Task1:multipleapps

creating project

Django-admin startproject projectname

cd projectname

creating application

python3 manage.py startapp appone

python3 manage.py startapp apptwo

python3 manage.py startapp appthree

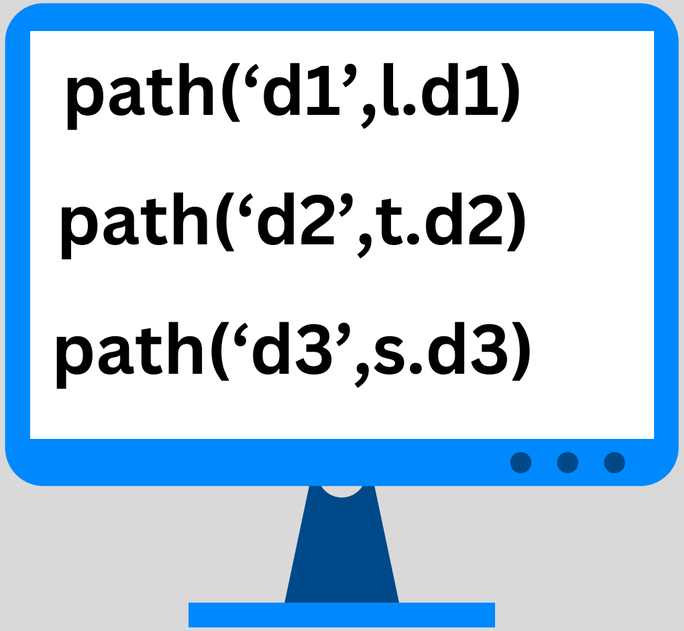
https://github.com/Swatilingshetty/Django24

https://localhost:8000/d1

https://localhost:8000/d2

https://localhost:8000/d3

urls.py



views.py

appone
d1
list

apptwo
d2
tuple

appthree
d3
str

http://127.0.0.1:8000/d1/
http://127.0.0.1:8000/d2/
http://127.0.0.1:8000/d3/

Task2: DTL:Django templete language

1. By using DTL we develop HTML pages
2. in HTML we cannot write code directly because use it Markup language to write python code in HTML we use DTL
3. We need to configure Django Templates configuration in settings.py file
4. ‘DIRS’:[os.path.join(BASE_DIR,’templates‘)] and import os
5. we need to configure the application in INSTALLED_APPS=[appone]

a.variables
b.Filters
c.Tags
d.Comments
e.Templete inheritance

Variables: Variables are associated with a context can be accessed ny {{}}

HTML file

```
<h1>{{Welcome}}</h1>
```

```
def welcomeMessage{request}
templete=loader.get_templete{’welcomeMessage.html}
text={’welome’:’welcome to django’}
return HTTpResponse(templete.render(context=text))
```

HTML file

```
<h1>{{key1}}</h1>
<h1>{{key2}}</h1>
<h1>{{key3}}</h1>
```

```
def welcomeMessage{request}
templete=loader.get_templete{’welcomeMessage.html}
text={”key”:”value”,”key”:”value”,”key”:”value”}
return HTTpResponse(templete.render(context=text))
```

Filters

HTML file

```
<h1>{{key1 | upper}}</h1>
<h1>{{key2 | lower}}</h1>
<h1>{{key3 | capfirst}}</h1>
<h1>{{key4 | length</h1>
<h1>{{key5 | truncatechars :10}}
</h1>
<h1>{{key6 | slice:”0.5"}}</h1>
<h1>{{key7 | truncateeords:”5"}}
</h1>
```

```
def getNames(request):
templete=loader.get_templete{’welcomeMessage.html}
names={
“key1”:”nameone”
“key1”:”nameone”
“key1”:”nameone”
“key1”:”nameone”
“key1”:”nameone”
“key1”:”nameone”
“key1”:”nameone”
}
return HttpResponse(template.render(context=names))
```

For loop

HTML File

```
<body>
(% for i in mobiebrands %)
<h1>{{i.productbrand}}</h1>
<h1>{{i.productcost}}</h1>
<h1>{{i.productmodel}}</h1>
{% end for %}
</body>
```

```
Mobilebrands ={
{
‘productbrand’:’samsung’
productcost’:’10000'
productmodel’:’note 10'
}
{
‘productbrand’:’redmi’
‘productcost’:11000'
‘productmodel’:’redmi 10'
}
def getproducts(request):
templete=loader.get_templete{’welcomeMessage.html}
p={’mobilebrands’:’mobilebrands’}
return HttpResponse(template.render(context=names))
```

if else

path('getID/',views.getID)

```
<body>
    {% if id%}
    <h1>user ID :{{id}}</h1>
    {% else %}
    <h1>no user ID : {{id}}</h1>
    {% endif %}
</body>
```

```
def getID(request):
    template=loader.get_template('getID.html')
    context={'id:': [ ]}
     return HttpResponse(template.render[context])
```

Tamplate inheritance

DTL supports Tamplate inheritance allowing you to create a base tamplate with common elements and extend or override specific blocks in child tamplate

```
from django.shortcuts import render

# Create your views here.
def home(request):

return render(request,'home.html')

def about(request):

return render(request,'about.html')

def contact(request):

return render(request,'contact.html')

def services(request):

return render(request,'services.html')
```

```
INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'appone',
]
```

'DIRS': [BASE_DIR/ 'templates'],

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

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" ,content="width=device-width", initial="s
<title>{% block title %}{% endblock %} </title>
<link rel="stylesheet" type="text/css" href="{% static 'style.css' %
</head>
<body>
<div id="content">
{% block content %}

{% endblock %}

</div>

</body>
```

Django models

Django models are a part of Django object Relational mapping (ORM) Layer
they can define structure of the database tables and provides a pythonic way
to interact with the database.
Each model maps to single databse tables

django-admin startproject projectone
python manage.py startapp appone
open the installed apps and configure the project (appone)
open templates and configure templates folder with base_Dir
open databases and configure
databasename,username,password,engine

models.py (we write models here ’
from django.db import models

models.py (we write models here)

from django.db import models

```
C:\Django24\django04_models\projectone>python
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from django.db import models
>>> dir(models)['Aggregate', 'AutoField', 'Avg', 'BLANK_CHOICE_DASH', 'BaseConstraint', 'BigAutoField', 'BigIntegerField',
'BinaryField', 'BooleanField', 'CASCADE', 'Case', 'CharField', 'CheckConstraint', 'Choices', 'CommaSeparatedIntegerField',
'Count', 'DEFERRED', 'DO_NOTHING', 'DateField', 'DateTimeField', 'DecimalField', 'Deferrable', 'DurationField', 'EmailField',
'Empty', 'Exists', 'Expression', 'ExpressionList', 'ExpressionWrapper', 'F', 'Field', 'FileField', 'FilePathField', 'FilteredRelation',
'FloatField', 'ForeignKey', 'ForeignObject', 'ForeignObjectRel', 'Func', 'GeneratedField', 'GenericIPAddressField',
'IPAddressField', 'ImageField', 'Index', 'IntegerChoices', 'IntegerField', 'JSONField', 'Lookup', 'Manager', 'ManyToManyField',
'ManyToManyRel', 'ManyToOneRel', 'Max', 'Min', 'Model', 'NOT_PROVIDED', 'NullBooleanField', 'ObjectDoesNotExist',
'OneToOneField', 'OneToOneRel', 'OrderBy', 'OrderWrt', 'OuterRef', 'PROTECT', 'PositiveBigIntegerField',
'PositiveIntegerField', 'PositiveSmallIntegerField', 'Prefetch', 'ProtectedError', 'Q', 'QuerySet', 'RESTRICT', 'RestrictedError',
'RowRange', 'SET', 'SET_DEFAULT', 'SET_NULL', 'SlugField', 'SmallAutoField', 'SmallIntegerField', 'StdDev', 'Subquery', 'Sum',
'TextChoices', 'TextField', 'TimeField', 'Transform', 'URLField', 'UUIDField', 'UniqueConstraint', 'Value', 'ValueRange',
'Variance', 'When', 'Window', 'WindowFrame', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__path__', '__spec__', 'aggregates', 'aggregates_all', 'aprefetch_related_objects', 'base', 'constants',
'constraints', 'constraints_all', 'deletion', 'enums', 'enums_all', 'expressions', 'fields', 'fields_all', 'functions', 'indexes',
'indexes_all', 'lookups', 'manager', 'options', 'prefetch_related_objects', 'query', 'query_utils', 'signals', 'sql', 'utils']
```

models.py (we write models here)

```
from django.db import models

# Create your views here.
class userprofile(models.model):

    username=models.charfield(max_length=100)
     email=models.Emailfield(max_length=10
    contact=models.Integerfield()
```

commands for migrations

pip install mysql
python manag.py makemigrations
python manag.py sqlmigrate appone 0001
python manage.py migrate

views.py(get all the records from the database)

```
from django.shortcuts import models
from django.template import loader
from django.http import HttpResponse
from appone.models import userprofile

# Create your views here.
def userprofiledetails(request):
                template=loader.get_template("userprofiledetails.html")
                details=userprofile.objects.all()
                 d={"details":details}
                 return HttpResponse(template.render(context=d))
```

urls.py

```
from django.contrib import admin
from django.urls import path
from appone import views

urlpatterns = [
  path('admin/', admin.site.urls),
  path('userprofiledetails/',views.userprofiledetails),
]
```

templates

userprofiledetails.html

```
<body>
   <h3>user profile</h3>
   {% if details %}
   <table style="width: 100;">
      <thead>
            <th>Username</th>
            <th>Email</th>
             <th>Contact</th>
    </thead>
    {% for p in details %}
    <tr>
       <td> {{p.Username}}</td>
       <td> {{p.Email}}</td>
       <td> {{p.Contact}}</td>
    </tr>
    {% endfor %}
    {% else %}
    <p>no records found in database</p>
    {% endif %}
   </table>
  </body>
```

http://127.0.0.1:8000/userprofiledetails/

http://127.0.0.1:8000/userprofiledetails/

comands for creating super user

python manage.py createsuperuser
username>admin
email address>
password>admin
password>admin
Y/N:Y

Admin.py

```
from django.contrib import admin
from appone.models import userprofile


# Register your models here.

admin.site.register(userprofile)
```

admin.py

```
from django.contrib import admin
from appone.models import userprofile
# Register your models here.

class userprofileadmin(admin.ModelAdmin):
        list_display=("username","email","contact")
        search_fields=(“username”,)

 admin.site.register(userprofile,userprofileadmin)
```

admin.py

```
from django.contrib import admin
from appone.models import userprofile
# Register your models here.

class userprofileadmin(admin.ModelAdmin):
        list_display=("username","email","contact")

admin.site.register(userprofile,userprofileadmin)
```

ORM

Django ORM (object relational mapping) is a component of Django that allow you to interact with your database like you would use SQL

Field lookups

__exact
__iexact
__contains
__gt
__gte etc

Steps in the console

Open last model project and insert some records
open shell and follow up some commands
>>>python manage.py shell

```
>>> from appone.models import userprofile
        >>> userprofile .object.all()

>>> u=userprofile.objects.all()
>>> print(u)

>>> print(type(u))
<class 'django.db.models.query.QuerySet'>

>>> u=userprofile.objects.get(id=1)
>>> print(u)
userprofile object (1)

userprofile object (1)
>>> print(u.username,u.email,u.contact)
nameone nameone@gmail.com 878765432
```

Field lookups

userprofile.objects.all()

userprofile.objects.filter(username__contains="swati")

userprofile.objects.filter(username__icontains="name")

userprofile.objects.filter(id__in=[2,6,3])

userprofile.objects.filter(username__istartswith="n")

userprofile.objects.filter(username__startswith="n")

userprofile.objects.filter(username__endswith="n")

userprofile.objects.filter(id__exact=None)

userprofile.objects.filter(id__exact=2)

userprofile.objects.filter(username__exact="nameone")

userprofile.objects.filter(username__startswith="none")| userprofile.objects.filter(username__startswith="swati")

userprofile.objects.filter(username__startswith="bheem")& userprofile.objects.filter(username__startswith="bheem")

userprofile.objects.filter(username__startswith="name")& userprofile.objects.filter(username__startswith="name")

userprofile.objects.all().values("username","email")

userprofile.objects.all().values("username","email",named=True)

Field lookups

```
userprofile.objects.all().order_by("salary")
userprofile.objects.all().order_by("-salary")
userprofile.objects.all().order_by("salary")[0:3]
userprofile.objects.all().order_by("username")[0:3]
userprofile.objects.all().order_by("username")
userprofile.objects.all().order_by("email")
```

Aggregation Functions

```
userprofile.objects.all().aggregate(Avg("salary"))
userprofile.objects.all().aggregate(Count("salary"))
userprofile.objects.all().aggregate(Max("salary"))
userprofile.objects.all().aggregate(Min("salary"))
userprofile.objects.all().aggregate(Sum("salary"))
```

Methods

create

```
userprofile.objects.create(username="harsha",email="harsha@gmail.com",contact=636445158,salary=35000)
```

Save

```
details=userprofile(username="harshith",email="harsha@gmail.com",contact=900872471,salary=30000)
details.save()
```

```
details=userprofile.objects.filter(username="bheem").update(salary="50000")
```

bulk_records

```
userprofile.objects.bulk_create([
   userprofile(username="vishal",email="vishal@gmail.com",contact=900872471,salary=30000),
   userprofile(username="harshini",email="harshini@gmail.com",contact=900872471,salary=30000),
   userprofile(username="satya",email="satya@gmail.com",contact=900872471,salary=30000),
   userprofile(username="shashii",email="shashii@gmail.com",contact=900872471,salary=30000),
])
```

update

```
userprofile.objects.filter(username="bheem").update(salary="50000") or
details=userprofile.objects.get(id=8)
details.salary=50000
details.save()
```

delete

```
details=userprofile.objects.filter(username="bheem").delete() or
# details=userprofile.objects.get(id=1)
details=userprofile.objects.all()
details.delete()
```

MODEL FORMS

Models.py

```
from django.db import models

# Create your models here.
class Userprofile(models.Model):

    firstname=models.CharField(max_length=100)
    lastname=models.CharField(max_length=100)
    coarsename= models.CharField(max_length=100)
   email=models.EmailField(max_length=100)
    joindate=models.DateField()
```

views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from . import forms

# Create your views here.

def adduser(request):

 form=forms.userprofileform()
 if request.method == 'POST':
  f=form.userprofileform(request.POST)
  print(f)

 return render(request,'adduser.html',{'form':form})
```

Forms.py

```
from django import forms
from appone.models import Userprofile

class userprofileform(forms.ModelForm):

        class Meta:
            model=Userprofile
            fields='__all__'
```

adduser.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport"  ,content="width=device-width", initial="scale=1.0">
    <title>table Pr UL </title>

  </head>
  <body>
    <h1>User registration form</h1>
    <form action="post">
     {{form.as_table}}
     {%csrf_token%}

    </form>
  </body>
</html>
```

views.py

```
 cleaned data
def adduser(request):

 form=forms.userprofileform()
 if request.method =='POST':

  f=form.userprofileform(request.POST)
  if f.is_valid():
   print(f.is_valid())
   print(f.cleaned_data)

   return HttpResponse("data submitted successfulyy")
  else:
   ff=forms.userprofileform()
   print(ff.is_valid())
   return HttpResponse("data not submitted")

return render(request,'adduser.html',{'form':form})
```

```
# for database we use this
def adduser(request):

  form=forms.userprofileform()
  if request.method =='POST':

      f=forms.userprofileform(request.POST)
      if f.is_valid():
          f.save()
       return HttpResponse("data submitted successfulyy")
 else:

      return HttpResponse("data not submitted ")

 return render(request,'adduser.html',{'form':form})
```

migrations

Form validation

1.create project
2.create application
3.configure setting

create forms.py file application

```
from django import forms


class userprofileform(forms.Form):


 first_name=forms.CharField(max_length=100)
 last_name=forms.CharField(max_length=100)
 email=forms.EmailField()
 phone=forms.CharField(max_length=15)
```

views.py

```
from django.shortcuts import render
from . import forms

# Create your views here.
def userprofileview(request):

 form=form.userprofileform()
 return render(request,'appone/userprofile.html',{'form':form})
```

create templates folder

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
  <meta name="viewport" ,content="width=device-width", initial="scale=1.0">
  <title>document </title>

</head>
<body>
 {{form.as_p}}
 {%csrf_token%}

 </body>
</html>
```

http://127.0.0.1:8000/

photo...

http://127.0.0.1:8000/userprofile/

photo..

photo..

Impliment cleaned_data

```
def userprofileview(request):

  form=forms.userprofileform()
  if request.method=='POST':
    form=forms.userprofileform(request.POST)
    if form.is_valid():
      print("validation success:")
      print("first name:", form.cleaned_data['first_name'])
      print("last name:", form.cleaned_data['last_name'])
      print("emails:", form.cleaned_data['email'])
      print("phone:", form.cleaned_data['phone'])

  return render(request,'userprofile.html',{'form':form})
```

Open interactive shell

```
C:\Django24\django07_formvalidation\projectone>
C:\Django24\django07_formvalidation\projectone>python manage.py shell
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from appone.forms import userprofileform
>>> u = userprofileform()
>>> print(u)
<div>
   <label for="id_first_name">First name:</label>
<input type="text" name="first_name" maxlength="100" required id="id_first_name">

</div>

 <div>
   <label for="id_last_name">Last name:</label>
<input type="text" name="last_name" maxlength="100" required id="id_last_name">
</div>

 <div>
   <label for="id_email">Email:</label>
<input type="email" name="email" maxlength="320" required id="id_email">
</div>

 <div>
   <label for="id_phone">Phone:</label>
<input type="text" name="phone" maxlength="15" required id="id_phone">
/div>
```

```
>>> print(u.as_p())
<p>
  <label for="id_first_name">First name:</label>
  <input type="text" name="first_name" maxlength="100" required id="id_first_name">
</p>
<p>
  <label for="id_last_name">Last name:</label>
  <input type="text" name="last_name" maxlength="100" required id="id_last_name">
</p>
<p>
  <label for="id_email">Email:</label>
  <input type="email" name="email" maxlength="320" required id="id_email">
</p>
<p>
  <label for="id_phone">Phone:</label>
  <input type="text" name="phone" maxlength="15" required id="id_phone">
 </p>
```

```
>>> print(u.as_table())
<tr>
   <th><label for="id_first_name">First name:</label></th>
   <td>
    <input type="text" name="first_name" maxlength="100" required id="id_first_name">
 </td>
  </tr>
<tr>
   <th><label for="id_last_name">Last name:</label></th>
   <td>
    <input type="text" name="last_name" maxlength="100" required id="id_last_name">
</td>
  </tr>
<tr>
   <th><label for="id_email">Email:</label></th>
   <td>
   <input type="email" name="email" maxlength="320" required id="id_email">
 </td>
  </tr>
<tr>
   <th><label for="id_phone">Phone:</label></th>
   <td>
    <input type="text" name="phone" maxlength="15" required id="id_phone">
</td>
  </tr>
>>>
```

Form class validations

```
>>> from appone.forms import userprofileform
>>> u=userprofileform()
>>> u.as_valid()
False

>>> u=userprofileform({})
>>> u.is_valid()
False

>>>u=userprofileform({'first_name':'sai','last_name':'kumar','email':'saikumar@gmail.com','phone':'8765432986'})
>>> u.is_valid()
True

>>> u.cleaned_data
{'first_name': 'sai', 'last_name': 'kumar', 'email': 'saikumar@gmail.com', 'phone': '8765432986'}

>>> type(u.cleaned_data['first_name'])
<class 'str'>

>>> type(u.cleaned_data['phone'])
<class 'str'>
>>>
```

Validation errosrs

```
>>> u=userprofileform()
>>> u.errors
{}
>>> u=userprofileform({})
>>> u.errors
{'first_name': ['This field is required.'], 'last_name': ['This field is required.'], 'email': ['This field is required.'], 'phone': ['This field is required.']}
>>>
```

Widgets

forms.py

```
from django import forms
from django.core.validators import RegexValidator

class userprofileform(forms.Form):

 first_name=forms.CharField(max_length=100)
 last_name=forms.CharField(max_length=100)
 email=forms.EmailField()
 phone_regex=r'^\d{1,12}$'
 phone=forms.CharField(validators={RegexValidator[phone_regex]})
 comment=forms.CharField(widget=forms.Textarea)
 LOCATION=[{'HYD','hyderabad'},{'Bnglr','Bangalore'}]
 location=forms.CharField(widget=forms.select(choices="LOCATION"))
 join_date=forms.DateField(widget=forms.SelectDateWidget)
```

views.py

```
def userprofileview(request):

 form=forms.userprofileform()
 if request.method=='POST':
  form=forms.userprofileform(request.POST)
  if form.is_valid():
   print(form.cleaned_data)

return render(request,'userprofile.html',{'form':form})
```

userprofile.html

```
<body>
  <fieldset>
  <legend>fill the form </legend>
  <form action="" method="post">
   {{form.as_p}}
   {% csrf_token %}

   <input type="submit" value="submit">
   </form>
  </fieldset>
  </body>
```

Custom clean Method

forms.py

```
from typing import Any
from django import forms
from django.core.validators import RegexValidator
from django.core.exceptions import ValidationError

class userprofileform(forms.Form):

 first_name=forms.CharField(max_length=100)
 last_name=forms.CharField(max_length=100)
 email=forms.EmailField()
 phone_regex=r'^\d{1,12}$'
 phone=forms.CharField(validators=[RegexValidator(phone_regex)])
 comment=forms.CharField(widget=forms.Textarea)
 LOCATION=[{"HYD","hyderabad"},{"Bnglr","Bangalore"}]
 location=forms.CharField(widget=forms.Select(choices=LOCATION))
 join_date=forms.DateField(widget=forms.SelectDateWidget)

 def clean_first_name(self):
  first_name= self.cleaned_data["first_name"]
  if not first_name.isalpha():
   raise forms.ValidationError("first name should be contain only alphabet")
  return first_name

 def clean_last_name(self):
  last_name= self.cleaned_data["last_name"]
  if not last_name.isalpha():
   raise forms.ValidationError("last name should be contain only alphabet")
  return last_name

 def clean(self):
  cleaned_data=super().clean()
  first_name=cleaned_data.get('first_name')
  last_name=cleaned_data.get('last_name')
  if first_name and last_name:
   if first_name==last_name:
    raise forms.ValidationError("ffirst name and last name cannot be same")

 def clean_email(self):
  email=self.cleaned_data['email']
  if not email.endswith('@mail.com'):
   raise forms.ValidationError("email should be gmail")
  return email
```

Build-in validators

```
from django import forms
from django.core.validators import RegexValidator
from django.core.exceptions import ValidationError
from django.core import validators

class userprofileform(forms.Form):

  first_name=forms.CharField(validators=[validators.MinLengthValidator(3)])
  last_name=forms.CharField(validators=[validators.MinLengthValidator(3)])
  email=forms.EmailField()
  # phone_regex=r'^\d{1,12}$'
  # phone=forms.CharField(validators=[RegexValidator(phone_regex)])
  phone=forms.CharField()
  comment=forms.CharField(widget=forms.Textarea)
  LOCATION=[{"HYD","hyderabad"},{"Bnglr","Bangalore"}]
  location=forms.CharField(widget=forms.Select(choices=LOCATION))
  join_date=forms.DateField(widget=forms.SelectDateWidget)
```

Custom validators

```
from django import forms
from django.core.validators import RegexValidator
from django.core.exceptions import ValidationError
from django.core import validators

# custom validators
def validate_name(value):
  if not value.isupper():
    raise ValidationError(f'{value} is not in uppercase')

class userprofileform(forms.Form):

  first_name=forms.CharField(validators=[validate_name,validators.MinLengthValidator(3)])
  last_name=forms.CharField(validators=[validate_name,validators.MinLengthValidator(3)])
  email=forms.EmailField()
  # phone_regex=r'^\d{1,12}$'
  # phone=forms.CharField(validators=[RegexValidator(phone_regex)])
  phone=forms.CharField()
  comment=forms.CharField(widget=forms.Textarea)
  LOCATION=[{"HYD","hyderabad"},{"Bnglr","Bangalore"}]
  location=forms.CharField(widget=forms.Select(choices=LOCATION))
  join_date=forms.DateField(widget=forms.SelectDateWidget)
```

Function based views

create project and app
configure the app settings,
template settings,
database settings,
static files settings,

models.py (migrations)
forms.py
views.py
urls.py

templates
index.html etc....
static
style.css

```
INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'appone',  # add this line to add the app to the project
```

```
DATABASES = {
 'default': {
  'ENGINE': 'django.db.backends.sqlite3',
  'NAME':'django08_functionbasedviews',
  'USER':'root',
  'PASSWORD':'Swati@123',
 }
}
```

```
DATABASES = {
 'default': {
  'ENGINE': 'django.db.backends.sqlite3',
  'NAME':'django08_functionbasedviews',
  'USER':'root',
  'PASSWORD':'Swati@123',
 }
}
```

```
STATIC_URL = 'static/'
STATICFILES_DIRs=[
  BASE_DIR / 'static'
]
```

Add a little bit of body text

create

views.py

```
def createproduct(request):

  form=productform()  # create an interface of the form
  if request.method == 'POST':   #if the from is submitted
    form=productform(request.POST)
    if form.is_valid():
      form.save(commit=True)
      return HttpResponseRedirect('/')
    return render(request,'create.html',{'form':form})
```

urls.py

```
from django.contrib import admin
from django.urls import path
from appone import views

urlpatterns = [
  path('admin/', admin.site.urls),
  path('',views.getproducts),
  path('product/create/',views.createproduct),
]
```

create.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" ,content="width=device-width", initial="scale=1.0">
    <title>document</title>
  </head>
  <body>
<form method=’POST’>
     <table>
      {{form.as_table}}
      {%csrf.token%}
     </table>
        <input type="submit">
      <.form>
  </body>
</html>
```