.entry\_set.all()  # Returns all Entry objects related to Blog.

b.entry\_set.filter(headline\_\_contains='Lennon')

b.entry\_set.count()

**Many-to-many with \_set**

**Views.py:**

e = Entry.objects.get(id=3)

e.authors.all() # Returns all Author objects for this Entry.

e.authors.count()

e.authors.filter(name\_\_contains='John')

**or**

a = Author.objects.get(id=5)

a.entry\_set.all() # Returns all Entry objects for this Author.

**Aggregate**

**Views.py:**

from django.db.models.aggregates import Avg, Sum, Min, Max, Count

def home(req):

    td = School.objects.all()

    avg = td.aggregate(Avg('marks'))

    sum = td.aggregate(Sum('marks'))

    min = td.aggregate(Min('marks'))

    max = td.aggregate(Max('marks'))

    count = td.aggregate(Count('marks'))

    td = School.objects.all()[1:5]

    return render(req,"home.html",{"td":td,"avg":avg,"sum":sum,"min":min,"max":max,"count":count})

**home.html:**

<h1>Average marks :: {{avg.marks\_\_avg}}</h1>

<h1>Sum marks :: {{sum.marks\_\_sum}}</h1>

<h1>Min marks :: {{min.marks\_\_min}}</h1>

<h1>Max marks :: {{max.marks\_\_max}}</h1>

<h1>Count marks :: {{count.marks\_\_count}}</h1>

**Model Inheritance**

**Models.py:**

1. **Abstract Class:**

============================== Abstract Method ================================

class Commoninfo(models.Model):

    name = models.CharField(max\_length=20)

    age = models.IntegerField()

    date = models.DateField()

    class Meta:

        abstract = True

class Student(Commoninfo):

    fees = models.IntegerField()

    date = None

    def \_\_str\_\_(self):

        return self.name

class Teacher(Commoninfo):

    salary = models.IntegerField()

    def \_\_str\_\_(self):

        return self.name

class Contractor(Commoninfo):

    payment = models.IntegerField()

    date = models.DateTimeField()

    def \_\_str\_\_(self):

        return self.name

1. **One to One table relation:**

================= one to one model relation ===============

class Examcenter(models.Model):

    cname = models.CharField(max\_length=20)

    city = models.CharField(max\_length=25)

    def \_\_str\_\_(self):

        return self.city

class Student(Examcenter):

    name = models.CharField(max\_length=20)

    roll = models.IntegerField()

    def \_\_str\_\_(self):

        return self.name

1. **Proxy Table:**

# ===================== Proxy table ===========

class Examcenter(models.Model):

    cname = models.CharField(max\_length=20)

    city = models.CharField(max\_length=25)

class Myexamcenter(Examcenter):

    class Meta:

        proxy = True

**admin.py:**

from django.contrib import admin

from .models import Student

from .models import Examcenter

from .models import Myexamcenter

from .models import Student, Teacher, Contractor

admin.site.register(Student)

admin.site.register(Teacher)

admin.site.register(Contractor)

admin.site.register(Student)

admin.site.register(Examcenter)

admin.site.register(Examcenter)

admin.site.register(Myexamcenter)

**Change Model Manager**

* **Student.objects.all()**
* **Here model manager is “objects” to intrect with sql query**

**Models.py:**

# ================================  Change Manager =====================

class Student(models.Model):

    name = models.CharField(max\_length=20)

    age = models.IntegerField()

    student = models.Manager()

**admin.py:**

# ========================== Model Manager ==================

admin.site.register(Student)

**one to one Relationship**

**models.py:**

1. **on\_delete = models.CASCADE**

class Page(models.Model):

    user = models.OneToOneField(User,on\_delete = models.CASCADE,primary\_key=True)

    page\_name = models.CharField(max\_length=20)

    page\_cat = models.CharField(max\_length=20)

    page\_publish\_date = models.DateField()

    def \_\_str\_\_(self):

        return str(self.user)

1. **on\_delete = models.PROTECT**

class Page(models.Model):

    user = models.OneToOneField(User,on\_delete = models.PROTECT,primary\_key=True)

    page\_name = models.CharField(max\_length=20)

    page\_cat = models.CharField(max\_length=20)

    page\_publish\_date = models.DateField()

    def \_\_str\_\_(self):

        return str(self.user)

1. **limit\_choices\_to**

class Page(models.Model):

    user = models.OneToOneField(User,on\_delete = models.CASCADE,primary\_key=True,limit\_choices\_to={"is\_staff":True})

    page\_name = models.CharField(max\_length=20)

    page\_cat = models.CharField(max\_length=20)

    page\_publish\_date = models.DateField()

    def \_\_str\_\_(self):

        return str(self.user)

* **When page will delete at that time user delete…Here use signal**

**Signal.py:**

from django.dispatch.dispatcher import receiver

from django.db.models.signals import post\_delete

from .models import Page

@receiver(post\_delete,sender=Page)

def delete\_related\_user(sender,instance,\*\*kwargs):

    print("Delete User")

    print(instance)

    instance.user.delete()

**\_\_init\_\_.py:**

default\_app\_config = 'myapp.apps.MyappConfig'

**apps.py:**

from django.apps import AppConfig

class MyappConfig(AppConfig):

    name = 'myapp'

    def ready(self):

        import myapp.signal

**Many to One Relationship**

**Models.py:**

**Foreignkey**

class Page(models.Model):

    user = models.ForeignKey(to=User,on\_delete = models.CASCADE)

    #user = models.ForeignKey(to=User,on\_delete = models.PROTECT)

    #user = models.ForeignKey(to=User,on\_delete = models.SET\_NULL,null=True)

    page\_name = models.CharField(max\_length=20)

    page\_publish\_date = models.DateField()

    def \_\_str\_\_(self):

        return str(self.user)

**Many to Many Relationship**

**Models.py:**

class Song(models.Model):

    user = models.ManyToManyField(to=User)

    song\_name = models.CharField(max\_length=20)

    song\_cat = models.CharField(max\_length=20)

    song\_publish\_date = models.DateField()

    def singer\_name(self):

        return ",".join([str(i) for i in self.user.all()])

**admin.py:**

class SongAdmin(admin.ModelAdmin):

    list\_display = ["song\_name","singer\_name"]

admin.site.register(Song,SongAdmin)

**How to access all element of manytomany sield:**

 <tr class="my\_profile">

                  <td><i class="fa fa-dropbox" aria-hidden="true"></i> &nbsp; Intrested :</td>

                     <td>{{pf.interest.all|join:","}}</td>

                </tr>

**Use for loop also…..**

**Class base View**

1. **View**

**Forms.py:**

from django import forms

class Myform(forms.Form):

    name = forms.CharField(max\_length=20)

**views.py:**

from django.shortcuts import render

from django.views.generic.base import View

from django.http.response import HttpResponse

from .forms import Myform

=========================== Class Base Views =======================

=========================== [1] get request===============================

class Myview(View):

    def get(self,request):

        cn  = {"name":"pradip"}

        return render(request,"home.html",cn)

===================== [2] Post request =====================

class MyForm(View):

    def get(self,request):

        fm = Myform()

        return render(request,"home.html",{"fm":fm})

    def post(self,request):

        fm = Myform(request.POST)

        if fm.is\_valid():

            print("Your form is successfully submitted")

        return HttpResponse("Thank you for submitted!!!")

**home.html:**

    <form action="" method="POST">

        {% csrf\_token %}

    <h2>{{fm}}</h2>

    <input type="submit" value="submit">

1. **TemplateView**

**Views.py:**

# ========================== Template View =======================

class Mytemplateview(TemplateView):

    template\_name = "home.html"

    def get\_context\_data(self,\*\*kwargs):

        context = super().get\_context\_data(\*\*kwargs)

        context['name'] = "Pradip"

        context['surname'] = "Kachhadiya"

        return context

**home.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>home</title>

</head>

<body>

    <h1>This is template view</h1>

    <h2>Name :: {{name}}</h2>

    <h2>Surname :: {{surname}}</h2>

</form>

</body>

</html>

1. **Redirectview**

**Views.py:**

from django.urls import path

from . import views

from django.views.generic.base import RedirectView

urlpatterns = [

    # path('',views.Myview.as\_view()),

    # path('form/',views.MyForm.as\_view()),

    path('temp/',views.Mytemplateview.as\_view()),

    path('',RedirectView.as\_view(url = "temp/"))

]

**Generic Display View**

1. **Listview:**

**Models.py:**

rom django.db import models

class School(models.Model):

    name = models.CharField(max\_length=20)

    roll = models.IntegerField()

    course = models.CharField(max\_length=30)

    def \_\_str\_\_(self):

        return str(self.name)

**admin.py:**

from django.contrib import admin

from .models import School

admin.site.register(School)

**urls.py:**

from django.urls import path

from . import views

urlpatterns = [

    path('', views.Studentlistview.as\_view()),

]

**Views.py:**

from django.views.generic.list import ListView

from .models import School

=========================================== List view ===========================================

class Studentlistview(ListView):

    model = School

**Customization in listview:**

class Studentlistview(ListView):

    model = School

    template\_name = 'school/school.html'

    template\_name\_suffix = '\_list'

    ordering = ["name"]

    context\_object\_name = "st" #Change in template’s context itreter

    def get\_queryset(self):

      return School.objects.filter(course="mech")

    def get\_template\_names(self):

      if self.request.user.is\_superuser:

        template\_name = "school/superuser.html"

      else:

        template\_name = self.template\_name

      return [template\_name]

**school\_list.py:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1><strong>This is listview</strong> </h1>

    {% for dt in school\_list %}

     <h1>{{dt.name}}

     {{dt.roll}}

     {{dt.course}}

    </h1>

    {% endfor %}

</body>

</html>

**Superuser.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>super</title>

</head>

<body>

    <h1>This is super user page</h1>

</body>

</html>

1. **Detailview:**

**Urls.py:**

from django.urls import path

from . import views

urlpatterns = [

    path('list/', views.Studentlistview.as\_view()),

    path('detail/<int:pk>', views.Studentdetail.as\_view()),

]

**Views.py:**

from django.views.generic.detail import DetailView

class Studentdetail(DetailView):

  model = School

# ============= Customization ==============

class Studentdetail(DetailView):

    model = School

    template\_name = 'school/school.html'

    context\_object\_name = "st"

    pk\_url\_kwarg = "id"

**school\_detail:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Detail</title>

</head>

<body>

    <h1>This is detail view</h1>

    {{school.id}}

    {{school.name}}

    {{school.roll}}

    {{school.course}}

    <br><br>

    <a href="/list">Back</a>

</body>

</html>

**Generic Editing View**

1. **Form view:**

**Forms.py:**

from django import forms

class Myform(forms.Form):

    name = forms.CharField(max\_length=20)

    email = forms.EmailField()

    msg = forms.CharField(widget=forms.Textarea)

**views.py:**

from django.views.generic.edit import FormView

from .forms import Myform

from django.views.generic.base import TemplateView

from django.http.response import HttpResponse

class My\_form(FormView):

  template\_name = 'school/contact.html'

  form\_class = Myform

  success\_url = '/thanks/'

  # =========================== Aditional feature =====================

  def form\_valid(self,form):

    print(form.cleaned\_data["name"])

    print(form.cleaned\_data["email"])

    print(form.cleaned\_data["msg"])

    return HttpResponse("Thank You For register bro!!!!!!!!!")

class Thanks(TemplateView):

    template\_name = 'school/thanks.html'

**contact.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>form</title>

</head>

<body>

    <form action="" method="POST" novalidate>

        {% csrf\_token %}

        {{form.as\_p}}

        <input type="submit" value="Submit">

    </form>

</body>

</html>

1. **CreateView:**

**urls.py:**

    path('create/', views.Formcreate.as\_view(),name = "formcreate"),

**views.py:**

from django.views.generic.edit import FormView, CreateView, UpdateView

class Formcreate(CreateView):

  model = School

  fields = ["name","roll","course"]

  success\_url  = '/thanks/'

**school\_form.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>form\_create</title>

</head>

<body>

    <h1>Create Form</h1>

    <form action="" method="POST">

        {% csrf\_token %}

        {{form.as\_p}}

        <input type="submit" value="Submit">

    </form>

</body>

</html>

**Models.py:  This is additional feature**

class School(models.Model):

    name = models.CharField(max\_length=20)

    roll = models.IntegerField()

    course = models.CharField(max\_length=30)

    def \_\_str\_\_(self):

        return str(self.name)

    def get\_absolute\_url(self):

        return reverse("formcreate")

* **Give a css in form using below method…..**

**Views.py:**

class Formcreate(CreateView):

  model = School

  fields = ["name","roll","course"]

  success\_url  = '/thanks/'

  def get\_form(self):

    form = super().get\_form()

    form.fields["name"].widget = forms.TextInput(attrs={'class':'myclass'})

    form.fields["roll"].widget = forms.NumberInput(attrs={'class':'myint'})

    form.fields["course"].widget = forms.TextInput(attrs={'class':'mycs'})

    return form

1. **Updateview:**

**Views.py:**

from django.views.generic.edit import FormView, CreateView, UpdateView

class Formupdate(UpdateView):

  model = School

  fields = '\_\_all\_\_'

  success\_url = '/thanks/'

  # =================== For apply css =================

  def get\_form(self):

    form = super().get\_form()

    form.fields["name"].widget = forms.TextInput(attrs={'class':'myclass'})

    form.fields["roll"].widget = forms.NumberInput(attrs={'class':'myint'})

    form.fields["course"].widget = forms.TextInput(attrs={'class':'mycs'})

    return form

**urls.py:**

path('update/<int:pk>', views.Formupdate.as\_view(),name = "formupdate"),

**school\_form.html: Same as create views**

1. **Delete View:**

**Views.py:**

class Formdelete(DeleteView):

  model = School

  success\_url = '/list/'

**urls.py:**

    path('delete/<int:pk>', views.Formdelete.as\_view(),name = "formdelete"),

**school\_comfirm\_delete.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>delete</title>

</head>

<body>

    <h1>Are You Sure!!!!!</h1>

    <form action="" method="POST">

        {% csrf\_token %}

        <input type="submit" value="Yes">

        <a href="/list">Cancel</a>

    </form>

</body>

</html>

**Authentication View**

**Django Bydefault provide this view…….**

1. **Function base view**

**Urls.py:**

from django.contrib import admin

from django.urls import path

from django.urls.conf import include

from college import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('accounts/', include('django.contrib.auth.urls')),

    path('accounts/profile/', views.profile),

    path('',views.home),

]

**Views.py:**

from django.shortcuts import render

def profile(req):

    return render(req,"registration/profile.html")

def home(req):

    return render(req,"registration/home.html")

**home.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>home</title>

</head>

<body>

    <h1>

       <a href="/accounts/login/">login</a>  <br>

       <a href="/accounts/logout/">logout</a><br>

       <a href="/accounts/password\_change/">password\_change</a>  <br>

       <a href="/accounts/password\_reset/">password\_reset</a>  <br>

       <a href="/accounts/profile/">profile</a></h1><br>

</body>

</html>

**Login.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Login</title>

</head>

<body>

    <h1>Login Page</h1>

    <form action="" method="POST">

        {% csrf\_token %}

        {{form.as\_p}}

        <input type="submit" value="Submit">

    </form>

</body>

</html>

**Profile.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Profile</title>

</head>

<body>

    <h1>My Profile</h1>

    <br>

    <a href="/">Home</a>

</body>

</html>

**Settings.py: See the reset password link in consol at only devlopment mode**

EMAIL\_BACKEND = 'django.core.mail.backends.console.EmailBackend'

# LOGIN\_REDIRECT\_URL = '/'

**Login Required:**

**Views.py:**

from django.shortcuts import render

from django.contrib.auth.decorators import login\_required

from django.contrib.admin.views.decorators import staff\_member\_required

@login\_required

def profile(req):

    return render(req,"registration/profile.html")

# @staff\_member\_required

# def profile(req):

#     return render(req,"registration/profile.html")

1. **Class base view**

**views.py:**

from django.contrib.admin.views.decorators import staff\_member\_required

from django.views.generic.base import TemplateView

from django.utils.decorators import method\_decorator

@method\_decorator(login\_required,name='dispatch')

class Myprofile(TemplateView):

    template\_name = 'registration/profile.html'

# @method\_decorator(staff\_member\_required,name='dispatch')

# class Myprofile(TemplateView):

#     template\_name = 'registration/profile.html'

def home(req):

    return render(req,"registration/home.html")

**urls.py:**

from django.contrib import admin

from django.urls import path

from django.urls.conf import include

from college import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('accounts/', include('django.contrib.auth.urls')),

    path('accounts/profile/', views.Myprofile.as\_view()),

    path('',views.home),

]

**Customiz Authentication View**

**Settings.pt:**

EMAIL\_BACKEND = 'django.core.mail.backends.console.EmailBackend'

LOGIN\_URL = '/login/'

LOGIN\_REDIRECT\_URL = '/dashboard/'

# LOGOUT\_REDIRECT\_URL = '/'

**Forms.py:**

from django.contrib.auth.forms import AuthenticationForm, UsernameField

from django import forms

from django.utils.translation import gettext, gettext\_lazy as \_

class Loginform(AuthenticationForm):

    username = UsernameField(widget=forms.TextInput(attrs={'autofocus':True,'class':'myuser'}))

    password = forms.CharField(

        label= \_("password"),

        strip = False,

        widget=forms.PasswordInput(attrs={'autocomplete':'curent-password','class':'mypass'}),

    )

**Urls.py:**

from django.contrib import admin

from django.urls import path

from django.urls.conf import include

from college import views

from django.contrib.auth import views as auth\_view

from django.views.generic.base import TemplateView

from college.forms import Loginform

urlpatterns = [

   path('admin/', admin.site.urls),

   path('',views.home),

   path('profile/', views.Myprofile.as\_view()),

   path('dashboard/',TemplateView.as\_view(template\_name='registration/dashboard.html')),

   path('login/',auth\_view.LoginView.as\_view(template\_name='registration/login.html',authentication\_form=Loginform)),

   path('logout/',auth\_view.LogoutView.as\_view(template\_name='registration/logout.html')),

   path('changepass/',views.Mychangepass.as\_view()),

   path('changepassdone/',auth\_view.PasswordResetDoneView.as\_view(template\_name='registration/changepassdone.html')),

]

**Views.py: or You can write this code in urls.py……**

class Mychangepass(PasswordChangeView):

    template\_name='registration/changepass.html'

    success\_url='/changepassdone/'

**login.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Login</title>

    <style>

        .myuser{

            color: green;

        }

        .mypass{

            color: red;

        }

    </style>

</head>

<body>

    <h1>Login Page</h1>

    <form action="" method="POST" novalidate>

        {% csrf\_token %}

        {{form.as\_p}}

        <input type="submit" value="Submit">

    </form>

</body>

</html>

**Make a templates for your required……**

**Database Settings**

**Settings.py: [Use Mysql,Oracle,Postgres,etc…..]**

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.mysql',

        'NAME': 'database\_name',

        'USER' : 'root',

        'PASSWORD' : 'password',

        'HOST' : 'localhost',       # optional

        'PORT': 'port\_number' # optional

    }

}

**Reset forgotten Password in postgresql**

1. Go to program file
2. Path : C:\Program Files\PostgreSQL\13\data
3. open **pg\_hba** in notpad
4. at bottom replace this :

# TYPE DATABASE USER ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all all trust

# IPv4 local connections:

host all all 127.0.0.1/32 trust

# IPv6 local connections:

host all all ::1/128 trust

# Allow replication connections from localhost, by a user with the

# replication privilege.

local replication all trust

host replication all 127.0.0.1/32 trust

host replication all ::1/128 trust

1. open run terminal..
2. write this command : **services.msc**
3. restart postgresql-x64-13-pos….(right click and restart it)
4. open databases/postgres and press **query tool** which is located at tool section.
5. Write this command **ALTER USER postgres WITH PASSWORD ‘strong\_password’.**
6. Then follow 1 to 7 step again and correct **pg\_hba** filewith replace **trust** with **scram-sha-256.**

**Paginator**

**Views.py: [Function base view]**

from django.shortcuts import render

from .models import Post

from django.core.paginator import Paginator

def all\_post(request):

    ps = Post.objects.all().order\_by('id')

    # print("all page::",ps)

    paginator = Paginator(ps,3,orphans=2)

    page\_number = request.GET.get('page')

    # print("Page number :: ",page\_number)

    page\_obj = paginator.get\_page(page\_number)

    # print("page obj::",page\_obj)

    return render(request,'home.html',{'page\_obj':page\_obj})

**Views.py: [Class base view]**

from .models import Post

from django.core.paginator import Paginator

from django.views.generic.list import ListView

from django.http.response import Http404

from django.views.generic.detail import DetailView

class Allpost(ListView):

    model = Post

    template\_name = 'home.html'

    ordering = ['id']

    paginate\_by = 3

    paginate\_orphans = 1

   # def get\_context\_data(self,\*args,\*\*kwargs):

        try:

            return super(Allpost,self).get\_context\_data(\*args,\*\*kwargs)

        except Http404:

            self.kwargs['page'] = 1

            return super(Allpost,self).get\_context\_data(\*args,\*\*kwargs)

    def paginate\_queryset(self,queyrset,page\_size):

        try:

            return super(Allpost,self).paginate\_queryset(queyrset,page\_size)

        except Http404:

            self.kwargs['page'] = 1

            return super(Allpost,self).paginate\_queryset(queyrset,page\_size)

class Allpostdetail(DetailView):

    model = Post

    template\_name = 'detail.html'

**urls.py:**

from django.contrib import admin

from django.urls import path

from blog import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', views.all\_post),

    path('', views.Allpost.as\_view()),

    path('detail/<int:pk>', views.Allpostdetail.as\_view()),

]

**Home.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Home</title>

</head>

<body>

    <center><h1>Home Page</h1></center>

{% if is\_paginated %}

    {% for post in page\_obj %}

    <h2>{{post.id}}). {{post.title}}</h2>

    <p>{{post.descSU|truncatewords:1}}<a href="/detail/{{post.id}}">Read More</a></p>

    <small>{{post.cr\_date}}</small>

    {% endfor %}

<br><br>

<center>

{% if page\_obj.has\_previous %}

   <a href="?page={{page\_obj.previous\_page\_number}}">Previous</a>

{% endif %}

<span>page:{{page\_obj.number}}of {{page\_obj.paginator.num\_pages}} </span>

{% if page\_obj.has\_next %}

   <a href="?page={{page\_obj.next\_page\_number}}">Next</a>

{% endif %}

</center>

{% endif %}

</body>

</html>

**detail.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>detail</title>

</head>

<body>

    <center><h1>Detail View</h1></center>

    <h1>{{post.title}}</h1>

    <small>{{post.cr\_date}}</small>

    <p>{{post.desc}}</p>

    <button><a href="/">Back</a></button>

</body>

</html>

**Django-registration-redux**

1. **pip install django-registration-redux**

urls.py:

path('accounts/', include('registration.backends.default.urls')),

**settings.py:**

1. INSTALLED\_APPS = [
2. 'django.contrib.admin',
3. 'django.contrib.auth',
4. **'django.contrib.contenttypes',**
5. **'django.contrib.sessions',**
6. **'django.contrib.messages',**
7. **'django.contrib.staticfiles',**
8. **'social',**
9. **'registration',**
10. **'crispy\_forms',**

**ACCOUNT\_ACTIVATION\_DAYS=3**

**EMAIL\_HOST = 'smtp.gmail.com'**

**EMAIL\_HOST\_USER = 'example@gmail.com'**

**EMAIL\_HOST\_PASSWORD = 'xxxxxxxx'**

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

LOGIN\_REDIRECT\_URL='/accounts/profile/'

**Custom Urls & Forms & Views**

**Urls.py:**

from django.urls import path

from . import views

from django.urls.conf import include

from django.contrib.auth import views as auth\_view

from .forms import Loginform,Passwordresetform,PasswordchangeForm,SetpasswordForm

from django.urls.base import reverse\_lazy

from registration.backends.default import views as re

urlpatterns = [

    path('accounts/profile/',views.home),

    path('accounts/login/',auth\_view.LoginView.as\_view(template\_name='registration/login.html',authentication\_form=Loginform),name='login\_user'),

    path('accounts/password/change/',auth\_view.PasswordChangeView.as\_view(success\_url=reverse\_lazy('passdone'),template\_name='registration/password\_change.html',form\_class=PasswordchangeForm),name='pass\_change'),

    path('accounts/password/change/done/',auth\_view.PasswordChangeDoneView.as\_view(template\_name='registration/password\_done.html'),name='passdone'),

    path('accounts/password/reset/',auth\_view.PasswordResetView.as\_view(template\_name='registration/password\_reset.html',form\_class=Passwordresetform,success\_url = reverse\_lazy('reset\_send'),email\_template\_name = 'registration/password\_resetemail.html'),name='reset\_password'),

    path('accounts/password/reset/done/',auth\_view.PasswordResetCompleteView.as\_view(template\_name='registration/password\_reset\_send.html'),name='reset\_send'),

    path('accounts/password/reset/confirm/<uidb64>/<token>/',auth\_view.PasswordResetConfirmView.as\_view(template\_name='registration/password\_resetform.html',form\_class=SetpasswordForm,success\_url = reverse\_lazy('reset\_done')),name='password\_reset'),

    path('accounts/password\_reset/done/',auth\_view.PasswordResetDoneView.as\_view(template\_name='registration/password\_resetdone.html'),name='reset\_done'),

    path('accounts/activate/resend/',re.ResendActivationView.as\_view(),name='resend\_pass'),

    path('accounts/',include('registration.backends.default.urls')),

]

**Urls.py:**

from django import forms

from django.utils.translation import ugettext\_lazy as \_

from django.contrib.auth.forms import UsernameField, AuthenticationForm,\

    PasswordResetForm, PasswordChangeForm, SetPasswordForm

from django.contrib.auth import password\_validation

class Loginform(AuthenticationForm):

    username = UsernameField(widget=forms.TextInput(attrs={'placeholder':'Username','autofocus':True,'class':'text-success font-weight-bold mt-2 form-control'}),

                             error\_messages={"required":"Please Enter Username"})

    password = forms.CharField(

        label= \_("Password"),

        strip = False,

        widget=forms.PasswordInput(attrs={'placeholder':'Password','autocomplete':'curent-password','class':'login\_pass mt-2 font-weight-bold text-danger form-control'}),

        error\_messages={"required":"Please Enter Password"}

    )

class Passwordresetform(PasswordResetForm):

    email = forms.EmailField(

    label=\_("Email ID"),

    max\_length=100,

    widget=forms.EmailInput(attrs={'placeholder':'Email Address','autofocus':True,'autocomplete': 'email','class':'form-control mt-2 font-weight-bold text-success'}),

    error\_messages={"required":"Please Enter Email"}

    )

class PasswordchangeForm(PasswordChangeForm,SetPasswordForm):

    error\_messages = {

    'password\_mismatch': \_('The two password fields didn’t match.'),

    }

    new\_password1 = forms.CharField(

        label=\_("New password"),

        widget=forms.PasswordInput(attrs={'placeholder':'Password','autocomplete': 'new-password','class':'form-control text-success text-weight-bold my-2',}),

        strip=False,

        help\_text=password\_validation.password\_validators\_help\_text\_html(),

    )

    new\_password2 = forms.CharField(

        label=\_("New password confirmation"),

        strip=False,

        widget=forms.PasswordInput(attrs={'placeholder':'Confirm Password','autocomplete': 'new-password','class':'form-control text-success text-weight-bold my-2',}),

    )

    error\_messages = {

    \*\*SetPasswordForm.error\_messages,

    'password\_incorrect': \_("Your old password was entered incorrectly. Please enter it again."),

    }

    old\_password = forms.CharField(

        label=\_("Old password"),

        strip=False,

        widget=forms.PasswordInput(attrs={'placeholder':'Old Password','autocomplete': 'current-password','class':'form-control my-2 text-danger text-weight-bold','autofocus':True}),

    )

    field\_order = ['old\_password', 'new\_password1', 'new\_password2']

class SetpasswordForm(SetPasswordForm):

    error\_messages = {

        'password\_mismatch': \_('The two password fields didn’t match.'),

    }

    new\_password1 = forms.CharField(

        label=\_("New password"),

        widget=forms.PasswordInput(attrs={'placeholder':'Password','autocomplete': 'new-password','class':'form-control text-success text-weight-bold my-2'}),

        strip=False,

        help\_text=password\_validation.password\_validators\_help\_text\_html(),

    )

    new\_password2 = forms.CharField(

        label=\_("New password confirmation"),

        strip=False,

        widget=forms.PasswordInput(attrs={'placeholder':'Confirm Password','autocomplete': 'new-password','class':'form-control text-success text-weight-bold my-2'}),

    )

1. LOGIN : “/accounts/login/”
2. LOGOUT : “/accounts/logout/”
3. REGISTER : “/accounts/register/”
4. RESET PASSWORD : “/accounts/password/reset/”

**Media File**

**Settings.py:**

PROJECT\_ROOT = os.path.realpath(os.path.dirname(\_\_file\_\_))

MEDIA\_ROOT = PROJECT\_ROOT + '/media/'

MEDIA\_URL = '/media/'

**Dj/urls.py:**

from django.contrib import admin

from django.urls import path

from django.urls.conf import include

from django.conf.urls.static import static

from . import settings

urlpatterns = [

    path('admin/', admin.site.urls),

    path('',include('news.urls')),

] + static(settings.MEDIA\_URL,document\_root = settings.MEDIA\_ROOT)

**Custom login Field**

**Dj/views.py:**

from django.contrib.auth.models import User

from django.contrib.auth.backends import ModelBackend

import re

class UsernameOrEmailBackend(ModelBackend):

    def authenticate(self,request, username=None, password=None,\*\*kwargs):

        if '@' in username:

            kwargs = {'email': username}

        else:

            kwargs = {'username': username}

        try:

            user = User.objects.get(\*\*kwargs)

            if user.check\_password(password):

                return user

        except User.DoesNotExist:

            return None

    def get\_user(self, user\_id):

        try:

            return User.objects.get(pk=user\_id)

        except User.DoesNotExist:

            return None

**settings.py:**

AUTHENTICATION\_BACKENDS = (

    'dj.views.UsernameOrEmailBackend',

    'django.contrib.auth.backends.ModelBackend',

)

**Deploying app To AWS**

**Best article:**

**https://realpython.com/deploying-a-django-app-and-postgresql-to-aws-elastic-beanstalk/**

**Deploying app To Heroku**

1. **Install git and login:**

git config --global user.name "Pradip”

git config --global user.email "pradip@gmail.com"

1. Make sure your virtual environment is activated
2. **pip install gunicorn**
3. **pip install django-heroku**

IN **settings.py :**

1. import django\_heroku #At the top
2. django\_heroku.settings(locals()) # At the bottom
3. pip freeze > requirements.txt
4. Add this in settings.py:

STATIC\_ROOT = os.path.join(BASE\_DIR, 'static')

1. Make **Procfile:**

web: gunicorn myproject.wsgi

1. [Make a Heroku account](https://signup.heroku.com/)
2. [Download Heroku CLI](https://devcenter.heroku.com/articles/heroku-cli)
3. **Configuration:**

**(Followlink)**[**https://devcenter.heroku.com/articles/django-app-configuration**](https://devcenter.heroku.com/articles/django-app-configuration)

1. DEBUG = False in settings.py
2. ALLOWED\_HOSTS = ['your\_app\_name.herokuapp.com', 'localhost', '127.0.0.1'] in settings.py
3. In your terminal, type in
4. git init
5. git add .
6. git commit -m "first commit"
7. heroku login
8. heroku create app\_name
9. git push heroku master
10. heroku git:remote -a shopstarofficial (if fatal error occurred:git pull origin master -r)
11. heroku open
12. heroku run python manage.py migrate
13. PS: if Heroku isn't recognized as a command, please close your terminal and editor and then re-open it.
14. If you make edits, then just type in the terminal:

git add .

git commit -m "edit"

git push heroku master

1. heroku run python manage.py createsuperuser

**Static File Error Solution:**

1. heroku config:set DISABLE\_COLLECTSTATIC=1
2. git push heroku master
3. python manage.py collectstatic
4. python manage.py test
5. If any error occurred after running test..check your STATIC\_ROOT is correct like this ==> STATIC\_ROOT = os.path.join(BASE\_DIR, 'static').
6. After run collectstatic command check all static files are store in **static** directory for your root dir. level(manage.py dir. level)...
7. heroku run python manage.py collectstatic.
8. heroku run python manage.py migrate
9. heroku config:unset DISABLE\_COLLECTSTATIC(for future use).

**Media file serve in production:**

**In Huroku:**

1. Add **Cloudinary** Addon to your ‘**Heroku app’**.
2. click on cloudinary and install it.
3. Then click on Cloudinary addon.
4. From this dashboard you will be able to see your credentials to connect with.
5. IN your terminal type this command:

**pip install django-cloudinary-storage**

**pip install cloudinary**

**pip install Pillow**

  INSTALLED\_APPS = [

…………….

'django.contrib.staticfiles',

   'cloudinary\_storage',

  'cloudinary',

……………………

]

CLOUDINARY\_STORAGE = {

'CLOUD\_NAME': 'your\_cloud\_name',

'API\_KEY': 'your\_api\_key',

'API\_SECRET': 'your\_api\_secret'

}

MEDIA\_URL = '/media/' # or any prefix you choose

DEFAULT\_FILE\_STORAGE = 'cloudinary\_storage.storage.MediaCloudinaryStorage'

class TestModel(models.Model):

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

image = models.ImageField(upload\_to='images/', blank=True)

Now, in order to put this image into your template, you can just type:

<img src="{{ test\_model\_instance.image.url }}" alt="{{ test\_model\_instance.image.name }}">

**requirements.txt:**

**...**

**cloudinary**==**1.17.0**

**django-cloudinary-storage**==**0.2.3**

**Ck Editor Setting**

CKEDITOR\_UPLOAD\_PATH = 'uploads/'

CKEDITOR\_IMAGE\_BACKEND = "pillow"

CKEDITOR\_THUMBNAIL\_SIZE = (500, 500)

CKEDITOR\_IMAGE\_QUALITY = 40

CKEDITOR\_BROWSE\_SHOW\_DIRS = True

CKEDITOR\_JQUERY\_URL = 'http://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js'

CKEDITOR\_CONFIGS = {

    'special':

        {'toolbar': 'Special', 'height': 1000,'width':1000,

         'config.toolbarGroups':

             [

                 ['Bold', 'Image','Youtube'],

             ],

             'extraPlugins': ','.join(['codesnippet','youtube']),

         }

}

**Django defender setting**

DEFENDER\_LOGIN\_FAILURE\_LIMIT=7

DEFENDER\_LOCKOUT\_TEMPLATE='defender.html'

DEFENDER\_COOLOFF\_TIME=1200

DEFENDER\_REDIS\_URL= “redid url”

**Crispy form**

CRISPY\_TEMPLATE\_PACK = 'bootstrap4'

**Email sending**

ACCOUNT\_ACTIVATION\_DAYS=1

EMAIL\_HOST = 'smtp.gmail.com'

EMAIL\_HOST\_USER = os.environ['EMAIL\_HOST\_USER']

EMAIL\_HOST\_PASSWORD = os.environ['EMAIL\_HOST\_PASSWORD']

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

**Redirect URL**

LOGIN\_URL = '/user/accounts/login/'

LOGIN\_REDIRECT\_URL='/'

REGISTRATION\_AUTO\_LOGIN = True

**Google recaptcha**

RECAPTCHA\_PUBLIC\_KEY = os.environ['RECAPTCHA\_PUBLIC\_KEY']

RECAPTCHA\_PRIVATE\_KEY = os.environ['RECAPTCHA\_PRIVATE\_KEY']

**Social\_Auth**

# ====================== Authentication With Facebook =================

SOCIAL\_AUTH\_FACEBOOK\_KEY = os.environ['SOCIAL\_AUTH\_FACEBOOK\_KEY']

SOCIAL\_AUTH\_FACEBOOK\_SECRET = os.environ['SOCIAL\_AUTH\_FACEBOOK\_SECRET']

SOCIAL\_AUTH\_FACEBOOK\_SCOPE = [

    'email',

]

# =========================== Authentication With Google+ ===================

SOCIAL\_AUTH\_GOOGLE\_OAUTH2\_KEY = os.environ['SOCIAL\_AUTH\_GOOGLE\_OAUTH2\_KEY']

SOCIAL\_AUTH\_GOOGLE\_OAUTH2\_SECRET = os.environ['SOCIAL\_AUTH\_GOOGLE\_OAUTH2\_SECRET']

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [os.path.join(BASE\_DIR, 'templates'),],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

                'social\_django.context\_processors.backends',

                'social\_django.context\_processors.login\_redirect',

            ],

        },

    },

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

    'social\_django.middleware.SocialAuthExceptionMiddleware',

    'defender.middleware.FailedLoginMiddleware',

]

**Views.py:(Optional)**

from social\_django.models import UserSocialAuth

def Myprofile(request,pk):

    pf = Profile.objects.get(pk=pk)

    if UserSocialAuth.objects.filter(user=pf.user):

        np = True

    else:

        np=False

    return render(request,"news/myprofile.html",{"pf":pf,"np":np})

**And add below setting…..**

**Authentication Backend**

AUTHENTICATION\_BACKENDS = (

    'social\_core.backends.facebook.FacebookOAuth2',

    'social\_core.backends.google.GoogleOAuth2',

    'dj.views.UsernameOrEmailBackend',

    'django.contrib.auth.backends.ModelBackend',

)

**Security**

CSRF\_COOKIE\_SECURE = True

SESSION\_COOKIE\_SECURE = True

CSRF\_USE\_SESSIONS = True

SECURE\_HSTS\_SECONDS = 31536000

SECURE\_HSTS\_INCLUDE\_SUBDOMAINS = True

SECURE\_HSTS\_PRELOAD = True

SECURE\_CONTENT\_TYPE\_NOSNIFF=True

SECURE\_SSL\_REDIRECT = True

SECURE\_BROWSER\_XSS\_FILTER = True

**Email Change**

MANAGERS = (

    ('Pradip', 'kpunive369@gmail.com'),

)

SUPPORT\_EMAIL='stara3690@gmail.com'

**Static Files**

STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')

STATIC\_URL = '/static/'

STATICFILES\_DIRS = (

    os.path.join(BASE\_DIR, 'staticfiles'),

)

**Media Files**

# PROJECT\_ROOT = os.path.realpath(os.path.dirname(\_\_file\_\_))

# MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

DEFAULT\_FILE\_STORAGE = 'cloudinary\_storage.storage.MediaCloudinaryStorage'

MEDIA\_URL = '/media/'

**Cloudinary Setting**

CLOUDINARY\_STORAGE = {

    'CLOUD\_NAME': os.environ['CLOUD\_NAME'],

    'API\_KEY': os.environ['API\_KEY'],

    'API\_SECRET': os.environ['API\_SECRET'],

    'SECURE': True,

    'STATIC\_IMAGES\_EXTENSIONS': ['jpg', 'jpe', 'jpeg', 'jpc', 'jp2', 'j2k', 'wdp', 'jxr','png', 'gif', 'webp']

}

**Secret Key and Debug**

import django\_heroku

import os

from django.conf.global\_settings import AUTH\_USER\_MODEL

import dotenv

dotenv\_file = os.path.join(BASE\_DIR, ".env")

if os.path.isfile(dotenv\_file):

    dotenv.load\_dotenv(dotenv\_file)

SECRET\_KEY = os.environ['SECRET\_KEY']

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = (os.environ['DEBUG'] == True)

ALLOWED\_HOSTS = ['https://infostar.herokuapp.com/','127.0.0.1']

**Custom User Model**

**AbstractUser vs AbstractBaseUser :**

The default User model in Django uses a username to uniquely identify a user during authentication. If you'd rather use an email address, you'll need to create a custom User model by either subclassing AbstractUser or AbstractBaseUser.

Options:

AbstractUser: Use this option if you are happy with the existing fields on the User model and just want to remove the username field.

AbstractBaseUser: Use this option if you want to start from scratch by creating your own, completely new User model.

**Model Manager:**

**Models.py**

**from** **django.contrib.auth.base\_user** **import** BaseUserManager

**from** **django.utils.translation** **import** ugettext\_lazy **as** \_

**class** **CustomUserManager**(BaseUserManager):

*"""*

*Custom user model manager where email is the unique identifiers*

*for authentication instead of usernames.*

*"""*

**def** create\_user(self, email, password, \*\*extra\_fields):

*"""*

*Create and save a User with the given email and password.*

*"""*

**if** **not** email:

**raise** **ValueError**(\_('The Email must be set'))

email = self.normalize\_email(email)

user = self.model(email=email, \*\*extra\_fields)

user.set\_password(password)

user.save()

**return** user

**def** create\_superuser(self, email, password, \*\*extra\_fields):

*"""*

*Create and save a SuperUser with the given email and password.*

*"""*

extra\_fields.setdefault('is\_staff', **True**)

extra\_fields.setdefault('is\_superuser', **True**)

extra\_fields.setdefault('is\_active', **True**)

**if** extra\_fields.get('is\_staff') **is** **not** **True**:

**raise** **ValueError**(\_('Superuser must have is\_staff=True.'))

**if** extra\_fields.get('is\_superuser') **is** **not** **True**:

**raise** **ValueError**(\_('Superuser must have is\_superuser=True.'))

**return** self.create\_user(email, password, \*\*extra\_fields)

### **AbstractBaseUser:**

**from** **django.db** **import** models

**from** **django.contrib.auth.models** **import** AbstractBaseUser

**from** **django.contrib.auth.models** **import** PermissionsMixin

**from** **django.utils.translation** **import** gettext\_lazy **as** \_

**from** **django.utils** **import** timezone

**from** **.managers** **import** CustomUserManager

**class** **CustomUser**(AbstractBaseUser, PermissionsMixin):

email = models.EmailField(\_('email address'), unique=**True**)

is\_staff = models.BooleanField(default=**False**)

is\_active = models.BooleanField(default=**True**)

date\_joined = models.DateTimeField(default=timezone.now)

USERNAME\_FIELD = 'email'

REQUIRED\_FIELDS = []

objects = CustomUserManager()

**def** \_\_str\_\_(self):

**return** self.email

**User Model:**

**AbstractUser**

**Models.py:**

from django.db import models

from django.core.validators import RegexValidator

from django.contrib.auth.models import AbstractUser

from django.contrib.auth.validators import UnicodeUsernameValidator

from django.utils.translation import gettext\_lazy as \_

class User(AbstractUser):

    username\_validator = UnicodeUsernameValidator()

    username = models.CharField(

        \_('username'),

        max\_length=80,

        unique=True,

        help\_text=\_('Required:Letters, digits and @/./+/-/\_ only.'),

        validators=[username\_validator],

        error\_messages={

            'unique': \_("A user with that username already exists."),

        },

    )

    first\_name = models.CharField(\_('first name'),null=True,max\_length=30, blank=True)

    last\_name = models.CharField(\_('last name'),null=True,max\_length=30, blank=True)

    email = models.EmailField(\_('email address'),

                unique=True,

                blank=True,

                error\_messages={

                               'unique': \_("A user with that email already exists."),

                                },

                )

    phone\_no = models.CharField(\_('phone no'),

                                   validators= [RegexValidator("^0?[5-9]{1}\d{9}$")],

                                   unique=True,

                                   blank=True,

                                   help\_text=\_('(Optional: Phone No. Field!!)'),

                                   error\_messages={

                                        'unique': \_("A user with that phone number already exists."),

                                                   },

                                   null=True,

                                   max\_length=15,

                                   )

**Admin.py:**

from import\_export.admin import ImportExportModelAdmin

from django.contrib.auth.admin import UserAdmin

from django.utils.translation import gettext\_lazy as \_

from .models import User

from django.contrib import admin

class CustomUserAdmin(UserAdmin,ImportExportModelAdmin):

    """Define admin model for custom User model with no username field."""

    fieldsets = (

        (None, {'fields': ('username','email','password')}),

        (\_('Personal info'), {'fields': ('first\_name', 'last\_name','phone\_no')}),

        (\_('Permissions'), {'fields': ('is\_active', 'is\_staff', 'is\_superuser',

                                       'groups', 'user\_permissions')}),

        (\_('Important dates'), {'fields': ('last\_login', 'date\_joined')}),

    )

    add\_fieldsets = (

        (None, {

            'classes': ('wide',),

            'fields': ('username', 'password1', 'password2'),

        }),

    )

    list\_display = ('username', 'email', 'first\_name', 'last\_name', 'is\_staff')

    search\_fields = ('username', 'first\_name', 'last\_name', 'email')

    ordering = ('username',)

    filter\_horizontal = ('groups', 'user\_permissions',)

admin.site.register(User,CustomUserAdmin)

**settings.py:**

AUTH\_USER\_MODEL= 'Account.User'

**Logind With Email or Phone or Username**

**Dj/Views.py:**

from django.contrib.auth.backends import ModelBackend

import re

from django.contrib.auth import get\_user\_model

User = get\_user\_model()

class UsernameOrEmailBackend(ModelBackend):

    def authenticate(self,request, username=None, password=None,\*\*kwargs):

        if '@' in username:

            kwargs = {'email': username}

        elif re.search("^0?[5-9]{1}\d{9}$",username):

            kwargs = {'phone\_no':username}

        else:

            kwargs = {'username': username}

        try:

            user = User.objects.get(\*\*kwargs)

            if user.check\_password(password):

                return user

        except User.DoesNotExist:

            return None

    def get\_user(self, user\_id):

        try:

            return User.objects.get(pk=user\_id)

        except User.DoesNotExist:

            return None

**settings.py:**

AUTHENTICATION\_BACKENDS = (

     . . . . . . .

    'dj.views.UsernameOrEmailBackend',

    'django.contrib.auth.backends.ModelBackend',

)

**Like and dislike**

**Views.py:**

def Like\_user(request,pk):

    post = Post.objects.get(pk=pk)

    Like.objects.create(post=post,liked\_by=request.user)

    messages.error(request,"for like this post")

    return redirect(request.META['HTTP\_REFERER'])

def Unlike\_user(request,pk):

    post = Post.objects.get(pk=pk)

    Like.objects.filter(post=post,liked\_by=request.user).delete()

    return redirect(request.META['HTTP\_REFERER'])

**Search Function**

**Views.py:**

def search(request):

    if request.method == 'POST':

        srch = request.POST['src']

        if srch:

            match = Post.objects.filter(Q(title\_\_icontains=srch)|

                                        Q(description\_\_icontains=srch)|

                                        Q(caption\_\_icontains=srch)|

                                        Q(cr\_date\_\_icontains=srch)|

                                        Q(title\_\_istartswith=srch)|

                                        Q(description\_\_istartswith=srch)

                                        )

            if match:

                return render(request,'news/search.html',{"sr":match})

            else:

                messages.error(request,"No results found")

                return render(request,'news/search.html')

        else:

            return redirect(request.META['HTTP\_REFERER'])

    return render(request,'news/post\_list.html')

**Change Username**

**Views.py:**

@method\_decorator(login\_required,name="dispatch")

class Changeusername(SuccessMessageMixin,UpdateView):

    model = User

    fields = ['username','phone\_no']

    template\_name = 'news/change\_username.html'

    success\_message = "Your username and phone number successfully updated!!"

    def get\_success\_url(self):

        companyid=self.request.user.profile.id

        return reverse\_lazy('my\_profile', kwargs={'pk': companyid}

**Django Email**

**Method No. 1 : EmailMessage**

**Views.py:**

        template = render\_to\_string('email\_conform.html',{'name':customer})

        email = EmailMessage(

                            'Thank for purchasing the ecommerce course!',

                            template,

                            settings.EMAIL\_HOST\_USER,

                            [request.user.email],

                            )

        email.fail\_silently = False

        email.send()

**email\_change.html:**

Hey {{name}}!

Tye your email message here……….

Pradip Kachhadiya

**Method No. : 2 : send\_mail**

**Views.py:**

        email\_text = render\_to\_string('email\_conform.txt',{'name':customer})

        email\_template = render\_to\_string('email\_conform.html',{'name':customer})

        send\_mail(

                'Thank for purchasing the ecommerce course!',

                email\_text,

                settings.EMAIL\_HOST\_USER,

                [request.user.email],

                html\_message=email\_template,

                fail\_silently=False,

                )

**Method No. : 3 EmailMultiAlternatives**

from django.core.mail.message import EmailMultiAlternatives

from django.template.loader import render\_to\_string

   email\_template3 = render\_to\_string('email\_buyer3.html',{"postData" : postData,"shipping\_info":shipping\_info,"orderitem1":orderitem1,"user":order.customer.id})

    mail\_buyer3 = EmailMultiAlternatives(

                        "Order Completed",

                        "Your order detail",

                        settings.EMAIL\_HOST\_USER,

                        [shipping\_info.customer.email],

                    )

    mail\_buyer3.attach\_alternative(email\_template3, 'text/html')

    mail\_buyer3.send()

**SideBar with Html & Css & Js**

**Sidebar.html:**

<div class="sidebar\_\_btn">

    <span class="fa fa-bars bg-warning text-dark p-2 px-3 border-dark btn rounded font-weight-bold" style="font-size: 20px;" aria-hidden="true"></span>

</div>

<nav class="sidebar">

    <div class="text">Side Bar</div>

    <ul class="sidebar\_\_nav">

            <li class="main\_\_li"><a href="#">Dashboard</a></li>

            <li class="main\_\_li">

                <a href="#" class="feat\_btn">Searvices <span class="first"><i class="fa fa-caret-down" aria-hidden="true"></i></span></a>

                <ul class="sidebar\_\_nav submenu\_\_show">

                    <li><a href="#">Webdesign</a></li>

                    <li><a href="#">App master</a></li>

                </ul>

            </li>

            <li class="main\_\_li">

                <a href="#" class="srv\_btn">Feature <span class="second"><i class="fa fa-caret-down" aria-hidden="true"></i></span></a>

                <ul class="sidebar\_\_nav submenu\_\_show1">

                    <li><a href="#">pages</a></li>

                    <li><a href="#">Element</a></li>

                </ul>

            </li>

            <li class="main\_\_li"><a href="#">Portfolio</a></li>

            <li class="main\_\_li"><a href="#">Overview</a></li>

            <li class="main\_\_li"><a href="#">ShortCuts</a></li>

            <li class="main\_\_li"><a href="#">Feedback</a></li>

        </ul>

</nav>

**Javascript.js: (Add it above </body>)**

<script>

    $('.sidebar\_\_btn').click(function(){

        $(this).toggleClass('click');

        $('.sidebar').toggleClass('show');

    })

    $('.feat\_btn').click(function () {

        $('.submenu\_\_show').toggleClass("show");

        $('.first').toggleClass("rotate");

    });

    $('.srv\_btn').click(function () {

        $('.submenu\_\_show1').toggleClass("show1");

        $('.second').toggleClass("rotate");

    });

</script>

**Style.css:**

.sidebar {

  position: fixed;

  width: 250px;

  left: -250px;

  top: 0%;

  background-image: linear-gradient(

    to bottom,

    rgb(185, 218, 231),

    rgb(248, 212, 145),

    rgb(223, 147, 223)

  );

  height: 100%;

  overflow-y: auto;

  transition: left 0.7s ease;

}

.sidebar::-webkit-scrollbar {

  display: none;

}

.sidebar {

  -ms-overflow-style: none; /\* IE and Edge \*/

  scrollbar-width: none; /\* Firefox \*/

}

.sidebar .text {

  color: white;

  font-size: 25px;

  font-weight: 600;

  line-height: 150px;

  background: rgb(58, 57, 57);

  letter-spacing: 1px;

}

.sidebar\_\_nav {

  list-style: none;

  width: 100%;

  height: 100%;

  margin: 0;

  padding: 0;

}

.main\_\_li {

  line-height: 55px;

  border-bottom: 1px solid rgb(109, 105, 105, 0.2);

}

.sidebar\_\_nav > li > a {

  position: relative;

  color: black;

  font-weight: 500;

  text-decoration: none;

  font-size: 18px;

  font-weight: 400;

  display: block;

  width: 100%;

  padding-left: 40px;

  border-left: 3px solid transparent;

  transition: 0.2s;

}

.sidebar\_\_nav > li > a:hover {

  color: orange;

  background-color: rgb(77, 75, 75);

  border-left-color: orange;

}

.sidebar > ul > li > ul {

  position: static;

  display: none;

}

.sidebar ul li .submenu\_\_show.show {

  display: block;

}

.sidebar ul li .submenu\_\_show1.show1 {

  display: block;

}

.main\_\_li > ul > li {

  line-height: 42px;

  border-bottom: none;

}

.main\_\_li > ul > li > a {

  font-size: 17px;

  color: rgb(0, 0, 0);

  padding-left: 80px;

  background-color: bisque;

  border-bottom: 0.2px solid rgb(128, 128, 128, 0.2);

}

.main\_\_li > a > span {

  position: absolute;

  top: 50%;

  right: 20px;

  transform: translateY(-50%);

  font-size: 20px;

  transition: transform 0.4s;

}

.sidebar\_\_nav > li > a > span.rotate {

  transform: translateY(-50%) rotate(-180deg);

}

.sidebar\_\_btn {

  position: absolute;

  top: 2px;

  left: 1px;

  transition: left 0.7s ease;

}

.sidebar\_\_btn.click {

  left: 249px;

  z-index: 1;

}

.sidebar\_\_btn.click span:before {

  content: "❌";

}

.sidebar.show {

  left: 0px;

}

**Permanently Delete the Software**

1. **Control panel  Uninstall software**
2. **%programfiles%**
3. **%programfiles(x86)%**
4. **%appdata%**
5. **Computer\HKEY\_CURRENT\_USER\SOFTWARE**
6. **Computer\HKEY\_LOCAL\_MACHINE\SOFTWARE**
7. **temp**
8. **%temp%**

**Save image as thumbnail**

**Models.py:**

    def save(self):

        super().save()  # saving image first

        if self.profile\_pic:

            img = Image.open(self.profile\_pic.path) # Open image using self

            if img.height > 300 or img.width > 300:

                new\_img = (300, 300)

                img.thumbnail(new\_img)

                img.save(self.profile\_pic.path)

    def image\_tag(self):

        if self.profile\_pic:

            return mark\_safe('<img src="%s" width="45px" height="45px" />' % (self.profile\_pic.url))

    image\_tag.short\_description = 'Profile Pic.'

    # def save(self, \*args, \*\*kwargs):

    #     super(Profile, self).save(\*args, \*\*kwargs)

    #     if self.profile\_pic:

    #         imag = Image.open(self.profile\_pic)

    #         output\_size = (300, 300)

    #         imag.thumbnail(output\_size,Image.ANTIALIAS)

    #         fh = storage.open(self.profile\_pic.name, "w")

    #         format = 'png'

    #         imag.save(fh,format)

    #         fh.close()

**Show image in admin panel**

@admin.register(Profile)

class ProfileAdmin(admin.ModelAdmin):

    list\_display= ["id","image\_tag","user","full\_name","country","created\_date"]

    search\_fields=["id","public\_name","full\_name",'gender',"country","phone\_number"]

    autocomplete\_fields = ["user"]

    list\_per\_page = 3

    list\_filter = ["gender","country","created\_date"]

    list\_display\_links = ["id","user","image\_tag","full\_name","created\_date"]

    readonly\_fields = ['image\_tag']

**Truncate the word at admin side**

@admin.register(PhotoComment)

class PhotoCommentAdmin(admin.ModelAdmin):

    list\_display= ["id","user","get\_comment\_text","comment\_reply","created\_date"]

    search\_fields=["id","comment\_text","comment\_reply",'created\_date']

    autocomplete\_fields = ["user"]

    list\_per\_page = 3

    list\_filter = ['created\_date']

    list\_display\_links = ["id","user","get\_comment\_text","comment\_reply","created\_date"]

    def get\_comment\_text(self, obj):

        return truncatewords(obj.comment\_text, 20)

    get\_comment\_text.short\_description = "Comment"

**List filter of foreign key**

**1). Give autocomplete\_fields :**

@admin.register(UserPhoto)

class UserPhotoAdmin(admin.ModelAdmin):

    list\_display= ["id","user","pr\_rank","created\_date"]

    search\_fields=["id","description","public\_upload",'created\_date']

    autocomplete\_fields = ["user"]

**2). Give search\_fields = ['foreinkeyfield\_\_name'] :**

search\_fields=["id","user\_\_username","description","public\_upload",'created\_date']

**annotate in django**

**annotate Use :** Add Extra field inside current query object and fetch it..

**aggregate use? : It return dictionary…**

*# Each publisher, each with a count of books as a "num\_books" attribute.*

>>> **from** **django.db.models** **import** Count

>>> pubs = Publisher.objects.annotate(num\_books=Count('book'))

>>> pubs

<QuerySet [<Publisher: BaloneyPress>, <Publisher: SalamiPress>, ...]>

>>> pubs[0].num\_books

73

**Show data as tabular or stackinline at admin site**

class PropertyImageInline(admin.StackedInline):

    model = PhotoComment

    extra = 1

**OR**

class PropertyImageInline(admin.TabularInline):

    model = PhotoComment

    extra = 1

@admin.register(UserPhoto)

class UserPhotoAdmin(admin.ModelAdmin):

    list\_display= ["id","user","pr\_rank","created\_date"]

    search\_fields=["id","user\_\_username","description","public\_upload",'created\_date']

    autocomplete\_fields = ["user"]

    list\_per\_page = 10

    list\_filter = ["pr\_rank","created\_date"]

    list\_display\_links = ["id","user","pr\_rank","created\_date"]

    inlines = [ PropertyImageInline, ]

**Here foreignkey is must with UserPhoto….**

**Delete Image permanently(clear old image)**

profile = Profile.objects.get(id = profile\_id)

profile.profile\_pic.delete()

profile.profile\_pic.delete() if serializer.validated\_data.get('profile\_pic') else False

* **OneToOneField =>** Reverse field execution return model object.
* **ForeignKey/ManyToMany** => Reverse field execution return model manager..So we can perforn .all(),,filter(),etc on this..

**Select Related**

**Use in foreignkey or one to one relation:**

**select\_related()**[**¶**](headline)

**select\_related(*\*fields*)**[**¶**](definition)

Returns a **QuerySet** that will “follow” foreign-key relationships, selecting additional related-object data when it executes its query. This is a performance booster which results in a single more complex query but means later use of foreign-key relationships won’t require database queries.

The following examples illustrate the difference between plain lookups and **select\_related()** lookups. Here’s standard lookup:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*# Hits the database.*

e = Entry.objects.get(id=5)

*# Hits the database again to get the related Blog object.*

b = e.blog

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*# Hits the database.*

e = Entry.objects.select\_related('blog').get(id=5)

*# Doesn't hit the database, because e.blog has been prepopulated*

*# in the previous query.*

b = e.blog

**This work only foreign key field….So here b contain Entry model object and blog model object…so performance is increase…**

**prefetch\_related**

**Use in many to many relation:**

Profile.objects.prefetch\_related('user\_blocked','pandingRequest','user\_following','user\_followers').get(user = self.request.user)

**Check Sql Query Execute Count**

* **print(len(connection.queries))** => Check sql execute Total count
* **reset\_queries()** => Clear of previous sql count

    def get(self, request,format=None):

        profile\_id = request.GET.get('profile\_id')

        print(len(connection.queries))

        reset\_queries()

        try:

            profile = Profile.objects.select\_related('user').get(id = profile\_id or request.user.profile.id)

            print(len(connection.queries))

            reset\_queries()

        except Exception:

            return Response({"Fail" : "No profile found!!"},status=status.HTTP\_400\_BAD\_REQUEST)

        serializer\_data = ProfileSerializer(profile).data

        print(len(connection.queries))

**Encryption Algo.**

if \_\_name\_\_ == "\_\_main\_\_":

    from settings import ENCRYPTION\_KEY

else:

    from .settings import ENCRYPTION\_KEY

import base64

from django.db import models

class Crypto(object):

    def \_\_init\_\_(self,data):

        self.key = ENCRYPTION\_KEY

        self.data = str(data)

    def encrypt(self):

        try:

            encoded\_chars = []

            for i in range(len(self.data)):

                key\_c = self.key[i % len(self.key)]

                encoded\_c = chr(ord(self.data[i]) + ord(key\_c) % 256)

                encoded\_chars.append(encoded\_c)

            encoded\_string = ''.join(encoded\_chars)

            en\_data = base64.b64encode(bytes(encoded\_string,'utf-8')).decode()   # Save this in database

            return en\_data

        except Exception as e:

            return print(e)

    def decrypt(self):

        try:

            sample\_string\_bytes = base64.b64decode(self.data)

            sample\_string = sample\_string\_bytes.decode()

            encoded\_chars = []

            for i in range(len(sample\_string)):

                key\_c = self.key[i % len(self.key)]

                encoded\_c = chr((ord(sample\_string[i]) - ord(key\_c) + 256) % 256)

                encoded\_chars.append(encoded\_c)

            encoded\_string = ''.join(encoded\_chars)

            return encoded\_string

        except Exception:

            return "Update this data again!! Something went wrong when fetching this data...."

**Customize Model Field**

* Here I use **Charfield** :

class EncodedField(models.CharField):

    '''

    Encripted model field

    '''

    prefix = 'enc\_sha256\_'

    def \_\_init\_\_(self, max\_length,\*args, \*\*kwargs):

        if max\_length:

            max\_length += len(self.prefix)

        super(EncodedField, self).\_\_init\_\_(max\_length=max\_length,\*args, \*\*kwargs)

    def get\_internal\_type(self):

        return "CharField"  Customize here like textfield,jsonfield,etc..

    def from\_db\_value(self, value, expression, connection):

        return self.to\_python(value)

    def get\_prep\_value(self, value):

        if value and not value.startswith(self.prefix):

            value = self.prefix + Crypto(value).encrypt()

        return value

    def to\_python(self, value):

        if value and (value.startswith(self.prefix)):

            retval = Crypto(value[len(self.prefix):]).decrypt()

        else:

            retval = value

        return retval

**models.py:**

phone\_number = EncodedField(null=True,blank=True,max\_length=20,unique=True)

**Currency Converter**

import requests

import json

class CurrencyConverter():

    def \_\_init\_\_(self):

        self.url1 = 'https://api.ratesapi.io/api/latest'

        self.url2 = 'https://api.exchangerate-api.com/v4/latest/USD'

    def convert\_try\_1(self,from\_currency, to\_currency, amount, date\_obj='latest'):

        payload = {'base': from\_currency, 'symbols': to\_currency}

        response = requests.get(self.url1, params=payload)

        if response.status\_code != 200:

            converted\_amount = self.convert\_try\_2(from\_currency,to\_currency,amount)

        else:

            rate = json.loads(response.text)

            converted\_amount = rate.get('rates').get(to\_currency) \* amount

        return converted\_amount

    def convert\_try\_2(self, from\_currency, to\_currency, amount):

        data= requests.get(self.url2).json()

        currencies = data['rates']

        if from\_currency != 'USD' :

            amount = amount / currencies[from\_currency]

        converted\_amount = amount \* currencies[to\_currency]

        return converted\_amount

if \_\_name\_\_ == '\_\_main\_\_':

    rate = CurrencyConverter().convert\_try\_1('KWD','INR',1)

    print(rate)

'''

possiibilities:

1. INR

2. GBP(£) to INR

3. EUR(€) to INR

4. USD($) to INR

5. CAD(C$,Can$) to INR

6. AUD(A$) to INR

7. RUB(₽) to INR

8. KWD(₽) to INR

'''

**Send Email by using Thread**

from django.template.loader import render\_to\_string

from django.core.mail.message import EmailMultiAlternatives

from django.conf import settings

import threading

class EmailThread(threading.Thread):

    def \_\_init\_\_(self,template,subject,receiver\_email,\*\*kwargs):

        self.template = template

        self.subject = subject

        self.receiver\_email = receiver\_email

        self.kwargs = kwargs

        threading.Thread.\_\_init\_\_(self)

    def run(self):

        try:

            print("Email start")

            email\_template = render\_to\_string(self.template,self.kwargs)

            email\_content = EmailMultiAlternatives(

                            self.subject,

                            None,

                            settings.EMAIL\_HOST\_USER,

                            [self.receiver\_email],

                        )

            email\_content.attach\_alternative(email\_template, 'text/html')

            email\_content.send()

            print("Email sendddd")

        except Exception:

            return None

def send\_email(template,subject,receiver\_email,\*\*kwargs):

    return EmailThread(template, subject, receiver\_email,\*\*kwargs).start()

**Create Public Name**

**Signals.py:**

from django.dispatch.dispatcher import receiver

from django.db.models.signals import post\_save

from .models import Profile

from random import randint as random\_randint

@receiver(post\_save, sender=Profile)

def create\_public\_name(sender, instance, created, \*\*kwargs):

    if created:

        full\_name = instance.full\_name.replace(" ", "\_")

        number = random\_randint(100,999)

        prof = Profile.objects.get(user = instance.user)

        prof.public\_name = (full\_name[0 : : 3]).upper() + '@' + "\_".join(str(number))

        prof.save(update\_fields = ['public\_name'])

**Delete Image Path :**

from django.dispatch.dispatcher import receiver

from django.db.models.signals import pre\_save

from .models import Category

@receiver(pre\_save, sender=Category)

def pre\_save\_image(sender, instance, \*args, \*\*kwargs):

    try:

        old\_img = instance.\_\_class\_\_.objects.get(id=instance.id).category\_img.path

        try:

            new\_img = instance.category\_img.path

        except:

            new\_img = None

        if new\_img != old\_img:

            import os

            if os.path.exists(old\_img):

                os.remove(old\_img)

    except:

        pass

**Use of transaction decorator in Django**

Jab kisi bhi view ke uper ye decorator laga huva hei tab ye view ke under jitni bhi sql query run hogi vo puri tarah se chalegi ya to fir nahi chalegi agar koi bhi query missing hogi tab..

**Source Link** : https://docs.djangoproject.com/en/3.2/topics/db/transactions/

**Where should we use content types and generic relations in django?**

**Source Link :** [**https://django.cowhite.com/blog/where-should-we-use-content-types-and-generic-relations-in-django/**](https://django.cowhite.com/blog/where-should-we-use-content-types-and-generic-relations-in-django/)

**settings.py:**

INSTALLED\_APPS = [

...

'django.contrib.contenttypes',

...

]

**Views.py:**

from django.db import models

from django.contrib.contenttypes.fields import GenericForeignKey

from django.contrib.contenttypes.models import ContentType

class Like(models.Model):

liked\_by = models.ForeignKey(User)

created\_at = models.DateTimeField(auto\_now\_add=True)

# Listed below are the mandatory fields for a generic relation

content\_type = models.ForeignKey(ContentType, on\_delete=models.CASCADE)

object\_id = models.PositiveIntegerField()

content\_object = GenericForeignKey()

from django.db import models

from django.contrib.contenttypes.fields import GenericRelation

class Post(models.Model):

...

likes = GenericRelation(Like)

class Page(models.Model):

...

likes = GenericRelation(Like)

class Comment(models.Model):

...

likes = GenericRelation(Like)

* **Now let's see how to add likes for a post. We can do it in different ways:**

# Get the post object

post = Post.objects.get(pk=1)

# Add a like for the post

post.likes.create(liked\_by=request.user)

# Or in a similar way using the Like model to add the like

Like.objects.create(content\_object=post, liked\_by=request.user)

* You could make your life easier by adding a reverse reference or **reverse\_query\_name** as Django calls it, to the **GenericRelation** definition. And query the Like model using this reverse query name as a lookup in the **queryset**.

from django.db import models

from django.contrib.contenttypes.fields import GenericRelation

class Post(models.Model):

...

posted\_by = models.ForeignKey(User)

likes = GenericRelation(Like, related\_query\_name='post'))

...

...

Like.objects.filter(post\_\_posted\_by\_\_first\_name='Bob')

**Elastic Search In Django**

**Refrence Link :** <https://medium.com/geekculture/how-to-use-elasticsearch-with-django-ff49fe02b58d>

**DownLoad ElasticSearch** :  [https://www.elastic.co/downloads/elasticsearch](./%20https:/www.elastic.co/downloads/elasticsearch)

**Def** :

* **Index** — a collection of different types of documents and document properties. For example, a document set may contain the data of a social networking application.
* **Type/Mapping** − a collection of documents sharing a set of common fields present in the same index. For example, an index contains data of a social networking application; there can be a specific type for user profile data, another type for messaging data, and yet another one for comments data.
* **Document** − a collection of fields defined in the JSON format in a specific manner. Every document belongs to a type and resides inside an index. Every document is associated with a unique identifier, called the UID.
* **Field** — Elasticsearch fields can include multiple values of the same type (essentially a list). In SQL, on the other hand, a column can contain exactly one value of the said type.
* pip install django-elasticsearch-dsl
* Then add django\_elasticsearch\_dsl to the INSTALLED\_APPS

Settings.py

ELASTICSEARCH\_DSL={

'default': {

'hosts': 'localhost:9200'

},

}

**Documents.py :**

from django\_elasticsearch\_dsl import Document

from django\_elasticsearch\_dsl.registries import registry

from .models import Product

@registry.register\_document

class CategoryDocument(Document):

    class Index:

        name = 'product'

        settings = {

        'number\_of\_shards': 1,

        'number\_of\_replicas': 0

        }

    class Django:

        model = Product

        fields = [

             'product\_name',

         ]

* To create and populate the Elasticsearch index and mapping use the search\_index command: python manage.py search\_index -- rebuild
* For more help use python manage.py search\_index — help command
* **pip install django-elasticsearch-dsl-drf**

**serializer.py:**

from rest\_framework.serializers import ModelSerializer

from .models import Product

from django\_elasticsearch\_dsl\_drf.serializers import DocumentSerializer

from .documents import CategoryDocument

class ProductSerializer(ModelSerializer):

    class Meta:

        model = Product

        fields = '\_\_all\_\_'

class ProductSearchSerializer(DocumentSerializer):

    class Meta:

        document = CategoryDocument

        fields = ['id','product\_name','price','quantity']

**views.py:**

from rest\_framework.viewsets import ModelViewSet

from .models import Product

from .serializers import ProductSerializer,ProductSearchSerializer

from .documents import CategoryDocument

from django\_elasticsearch\_dsl\_drf.filter\_backends import SearchFilterBackend

from django\_elasticsearch\_dsl\_drf.viewsets import DocumentViewSet

class ProductViewSet(ModelViewSet):

    queryset = Product.objects.all()

    serializer\_class = ProductSerializer

class ProductSearchViewset(DocumentViewSet):

    document = CategoryDocument

    serializer\_class = ProductSearchSerializer

    search\_fields = ['product\_name']

**start the elastic search :**

* Run bin/elasticsearch (or bin\elasticsearch.bat on Windows)

**Postgresql Command**

1. sudo service postgresql start  Start the postgres service
2. sudo -u postgres psql  enter in db
3. \l  Show all db and their details
4. alter role postgres with password 'root';
5. \q  for the exit
6. psql -h localhost -U postgres
7. create database dealsautomation owner postgres;  Create new db
8. drop database dealsautomation;  Delete the database

**create role:**

sudo -u postgres psql  
postgres=# create database mydb;  
postgres=# create user myuser with encrypted password 'mypass';  
postgres=# grant all privileges on database mydb to myuser;