Explain (views.py)

The views.py is one of the most essential components of the framework. It acts as the bridge between the model and template.

Functions and roles of "views.py" in Django Project

(I) Handler Http response :

Each view function accepts an Http response & returns an http response,

(II) Retrive & Process Data :

views fetch data from database via models & manipulate it if necessary before rendering.

(III) Rendering templates :

views often used templates to generate dynamic Html content. They pass content data to template for rendering.

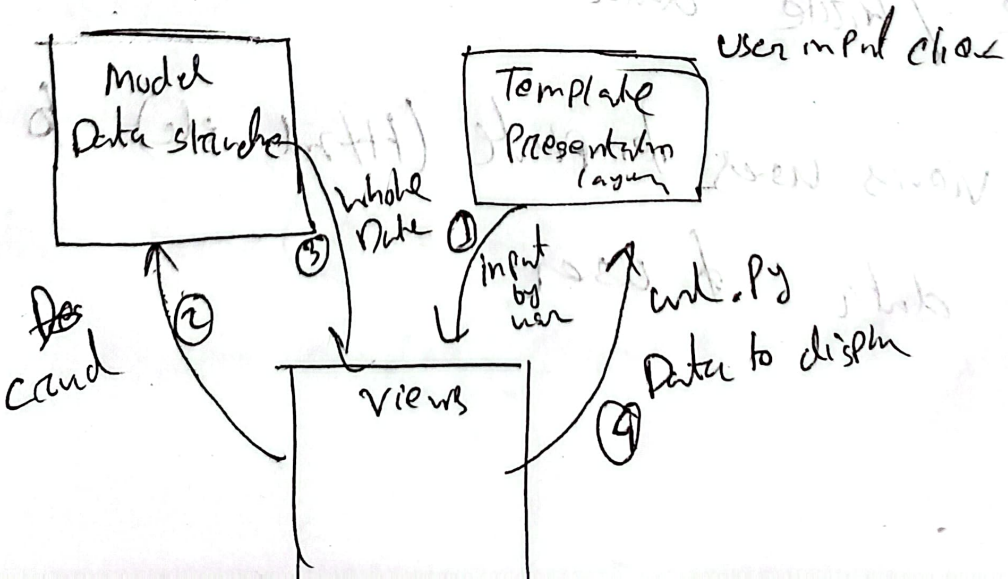
Explain MVT Model

Django follows MVT structure. It's similar to MVC structure. Where Template acts as view and view act as controller.

Model: It acts as interface for maintaing Data, This is a logical data structure behind the app & helps to handle database

Template: It contains the static part of ~~app~~ Html along with special syntax. It's the Presentation layer

View: It's a user interface that communicates with database and model and transfer dat to the template for viewing. Used to receive HttpRequest from clients & return Http response back to client responsible for displaying all/portion of data to user



This how Django handles HTTP request response and generates response through architecture.

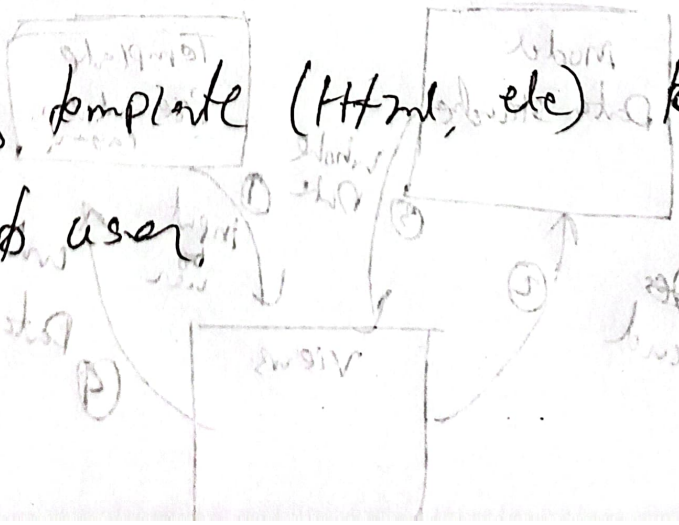
(1) A web browser request page by its url and the web server passes the HTTP request to Dja

(2) Django run through its configured URL Patter it stops at the first one that matches with requ url

(3) Django executes the views that corresponds to the match url.

(4) views also parallely does to data model to retrive / write data to ~~model~~ database.

(5) Then views uses template (Html, etc) to show th retrive data to user.



Home Page Render:

1) First create a Project
   - first install virtual env
   - install Django - pip install Django
   - create a Project by `django-admin startproject Pro`

2) Now create a 'template' Directory in Project to store Html file

3) In "settings.py" of the Project Define template.

```
TEMPLATE = [ DIR [ BASE_DIR ["template"]
```

4) Create a home.html file in the template folder.

5) Create a function in views.py to render homepage.

```
From django.shortcut import render.

def home(render):
    return render(request, 'home.html')
```

6) now import the views and define the url Path in urls.py.

```
From import views
Urls Pattern =[
    Path ("home/", views.home, name = 'home')
```

7) Now ~~run~~ migrate and run server,

```
Python manage.py migrate
Python manage.py makemigration.
Python manage.py runserver
```

Create Superuser:

(i) migrate : python manage.py ~~or~~ migrate,

(ii) Python manage.py ~~makemigration~~ Create superuser.

(iii) to acces superuser,

url admin.

Make Model:

To create a model first we have to create a app. inside the app we get models.py where we can define model.

(I) Python manage.py startapp myapp

(II) Go to settings.py to define the new created app in installed app.

```
INSTALLED APP = [ "myapp", ]
```

(III) Create a urls.py in new app to define the urls.

```
from Django.urls import include, path.

urlspattern = [
        path('myapp', include('myapp.urls')),]
```

(IV) Now Define the model in models.py.

```
from django.db import model

class Product (models.model):
    name = models. Charfield (max-length = 255)
    age = models. floatfield ()

def __str__(self):
    return self.name..
```

(V) Now add model to admin.py.

```
from .model import Product
admin. site. register (Product)
```

(VI) migrate the model

(i) Python manage.py migrate,

(ii) Python manage.py 'make migration

CRUD

To create first we need to make a form.py

(I) Import forms,
Import model class.

```
class ProductForm (forms.ModelForm):
    class Meta = Product
    field = ["name", 'age"]
```

(II) Now in app's views.py create a create_function

```
def create_product (request):
    if request.method == 'Post':
        form = ProductForm (request.POST)
        if form.is_valid():
            form.save()
            return HttpResponse("Success")

    else,
        form = ProductForm()   → instance = P
    return render (request, 'form.html', {
```

(iii) Now Define the function in app's urls.py

```
urlpatterns = [
    path('creat/', views.create, name="create")
```

Delete:

```
def delete(request, p_id):
    p = Product.objects.get(pk=p_id).delete()
    return render(request, "home.html")
```

Handle media

(I) First install a Python Package

Pip install Pillow

(II) Create a model that has image field,

```
Class Photo(models.Model):
    title = models.CharField(max_length=255)
    image = model.ImageField(upload_to='photos/')
    Update = models.DateTimeField(auto_add_add=True)
```

(III) Configure Media in settings,

```
MEDIA_URL = '/media/'
MEDIA_ROOT = OS.Path.Join(BASE_DIR, 'media')
```

(IV) Define url:

```
Path('photo/upload/', views.photo_up, name='image_up')
```

⑤ Define view

Same code as create

⑥ Setup static & media URLs Below url patterns,

if setting. DEBUG

URLPatterns += static(settings. MEDIA_URL, document_root
=setting.me

⑦ Run migration

Python manage.py migrate

Python manage.py make migrations

⑧ python manage.py runserver.

1) List: List is a

5 types of inheritance:

1) Single inheritance: Parent child inheritance.

```
class Parent:
    def hi(self):
        Print (hi)

Cls Child (Parent):
    Pass.

ch = Child()
ch. hi
```

2) Multiple inheritance: Child inherit from 2 or more Parents.

```
Class Parent1:

class Parent2:

class Child (Parent1, Parent2):
```

3) Multi level: Child class is inherit from Parent class another child is inherits from this child class

4) Hierarchical inheritance

Multiple classes are inherit from Parent class

5) Hybrid inheritances

Combination of 2 or more types of inheritance

```
Class A :
    def greet(self):
        print(hi)

Class B(A):
    pass

Class C(A):
    pass

Class D(B,C):
    pass
```

/ এর আগে Position এ Position
এ " পরে Keyword , এ Keyword

①

```
try :
    age = 95
  if age < 0:
  raise value error ("custom message);
 else :
 age = age + 2.

except valueerror as e:
      Print (e)
```