Django

Django :

-Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

-Released in 2005

Why Django :

-With Django, you can take web application from concept to launch in a matter of hours.

- Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel.

- It is free and open source

-Rediculously fast

-Fully Loaded

-Reassuringly secure

-Exceedingly scalable

-Incredibly versatile

Why Named Django :

- It is names after the famous Jazz Guitarist Django Reinhardt

    Django was created by Python Programmers Adrian Holovaty and Simon Willison

Getting Started with Django :

- PROJECT = APPLICATION + SETTING

  We are using PYCHARM COMMUNITY APPLICATION for coding

  First we have to install Django on Pycharm:

- Command for installing Django:
  pip install django
- Command for checking version
  python - m django - - version

HOW TO CREATE DJANGO PROJECT :

- Command:
  django-admin startproject projectname
  e.g. django-admin startproject HDFC
  #here we have taken HDFC as a project name
- When we create project following files are created in it
  __init__.py
  asgi.py
  settings.py
  urls.py
  wsgi.py

TO RUN DJANGO PROJECT (we have to run manage.py file):

- Command
  python manage.py runserver
- This command is to run a emulated server on our local computer. So after running this we can go to
  [localhost:8000] (http://localhost:8000) or

[127.0.0.1:8000] (http://127.0.0.1:8000)

SOME USEFUL TERMS :

1. PORT NUMBER : A port number is the logical address of each application or process that uses a network or the Internet to communicate. A port number uniquely identifies a network-based application on a computer. Each application/program is allocated a 16-bit integer port number. This number is assigned automatically by the OS, manually by the user or is set as a default for some popular applications.

2. URL : A Uniform Resource Locator (URL), otherwise known as a Universal Resource Locator, is the address of a resource on the Internet and the protocol used to access it.

3. IP : An IP address is a unique address that identifies a device on the internet or a local network. IP stands for "Internet Protocol," which is the set of rules governing the format of data sent via the internet or local network.

4. MAC : A media access control address (MAC address) is a unique identifier assigned to a network interface controller (NIC) for use as a network address in communications within a network segment. This use is common in most IEEE 802 networking technologies, including Ethernet, Wi-Fi, and Bluetooth.

HOW TO CREATE PROJECT APPLICATION:

- On command prompt terminal go to directory
  cd projectname
  cd HDFC
- Then create application
  python manage.py startapp appname

e.g. python manage.py startapp loan

here 'loan' is taken as app name

Q. Django follows which structure ?

- MVT model

- M- Model

  V- View

  T- Template

M- Model	V-View	C- Controller
Database Logic E.g. SQL(mangoDB, POST, mysql, mssql, oracle)	Business Logic E.g. Python	Presentation Logic E.g. HTML/JS/CSS/Bootstrap

[# while for .net framework (M-Model V-View C-Controller)]

#SQL is Tabular && NoSQL is unstructured

Data is of Three Types –

1. Structured (table)
2. Semistructured (json , XML, Yaml)
3. Unstructed (paragram)

HOW TO CREATE PROJECT LEVEL URL:

- First create a project:
  Django-admin startproject HDFC
- Then create application:
  Python manage.py startapp loan
- Register app into setting.py
- INSTALLED_APPS = [
      'django.contrib.admin',
      'django.contrib.auth',
      'django.contrib.contenttypes',
      'django.contrib.sessions',
      'django.contrib.messages',
      'django.contrib.staticfiles',
      'loan'
  ]

- Go to files → application 'loan' → views.py
  Write the code :

  from django.http import HttpResponse

  def functionname(r):
        return HttpResponse('<h1>Your Message</h1>')

  e.g.
  from django.http import HttpResponse

  def show(r):

```
        return HttpResponse('<h1>Hello World</h1>')
# show is a function name here
```

- Now we have to create Project Level URL
  Go to urls.py
  It will import two files
  Code:

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

  urlpatterns=[
        path( 'urlpattern/',views.functionname)
  ]

  E.g. Urlpatterns=[
        path( 'show/',views.show)
  ]
  # show is a function name here
  ```

- Now we have to run this program
  Go to Terminal
  python manage.py runserver
- This create server @ http://127.0.0.1:8000
- Click on the link , then type url like
  http://127.0.0.1:8000/show/
- Now we will be able to see the message which we have given in the views.py
  function

HOW TO CREATE APPLICATION LEVEL URL:

- First create a project:

Django-admin startproject HDFC
- Then create application:
  Python manage.py startapp loan
- Register app into setting.py
- INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'loan'
  ]

- Go to files → application 'loan' → views.py
  Write the code :

  from django.http import HttpResponse

  def show(r):
        return HttpResponse('<h1>Hello World</h1>')
  # show is a function name here

- Now we have to create Application Level URL
  For that we have to first create project url and then add application level url
  Copy the urls.py file and paste it in application 'loan'

  Go to project level urls.py
  (Click on import)
  It will import two files, add 'include' after path
  Code:

  from django.contrib import admin
  from django.urls import path,include

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('loan/', include('loan.urls')),
```

loan/ is a project url and here we are calling for application level urls from 'loan' application using 'include' function

- Now go to application level urls.py
  (Click on import)
  It will import two files
  Code:

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

urlpatterns=[
        path( 'show/',views.show)
]
```
show is a function name here

- Now we have to run this program
  Go to Terminal
  python manage.py runserver
- This create server @ http://127.0.0.1:8000
- Click on the link , then type url like
  http://127.0.0.1:8000/loan/show
  #here /loan is project level url and in that we have created /show as application level url
- Now we will be able to see the message which we have given in the views.py function

HOW TO CREATE MULTIPLE APPLICATION LEVEL URLS:

- First create a project:

  Django-admin startproject HDFC

- Then create application:

  Python manage.py startapp loan

- Register app into setting.py

- INSTALLED_APPS = [

  'django.contrib.admin',

  'django.contrib.auth',

  'django.contrib.contenttypes',

  'django.contrib.sessions',

  'django.contrib.messages',

  'django.contrib.staticfiles',

  'loan'

  ]

- Go to files → application 'loan' → views.py

  Here we create 3 functions

  Write the code :

  ```
  from django.http import HttpResponse

  def show(r):
       return HttpResponse('<h1>This is show page</h1>')

  def view(r):
       return HttpResponse('<h1> This is view page </h1>')

  def home(r):
       return HttpResponse('<h1> This is home page </h1>')
  ```

- Now we have to create Application Level URL
  For that we have to first create project url and then add application level url
  Copy the urls.py file and paste it in application 'loan'

  Go to project level urls.py
  (Click on import)
  It will import two files, add 'include' after path
  Code:

  from django.contrib import admin
  from django.urls import path,include

  urlpatterns = [
      path('admin/', admin.site.urls),
      path('loan/', include('loan.urls')),

  # loan/ is a project url and here we are calling for application level urls from 'loan'
  application using 'include' function

- Now go to application level urls.py
  (Click on import)
  It will import two files
  Code:

  from django.contrib import admin
  from django.urls import path
  from loan import views

  urlpatterns=[
        path('show/',views.show)
        path('view/',views.view)
        path('home/',views.home)
  ]

- Now we have to run this program

Go to Terminal

python manage.py runserver

- This create server @ http://127.0.0.1:8000
- Click on the link , then type url like

http://127.0.0.1:8000/loan/show

http://127.0.0.1:8000/loan/view

http://127.0.0.1:8000/loan/home

#here /loan is project level url and in that we have created /show , /view and /home as application level url

- Now we will be able to see the message which we have given in the three functions written in views.py

Problem Statement: If we write html code as we have written in views.py function then will make the code bulky

Solution: Template(

HOW TO CREATE TEMPLATE

- Django template is a text document or a Python string marked-up using the Django template language. Some constructs are recognized and interpreted by the template engine. The main ones are variables and tags.
- A template is rendered with a context. Rendering replaces variables with their values, which are looked up in the context, and executes tags. Everything else is output as is.

1. Create project

     django-admin startproject projectname

2. Create application

     *** change directory to that projectname

```
python manage.py startapp appname1
python manage.py startapp appname2
python manage.py startapp appnameN
```

3. Register appname inside project in settings.py

4. Create folder/directory at project level
   folder name = template

5. Give path inside settings.py in main project
   in line 16
   #Build paths inside the project like this:
   BASE_DIR = Path(__file__).resolve().parent.parent
   TEMPLATE_DIR = Path.joinpath(BASE_DIR,'template')

#here we are creating a path for template folder

6. We **direct** the template under settings.py in main project
*in 'DIRS': [TEMPLATE_DIR],

```
TEMPLATES = [
   {  'BACKEND':'django.template.backends.django.DjangoTemplates',
      'DIRS': [TEMPLATE_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',
```

```
            ],
        },
    },
]
```

6. Then create your html file under template folder

    inside main template folder
        create sub folder
    suppose under main template folder
        we are creating template for appname1, appname2, appname3
template → template_appname1
        template_appname2
        template_appname3

8 In subfolder, we have created in template folder, there we create html file as per our requirement. Like
    Go to template → appname1 → appname1.html
Like wise we can create many files as we want. As
template_appname1 → appname1a.html
        appname1b.html
        appname1N.html
9 Then go to respected app folder
    go to views.py of that app
consider for appname1 application

```
def a1(r):
    return render(r,'template_appname1/appname1a.html')
def a2(r):
    return render(r,'template_appname1/appname1b.html')
def aN(r):
```

```
                    return render(r,'template_appname1/appname1N.html')
```

10. Copy urls.py from main project to every app folder
        inside **urls.py** from specific **appname** folder
        from appname1 import views

        under urlpatterns
                path('name1/' , views.a1),
                path('name2/' , views.a2),
                path('nameN/' , views.aN),

        -similar to other applications

11. Then go to urls.py in main project folder
        write down-   **,include** after path like

```
from django.urls import path,include
```
*in line 17


```
then in urlpatterns = [
        path('admin/' , admin.site.urls),
        path('app1/' , include('appname1.urls')),
        path('app2/' , include('appname2.urls')),
        path('appN/' , include('appnameN.urls')),
        ]
```
12. Now we write code in html files
  - In appname1a.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
    <title>Home</title>
</head>
<body>
<h1 align="center"> Welcome To Appname1 A page </h1>
</body>
</html>
```

-Like for all html files

- Now we have to run this program

  Go to Terminal

  python manage.py runserver

- This create server @ http://127.0.0.1:8000

- Click on the link , then type url like

  http://127.0.0.1:8000/app1/name1

  http://127.0.0.1:8000/app1/name2 http://127.0.0.1:8000/appN/nameN

#here /app1 is project level url and in that we have created name1 , /name2 and /nameN as application level url

- Now we will be able to see the output for the code which we have written in html files.

VIRTUAL ENVIRONMENT

- The virtual environment is an environment which is used by Django to execute an application. It is recommended to create and execute a Django application in a separate environment. Python provides a tool **virtualenv** to create an isolated Python environment. We will use this tool to create a virtual environment for our Django application.

- Creating virtual environment

  python –m venv appname

- Activation

  cd appname

  >.\scripts\activate
- Deactivate

  >deactivate

HOW TO INJECT DYNAMIC CONTENT INTO TEMPLATE USING TEMPLATE TAG

1. Create project

   django-admin startproject HDFC

2. Create application

   *** change directory to that HDFC

   python manage.py startapp loan

3. Register 'loan' inside project in settings.py

4. Create folder/directory at project level

   folder name = template

5. Give path for template inside settings.py in main project

   in line 16

   #Build paths inside the project like this:

   BASE_DIR = Path(__file__).resolve().parent.parent

   TEMPLATE_DIR = Path.joinpath(BASE_DIR,'template')

   #here we are creating a path for template folder

6. We **direct** the template under settings.py in main project
*in 'DIRS': [TEMPLATE_DIR],

```
TEMPLATES = [
   {  'BACKEND':'django.template.backends.django.DjangoTemplates',
      'DIRS': [TEMPLATE_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',
         ],
      },
   },
]
```

6. Then create your html file under template folder

   inside main template folder, we create app folder i.e
   template → loan

8 In subfolder 'loan', we create html file.
loan → businessloan.html

9 Then go to loan app folder
      go to views.py
write the function in views.py
      def businessloan(r):

my_dict = {'name':'Arun', 'id':100}

return render(r,'loan/businessloan.html', context= my_dict)

#context attribute is used to pass dictionary to the html/web page

10. Copy urls.py from main project to every app folder

    inside **urls.py** from specific **loan** folder

in that write code:

from loan import views

urlpatterns=[

    path('business/' , views. businessloan),

]
KANTILAL SIR(SCODEEN GLOBAL)

11. Then go to urls.py in main project folder

    write down-   **,include** after path like

*in line 17

from django.urls import path,**include**

urlpatterns = [

    path('admin/' , admin.site.urls),

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

    ]

12. Now we write code in html files

TEMPLATE TAG : Django's template language comes with a wide variety of built-in tags and filters designed to address the presentation logic needs of your application. Nevertheless, you may find yourself needing functionality that is not covered by the core set of template

primitives. You can extend the template engine by defining custom tags and filters using Python, and then make them available to your templates using the **{% load %}** tag.

{{ }}   -----template tag

- In businessloan.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>BusinessLoan</title>
</head>
<body>
<h1 align="center"> Welcome To Business Loan Page</h1>
{{name}}
{{id}}
</body>
</html>
```

- Now we have to run this program
  Go to Terminal
  python manage.py runserver
- This create server @ http://127.0.0.1:8000
- Click on the link , then type url like
  http://127.0.0.1:8000/loan/business

  #here /loan is project level url and in that we have created /business as application level url
- Now we will be able to see the output for the code which we have written in html files as
  o/p on web browser:
  Welcome To Business Loan Page
  Arun
  100

WORKING WITH STATIC FILES (IMAGES & CSS)

- In this , we use template tag to show images on web and embed css files for designing web pages.
- Code:

1. Create project

    django-admin startproject STATICPRO

2. Create application

    *** change directory to that STATICPRO

    python manage.py startapp staticapp

3. Register 'staticapp' inside project in settings.py

4. Create two folder/directory at project level

    folder name = 'template' and 'static'

5. Give path for template and static inside settings.py in main project
    in line 16
    #Build paths inside the project like this:
    BASE_DIR = Path(__file__).resolve().parent.parent
    TEMPLATE_DIR = Path.joinpath(BASE_DIR,'template')
    STATIC_DIR = Path.joinpath(BASE_DIR,'static')

    #here we are creating a path for template and static folder

      6. We **direct** the template and static under settings.py in main project

    1] for template write **'DIRS': [TEMPLATE_DIR]**

    TEMPLATES = [

```
{ 'BACKEND':'django.template.backends.django.DjangoTemplates',
   'DIRS': [TEMPLATE_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',
       ],
   },
 },
]
```

    2] for static write

        STATIC_URL = 'static/'
        STATICFILES_DIRS = [STATIC_DIR]

7. Then create your html file under template folder

    inside main template folder, we create app folder i.e
  template →staticapp

8. In subfolder 'staticapp', we create html file.
  loan → home.html

9. In 'static' folder create two folder/directory named as 'images' and 'css'
  -like -> static →images

css

images folder is used to store images while in css we put the css code

Take one image and paste it in 'images' folder. Lets say we have image as 'abc.jpeg'

10.   Then go to staticapp app folder
          go to views.py
write the function in views.py
          def home(r):
              return render(r,'staticapp/home.html')

11.   Copy urls.py from main project to every app folder
          inside **urls.py** from specific **staticapp** folder
in that write code:

from staticapp import views

urlpatterns=[
              path('home/' , views.home),

]

11. Then go to urls.py in main project folder
       write down-   **,include** after path like
*in line 17
from django.urls import path,**include**

urlpatterns = [
              path('admin/' , admin.site.urls),
              path('staticapp/' , include('staticapp.urls')),

]

12.   Now we write code in html files

   - In home.html

```html
<!DOCTYPE html>
{% load static%}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>BusinessLoan</title>
</head>
<body>
<h1 align="center"> Welcome To Home  Page</h1>
<a href="{% static 'images/abc.jpeg' %}">
</body>
</html>
```

#Here we load our static contents i.e. image or css (here we have taken image) as **{% load static%}** below the DOCTYPE html. Then we have written anchor tag and given path of image like
<a href="{% static 'images/abc.jpeg' %}">

13.   Now we have to run this program
   Go to Terminal
   python manage.py runserver

14.   This create server @ http://127.0.0.1:8000

15.   Click on the link , then type url like
   http://127.0.0.1:8000/staticapp/home

   #here /staticapp is project level url and in that we have created /home as application level url.

Now we will be able to see the output for the code which we have written in html files as

NOTE:

If we want to load css in html code, we write a css code in file have extension .css and put that file in static css folder.

Then we have to give the path of that css file to apply css on our html page. Syntax is as:(say we have css named test.css)

</title>

<link rel="stylesheet" href="{% static 'css/test.css' %}"

HOW TO DUMP DATA INTO DATABASE

In django project when we want to connect to database , it need driver i.e. connector.

We can use mssql or mysql or any database

Installation for Databases:

1. **For mysql:**
   MYSQL APPLICATION
   in pycharm:
   pip install mysql

2. **For mssql:**
   MSSQL APPLICATION
   in pycharm:
   pip install django-mssql
   pip install mssql-django

How to Configure Database:

1. Create one project:

    django-admin startproject projectname

    django-admin startproject dbproject

2. Change directory:

    cd dbproject

3. Then create application under our project

    python manage.py startapp appname

    python manage.py startapp dbapp

    register your app in settings.py

4. Create database in mssql/mysql:

  1. For mysql:

    open mysql command line client

    enter your password entered at the time of installation

    then enter

            show databases;

    #it will show inbuilt databases;

    Now we have to create our database

        create database database_name;

        create database dbmarch;

2. . For mssql:

open mssql application

select new query
then type

show databases;

#it will show inbuilt databases;

Now we have to create our database

create database database_name;

create database dbmarch;

3. Now go to settings.py in project file:
Configure database here

For mysql:

DATABASES = {
 'default': {
 'ENGINE': 'django.db.backends.mysql',   * it is your database name might be mssql, mysql etc
 'NAME': 'nameofdatabase', * insert your database name
 'USER': 'root',   * its your user name while installation of mysql
 'PASSWORD':'PASSWORD'   * its your password of mysql installation
   }
}

For mssql:
DATABASES = {
   'default': {
      'ENGINE': 'mssql',

```
        'NAME': 'mydb',   * insert your database name
        'USER': '',          * set blank
        'PASSWORD': '',      * set blank
        'HOST': 'DESKTOP-1J4UDBO',   *Host name of mssql database
        'PORT': '',     * set blank(optional)


    },
}
```

That's all the step to configure Database in django

How to Check Connection of Database:

In Command Prompt of Pycharm:

Run command:

python manage.py shell

Method 1

In [1]: from django.db import connection

In [2]: c = connection.cursor()

In [3]: exit()

Method 2

in InteractiveConsole

>>> from django.db import connection

>>> c = connection.cursor()

>>> exit()

How to Convert Python Code to SQL:

1. First we create Model
    For that we write code in model.py file in application of project we have created
    In models.py:

    from django.db import models

    # Create your models here.
    class Employee(models.Model):
        name=models.CharField(max_length=30)
        age = models.IntegerField()

    #Here we have created two fields for data storage in database i.e name and age

    Now we have to convert this python code into sql and then it will create two
    columns in database

1. to convert py file in sql
        python manage.py makemigrations
2. to create table in sql
        python manage.py migrate

How to Check Tables in Databases:

1 command

  to select database

        use databasename;

  to see table inside this

show tables;

to access specific table

desc tablename;

CREATING ADMIN PANEL:

1 Create Project

django-admin startproject projectname

cd projectname   (change directory)

2 create app

python manage.py startapp appname

register your app in installed apps

3 Create template at project level

register the template

TEMPLATE_DIR = Path.joinpath(BASE_DIR, 'template')

4 Register your database

a)mysql

DATABASES = {

  'default': {

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

    'NAME': 'dbmarch',

```
        'USER': 'root',

        'PASSWORD': 'password'

    }

}


b)mssql

DATABASES = {

    'default': {

        'ENGINE': 'mssql',

        'NAME': 'dbmarch',

        'USER': '',

        'PASSWORD': '',

        'HOST': 'DESKTOP-1J4UDBO'

    }

}
```

5 To check connection is successfull

insert in terminal

python manage.py shell

In [1]: from django.db import connection

In [2]: c = connection.cursor()

In [3]: exit()

or

in InteractiveConsole

>>> from django.db import connection

>>> c = connection.cursor()

>>> exit()

that means connection is successfully done

6 Create class under models.py of application

from django.db import models

Create your models here.

class Feedback(models.Model):

    name = models.CharField(max_length=100)

    date_of_Birth = models.DateField()

    mobile_no = models.CharField(max_length=10)

7 Then go to admin.py for access the admin panel

from django.contrib import admin

from .models import Feedback

Register your models here.

class FeedAdmin(admin.ModelAdmin):

    list_display = ['name', 'date_of_Birth', 'mobile_no']

admin.site.register(Feedback, FeedAdmin)

8 go to views.py of application

from django.shortcuts import render

from .models import Feedback

Create your views here.

```
def view(r):
    form = Feedback.objects.all()     #to access your data from admin panel
    my_dict =        {'form': form}
    return render(r, 'asd1.html', context=my_dict)
```

9 Create your html files for respective to collect data and send data to front end

a) to view that feeded data to front(asd1.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```html
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <table border="1px">
        <tr>
<th> Name</th>
<th>Date of Birth</th>
<th>Mobile Number</th>
<!--<th>{{d.image}}</th> -->
    </tr>
{%if form %}
{% for d in form %}
        <tr>
<th> {{d.name}}</th>
<th>{{d.date_of_Birth}}</th>
<th>{{d.mobile_no}}</th>
<!--<th>{{d.image}}</th> -->
    </tr>
{%endfor%}
{% endif %}
</table>
</body>
</html>
```

KANTILAL SIR(SCODEEN GLOBAL)

10 create your urls project level or app level

   Here we create project level url

   from django.contrib import admin
   from django.urls import path
   from appname import views

   urlpatterns = [
       path('admin/', admin.site.urls),
       path('home/', views.view),

11 After creating class you have to change that python code into sql

       python manage.py makemigrations

KANTILAL SIR(SCODEEN GLOBAL)

12 To create sql data to table

       python manage.py migrate

13 To create Admin for username and password

insert in terminal

       python manage.py createsuperuser

       Username(): admin

       Email address: abc@abc.com

       Password:

       Password(again):

       Bypass password validation and create user anyway<[y/N]: y

Superuser created successfully

14 runserver

    python manage.py runserver

15 Go to admin url i.e. http://127.0.0.1:8000/admin

    Enter data manually from admin panel and view data on /home url

    http://127.0.0.1:8000/home

CREATING FORMS:

1 Create Project

    django-admin startproject projectname

cd projectname   (change directory)

2 create app

        python manage.py startapp appname

    register your app in installed apps

3 Create template at project level

    register the template

    TEMPLATE_DIR = Path.joinpath(BASE_DIR, 'template')

4 Register your database

a)mysql

```
DATABASES = {

    'default': {

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

        'NAME': 'dbmarch',

        'USER': 'root',

        'PASSWORD': 'password'

    }

}
```

b)mssql

```
DATABASES = {

    'default': {

        'ENGINE': 'mssql',

        'NAME': 'dbmarch',

        'USER': '',

        'PASSWORD': '',

        'HOST': 'DESKTOP-1J4UDBO'

    }

}
```

5 To check connection is successfull

insert in terminal

python manage.py shell

In [1]: from django.db import connection

In [2]: c = connection.cursor()

In [3]: exit()

or

in InteractiveConsole

>>> from django.db import connection

>>> c = connection.cursor()

>>> exit()

that means connection is successfully done

6 Create class under models.py of application

KANTILAL SIR(SCODEEN GLOBAL)

from django.db import models

Create your models here.

class Feedback(models.Model):

    name = models.CharField(max_length=100)

    date_of_Birth = models.DateField()

    mobile_no = models.CharField(max_length=10)

7 Create forms.py at application level and fetch the class info you created in models.py

from django import forms

```python
from .models import Feedback


class FeedForm(forms.ModelForm):
    class Meta:
        model = Feedback
        fields = '__all__'
```

8 go to views.py of application

```python
from django.shortcuts import render
from .forms import FeedForm
from .models import Feedback
```

```python
# Create your views here.
def show(r):
    form = FeedForm
    my_dict = {'form': form}
    if r.method == 'POST':                      #its for sending yourdata to database
        form = FeedForm(r.POST)
        if form.is_valid():
            form.save(commit=True)              #to save your data in database
    return render(r, 'asd1.html', context=my_dict)
def view(r):
    form = Feedback.objects.all()     #to access your data from admin panel
```

```
    my_dict =        {'form': form}

    return render(r, 'asd2.html', context=my_dict)
```

9 Create your html files for respective to collect data and send data to front end

a) for post method html file (asd1.html)

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Title</title>

</head>

<body>
```

KANTILAL SIR(SCODEEN GLOBAL)

```
<h1> welcome to forms</h1>

<form method = "POST">

    {{form.as_p}}

    {%csrf_token%}

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

</form>

</body>

</html>
```

b) to view that feeded data to front(asd2.html)

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <table border="1px">
        <tr>
<th> Name</th>
<th>Date of Birth</th>
<th>Mobile Number</th>
<!--<th>{{d.image}}</th> -->
    </tr>
 {%if form %}
{% for d in form %}
        <tr>
<th> {{d.name}}</th>
<th>{{d.date_of_Birth}}</th>
<th>{{d.mobile_no}}</th>
<!--<th>{{d.image}}</th> -->
    </tr>
{%endfor%}
{% endif %}
```

```
</table>

</body>

</html>
```

10 create your urls project level or app level

  Here we create project level url

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.show ),
    path('view/', views.view ),
```

KANTILAL SIR(SCODEEN GLOBAL)

11 After creating class you have to change that python code into sql

    python manage.py makemigrations

12 To create sql data to table

    python manage.py migrate

13 runserver

    python manage.py runserver

14 Go to url i.e. http://127.0.0.1:8000 to view from

  Enter data manually and submit data and view it on /home url i.e.

  http://127.0.0.1:8000/home

ADVANCE TEMPLATE:

In advanced template, we create base template and extends it in other html files.

1 Create project

      django-admin startproject advtemplate

*** change directory to that projectname

2 Create application

      python manage.py startapp advapp

3 write down advapp inside project

      settings.py

      register appname

4 Create folder at project level

      folder name = template

5 Give path inside settings.py in main project

      in line 16

               # Build paths inside the project like this: BASE_DIR / 'subdir'.

               BASE_DIR = Path(__file__).resolve().parent.parent

create under this TEMPLATE_DIR = Path.joinpath(BASE_DIR,'template')

*template = template folder

6 write down that direction under settings.py in main project

*in 'DIRS': [TEMPLATE_DIR,],

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
*   'DIRS': [**TEMPLATE_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',
            ],
        },
    },
]

7 then create your html file under template folder

        inside main template folder

                create sub folder

we are creating template for app i.e advapp

8 create name.html file as per your requirement

Then enter your text body inside your html file

suppose we are creating for advapp

base.html

home.html

contactus.html

9 Then go to app folder

go to views.py of advapp

def home(r):

return render(r,advapp/home.html')

def contact(r):

return render(r, advapp/contactus.html')

10 copy urls.py from main project to app folder

inside urls.py from advapp folder

from advapp import views

under urlpatterns

path('' , views.home),

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

11 then go to urls.py in main project folder

write down-   ,include in front of from django.urls import path,include

from django.urls import path,include   *in line 17

then in urlpatterns = [

path('admin/' , admin.site.urls),

path('' , include('advapp.urls')),

]

12 Go to template → advapp → base.html

when creating base file we write base block in the body and write code to show the contents which we want on every page.

It is mandatory to write base block in base.html as :

```
{% block base_block %}
{% endblock %}
```

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Navigation</title>
    <style>
        .navbar{
            background-color: black;
            border-radius: 30px;

        }
        .navbar ul{
            overflow: auto;
        }
         .navbar li{
            float:left;
            list-style: none;
            margin: 13px 20px;

        }
        .navbar li a{
            padding: 15px 15px;
            text-decoration: none;
            color: white;
```

```
        }
        .navbar li a:hover{
            color: grey
        }
        .search{
            float: right;
            color: white;
            padding: 12px 75px;
        }
        .navbar input{
            border: 2px solid black;
            border-radius: 14px;
            padding: 3px 17px;
            width: 129px;
        }
    </style>
</head>

<body>
    <header>
        <nav class="navbar">
            <ul>
                <li><a href="/">Home</a></li>
                <li><a href="/contact">Contact Us</a></li>
            </ul>
        </nav>
    </header>
{% block base_block %}
{% endblock %}
</body>

</html>
```

13 Go to template → advapp → home.html

When we extend base in other html pages we have to write code in base block as:

```
{% block base_block %}
#YOUR CODE HERE
{% endblock %}
```

```
<!DOCTYPE html>
{% extends 'base.html' %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{% block base_block %}
<h1>WELCOME TO HOME PAGE</h1>
{% endblock %}
</body>
</html>
```

14 Go to template → advapp → contact.html

```html
<!DOCTYPE html>
{% extends 'base.html' %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{% block base_block %}
<h1>WELCOME TO CONTACT PAGE</h1>
{% endblock %}
</body>
</html>
```

15 Now we have to run this program

Go to Terminal

python manage.py runserver

16.   This create server @ http://127.0.0.1:8000

17.   Click on the link , then type url like

http://127.0.0.1:8000/advapp/home

http://127.0.0.1:8000/advapp/contact

and check the output on the web browser

USER AUTHENTICATION:

   User authentication verifies the identity of a user attempting to gain access to a network or computing resource by authorizing a human-to-machine transfer of credentials during interactions on a network to confirm a user's authenticity.

Here we give authentication using login and logout to create sessions.

We have to made few changes to create user authentication i.e for login and logout.

FOR LOGIN

1. In views.py :

Here we have to import one library and then write '@login_required' before the function definition for where we want to give authentication.
As:

from django.contrib.auth.decorators import login_required

@login_required
def show(r):
    {block of code}   #just an example

2. In urls.py
a. Project level:
    Include path as :
    Path('accounts/', include('django.contrib.auth.urls'))

3. Create project level directory names as 'registration' like we create template directory, in that create login.html file
    Registration → login.html

    In login.html(extend base is optional) :

```
<!DOCTYPE html>
{% extends 'base.html' %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{% block base_block %}
<h1>WELCOME TO LOGIN PAGE</h1>
<form method ="POST">
    {{form.as_p}}
    {%csrf_token%}
```

```
<button type ="submit" class="btn btn-primary">LOGIN</button>

</form>
{% endblock %}
</body>
</html>
```

FOR LOGOUT

1. Create a tab for logout in base.html:

   `<a href='/accounts/logout/'>LOGOUT</a>`

2. Create logout.html in template directory:

   In logout.html

```
<!DOCTYPE html>
{% extends 'base.html' %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>THANKS FOR VISITING</h1>
<a href="accounts/login" >LOGIN</a>
</body>
</html>
```

3. In views.py , create function for logout:

```
def logout(r):
    return render(r, 'logout.html')
```

4. Create application level url:
   path('logout/',views.logout)

5. In settings.py:
   LOGIN_REDIRECT_URL = "/home"   #we can redirect login page here(login)
   LOGOUT_REDIRECT_URL = '/logout'

   #WE CAN ADD THIS IN ADVANCE TEMPLATE PROJECT

SIGNUP FORM:

1. For Creating signup form , we already have a inbuilt model named 'User'
   So no need to create model.py
2. So we directly jump on forms.py

   ```
   from django import forms
   from django.contrib.auth.models import User

   class SignupForm(forms.ModelForm):
       class Meta:
           model = User
           fields = ['username', 'password', 'first_name', 'last_name', 'email']
   ```

3. In views.py

   ```
   from .forms import signupform
   ```

```
from django.http import HttpResponseRidirect

def signup(r):
    form = SignupForm()
    if r.method == 'POST':
        form = SignupForm(r.POST)
        if form.is_valid():
            user = form.save()
            user = set_password(user.password)
            user.save()
```

4. Create url for signup
   path('signup/',view.signup)

5. Create tap for signup in base.html:
   LOGOUT

6. Now we have to run this program
   Go to Terminal
   python manage.py runserver

7. This create server @ http://127.0.0.1:8000

ORM QUERY:

Object-relational mapping (ORM) is a programming technique in which a metadata descriptor is used to connect object code to a relational database. ORM converts data between type systems that are unable to coexist within relational databases and OOP languages.

Orm is mapping of SQL to Python. It is written in views.py to fetch data from database

ORM QUERIES:

#Here Product is a model class name and here we are fetching data from database.

gt-greater than

gte-greater than equals to

lt- less than

lte- less than equal to

data = Product.objects.all()   ----all data

data = Product.objects.filter(qnty__gt=20)   ----quantity greater than 20

data = Product.objects.filter(qnty__exact=20)   ----quantity exactly 20

data = Product.objects.filter(qnty__lt=20)   ----quantity less than 20

data = Product.objects.filter(name__icontains='S') ----name contain S(case insensetive)

data = Product.objects.filter(id__in=[1,2,3]) ---ids 1 2 3 (ascending)

data = Product.objects.filter(name__endswith='d') ---name ends with d

data = Product.objects.filter(name__startswith='s') & Product.objects.filter(qnty__gt=5)   ---- name start with 's' (case sensitive) and quantity greater than 5

data = Product.objects.exclude(name__startswith='s')   name starts with s(case sensetive)

data = Product.objects.exclude(name__istartswith='s')   name starts with s(case insensetive)

```
data =  Product.objects.all().order_by('-qnty')[0:3]
```

CRUD OPERATION:

C- Create

R- Read/Retrieve

U- Update

D- Delete

To create project for execution of crud operation, follow the step

1. Create project - crudproject
2. Create app – crudapp
3. Register app in settings.py
4. Create template and create template path
5. Add html files in template
a. showform.html
b. viewdata.html
c. update.html
6. Configure Data in settings.py
7. In models.py

```
from django.db import models

# Create your models here.
class Employee(models.Model):
    eid = models.IntegerField(primary_key=True)
    ename = models.CharField(max_length=30)
    ecity = models.CharField(max_length=30)
    email = models.EmailField()
    date = models.DateTimeField(auto_now_add=True)
```

8. In forms.py

```python
from django import forms
from .models import Employee
from django.contrib.auth.models import User

class EmpForm(forms.ModelForm):
    class Meta:
        model = Employee
        fields = '__all__'

class SignupForm(forms.ModelForm):
    class Meta:
        model = User
        fields = ['first_name','last_name','email','username','password']
```

9. In admin.py

```python
from django.contrib import admin
from .models import Employee

# Register your models here.
class EmpAdmin(admin.ModelAdmin):
    list_display = ['eid','ename','ecity','email','date']


admin.site.register(Employee, EmpAdmin)
```

10.   In views.py

```python
from django.shortcuts import render
from .forms import EmpForm, SignupForm
from .models import Employee
from django.http import HttpResponse, HttpResponseRedirect
# Create your views here.

def index(r):          #function to view form
    form = EmpForm
    if r.method=='POST':
        form = EmpForm(r.POST)
        if form.is_valid():
            form.save(commit=True)
            return HttpResponseRedirect('/show/')
    return render(r,'crudapp/index.html', {'form':form})

def show(r):            #function to view data
    data = Employee.objects.all()
    return render(r,'crudapp/show.html', {'data':data})

def delete(r,id):          #function to delete data
    data = Employee.objects.get(eid=id)
    data.delete()
    return HttpResponseRedirect('/show/')

def update(r,id):              #function to update data
```

```
    data = Employee.objects.get(eid=id)
    if r.method=='POST':
        if r.POST['eid']==str(data.eid):
            form = EmpForm(r.POST,instance=data)
            if form.is_valid():
                form.save(commit=True)
                return HttpResponseRedirect('/show')
        else:
            return HttpResponse('RECORD NOT FOUND <a href="/show">GO BACK</a>')
    return render(r,'crudapp/update.html',{'data':data})
```

11.   Create URL:

a. Project Level Url :

```
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('crudapp.urls')),
]
```

b. Application level url-

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

urlpatterns = [
    path('', views.index),
    path('show/', views.show),
    path('delete/<id>', views.delete),
    path('update/<id>', views.update),
]
```

12.   Create HTML Pages:

a. Showform.html:

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{% block base_block%}
<div align="center">
<h1>FILL YOUR DATA HERE</h1><br><br>
    <form method="Post">
        {{form.as_p}}
        {%csrf_token%}
        <button type="submit" class="btn btn-primary btn-lg">Submit</button>
    </form>
    </div>
{% endblock %}
```

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

b. Viewdata.html:

```html
<!DOCTYPE html>
{% extends 'crudapp/base.html' %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{% block base_block%}
<div align="center">
<h1 >
    --Check Your Details Here--</h1>
    <table  border="1px" style="color:black;" >
        <tr >
    <th style="color:black; padding:15px;">Employee Id</th>
    <th >Name</th>
    <th >City</th>
    <th >Email</th>
    <th>Date</th>
     <th>Update / Delete</th>
  </tr>
    {% if data %}
    {% for d in data %}
        <tr >
    <th style="color:black; padding:15px;">{{d.eid}}</th>
    <th>{{d.ename}}</th>
    <th> {{d.ecity}}</th>
    <th >{{d.email}}</th>
    <th >{{d.date}}</th>
    <th ><a href="/update/{{d.eid}}">Update</a> / <a href="/delete/{{d.eid}}">Delete</a> </th>
        </tr>
    {% endfor %}
    {% endif %}
     </table>
    </div>
{% endblock%}
</body>
</html>
```

c. Update.html:

```html
<!DOCTYPE html>
{% extends 'crudapp/base.html' %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{% block base_block%}
```

```
<div align="center">
<h1>UPDATE YOUR DATA</h1><br><br>
<form method="post">
    Employee ID : <input type="text" name="eid" value="{{data.eid}}"><br><br>
    Employee NAME : <input type="text" name="ename" value="{{data.ename}}"><br><br>
    Employee CITY : <input type="text" name="ecity" value="{{data.ecity}}"><br><br>
    Employee EMAIL : <input type="text" name="email" value="{{data.email}}"><br><br>
{%csrf_token%}
    <button type="submit" class="btn btn-primary btn-lg">UPDATE</button>
</form>
</div>
{% endblock %}
</body>
</html>
```

13.   Now we have to run this program

Go to Terminal

python manage.py runserver

14.   This create server @ http://127.0.0.1:8000

15.   Now we have to enter data manually through form or admin panel and then we will able to see the data on viewdata.html page. We can now perform ORM queries on data.