DJANGOFRAMEWORK

Class 1 : Introduction of Django Framework

Framework: Framework is a fundamental structure for developing a software projects. It provides set of predefined tools and libraries that streamline and organize the projects.

Example : 1. Django is used for web development in python.
2. React is used for building user interfaces in javascript.

Django: Free and open source framework for building web apps with python.

It's not only one framework in python:

- Django
- Flask
- Tornado
- Bottle
- Falcon
- Hug

Django is a popular one because it builds a website in less time and less code.

That's why, many companies are using Django like:

1. You Tube
2. Instagram
3. Spotify
4. Drop box

Django we can also call it as, Batteries included framework which means lot of features are out of the box. So, we don't want to code them from the scratch.

Django features:

- Admin site --> To managing a data. It is a huge time saver.
- Object-Relational Mapper(ORM) --> It abstract the database so we can code this process the data without writing a lot of SQL queries.
- Authentication --> It is used for validation purpose.

Here, Django comes with a lot of features...

But you don't have to learn and use them all. Because, all these features are all optional.

Django framework: Django is a python based high level web framework and mainly is used to develop a rapid, flexible and complex web application. Which is used **MVT design pattern**.

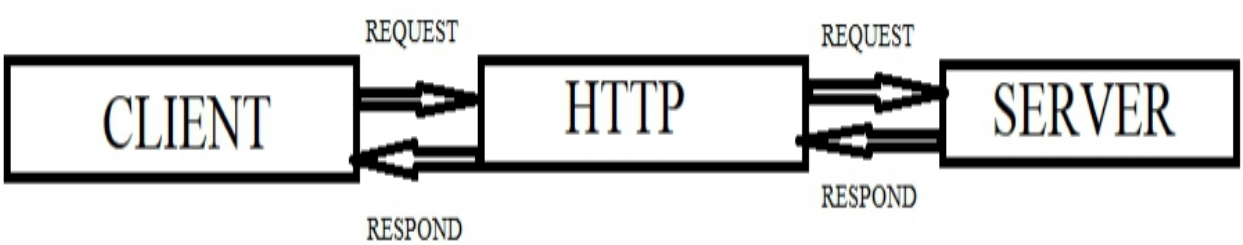
Django framework was developed by **Adrian Holovaty.**

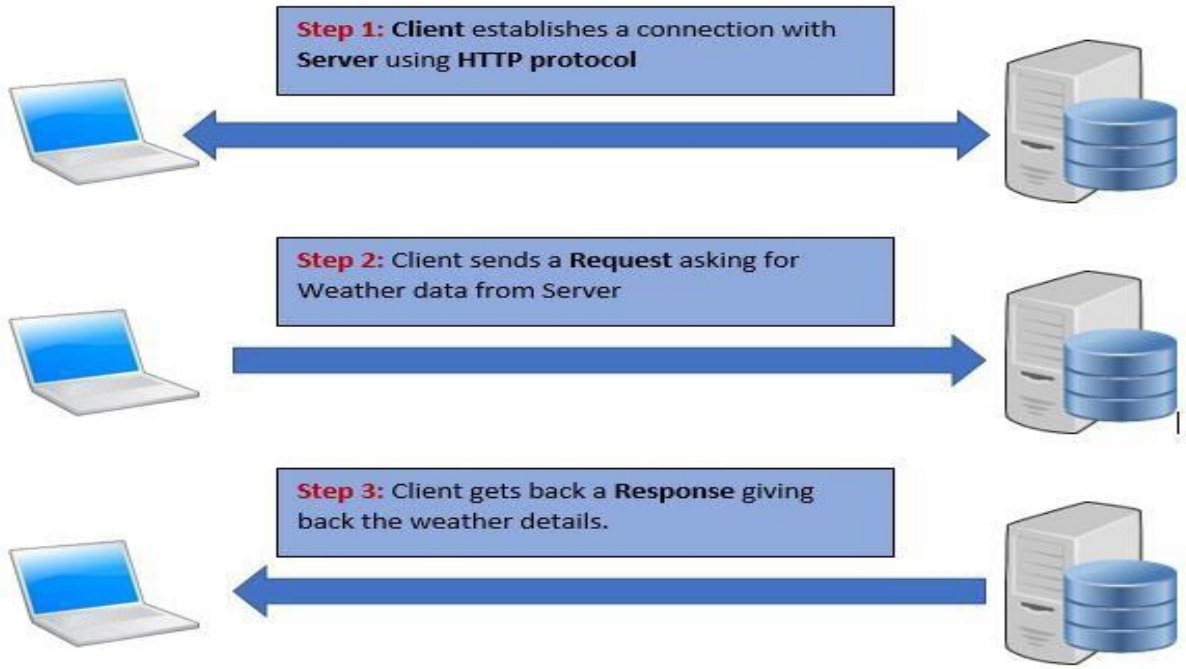
Client server architecture: **HTTP**

Step1: Client establish a connection with server using Http Protocol.

Step 2: Client sends a request asking for data from server.

Step 3: Client gets back a response giving back the details.





Pip(Preferred install Program):

---->Goto search/type cmd

C:\Users\KITS>**mkdir DjangoApplications**

C:\Users\KITS>**cd DjangoApplications**

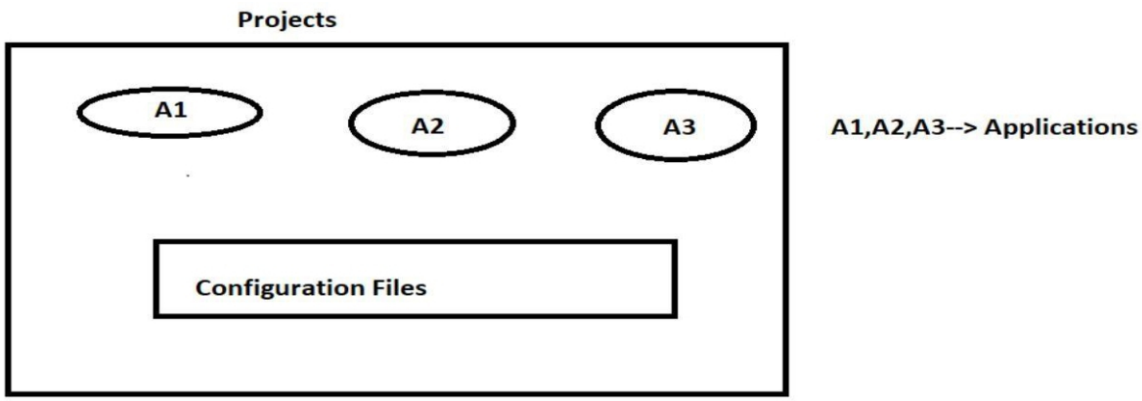
C:\Users\KITS>**DjangoApplications>cd..**

C:\Users\KITS>**pip install Django**

C:\Users\KITS>**django-admin –version**

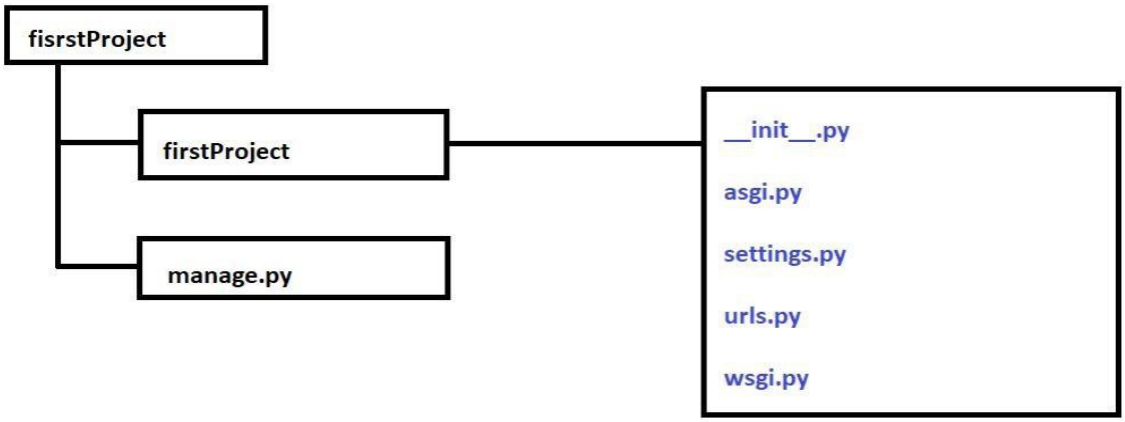
C:\Users\KITS>**cd DjangoApplications**

Django Projects and Applications:



Step 1: Create project using following command.

django-admin startproject firstProject



1. **__init__.py :** initialize the package.
2. **Asgi.py:** asgi (Asynchronous server gateway interface).

   this file is responsible for configuring the asgi application.
3. **Se翻ng.py**This is the file is used to configure an application.

This file contains following entries.

- Installed_apps: We will add our application in this entry.
- Middleware: used as processing unit between client and server.
- Templates: is used to render the response to end user.
- Database: used to specify database. By default Django provides a database called db.sqlite3.
- Auth_password_validator: used for Validation purposes.

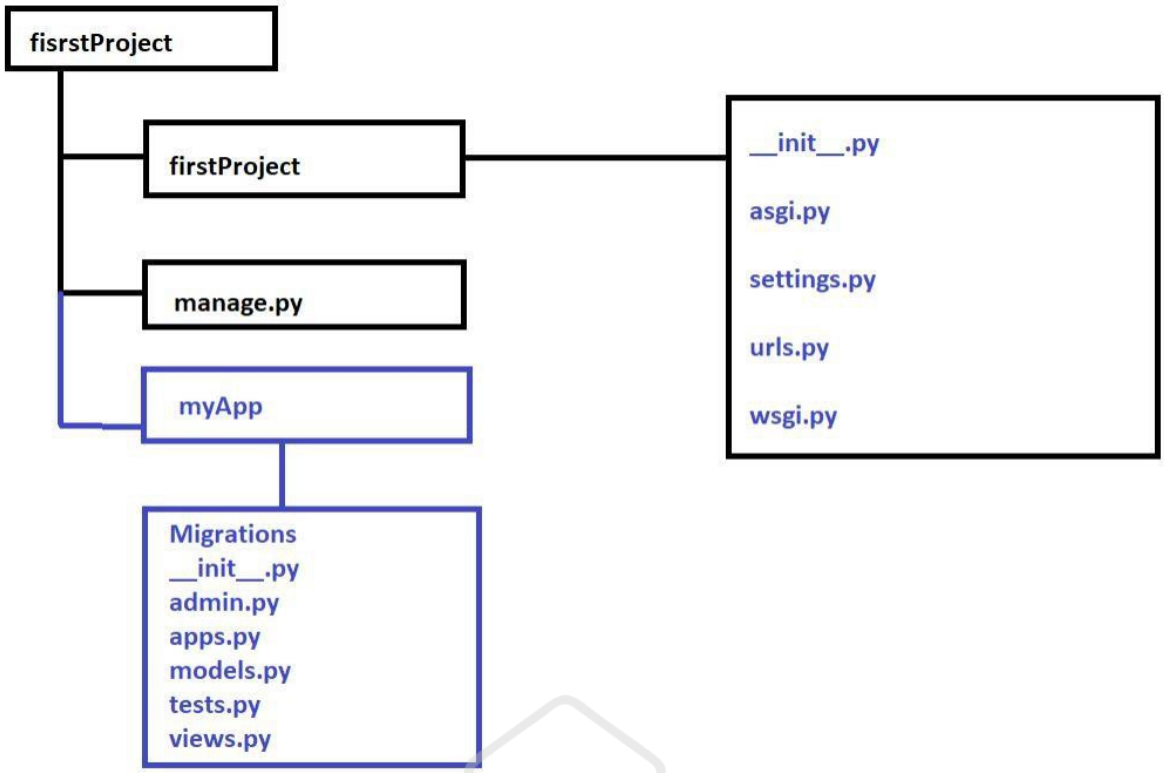
4. Urls.py: contains URL pattern CORRESPONDING to each view.
5. Wsgi.py: wsgi (Web server gateway interface) it is a standard component of django configured to use wsgi.

It serves to serve your projects.

Step 2: Create an application using following command.

C:\Users\KITS**DjangoApplications>cd firstProject**

python manage.py startapp myApp



Migration folder is used for database specific informations**.**

Admin.py: is used to register our application to admin interface.

Apps.py: Application specific configuration.

Models.py: we store application specific data models and it is used for database operations.

Tests.py: used to test our applications.

Views.py : used to write business logic or functions

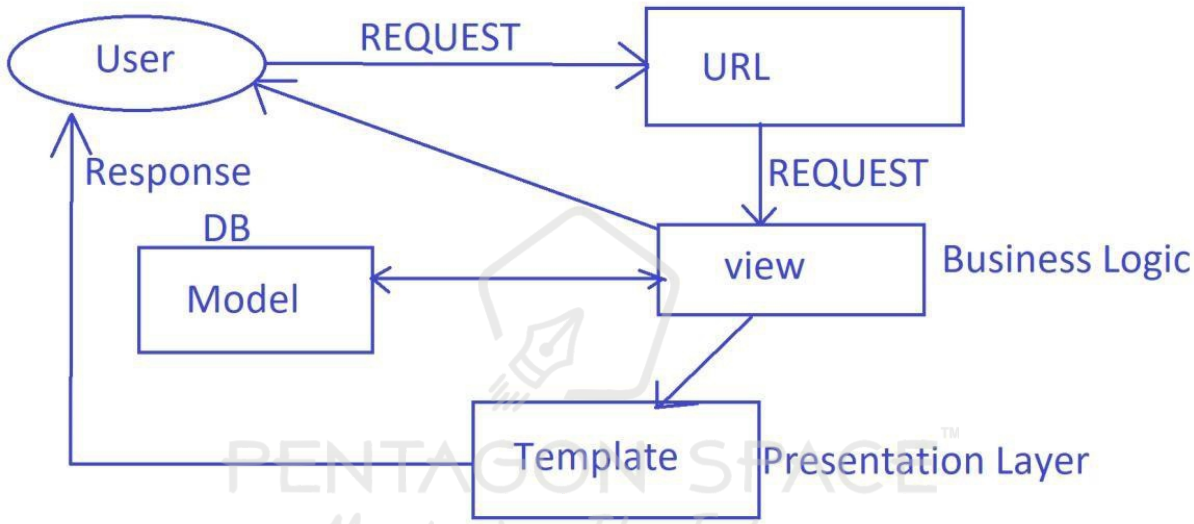
Class 2 : MVT Architecture

MVT is a **software design pattern** and it has 3 components **Model, view, Template.**

Model: The Model helps to **handle the database**.

Template: **Handles user interface**.

View: Execute the Business logic and **interact with the model**.



How to Run Django Server?

http://127.0.0.1:8000

127.0.0.1 Local Host(Base URL)

This url along with endpoint should be loaded from the browser

http://localhost:8000

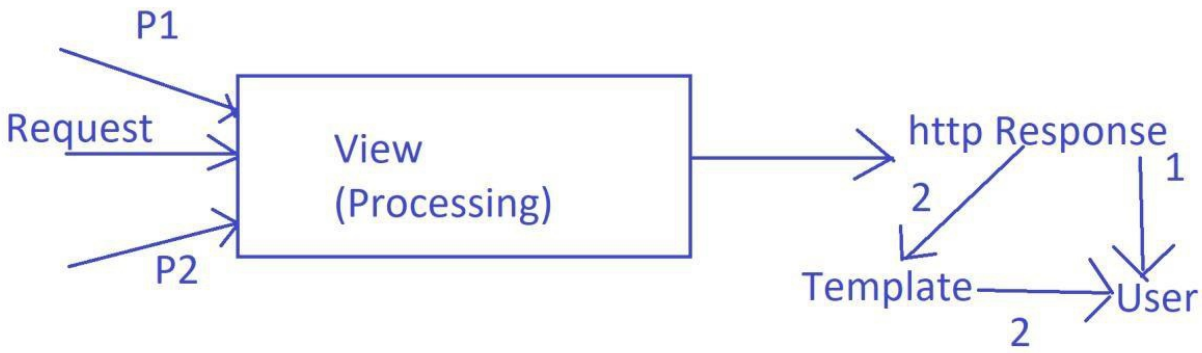
8000→ Default port number to run Django Applications

80**00**→ Default start number

Port Number	Protocol	Application
20	TCP	FTP data
21	TCP	FTP control
22	TCP	SSH
25	TCP	SMTP
53	UDP, TCP	DNS
80	TCP	HTTP (WWW)
110	TCP	POP3
443	TCP	SSL

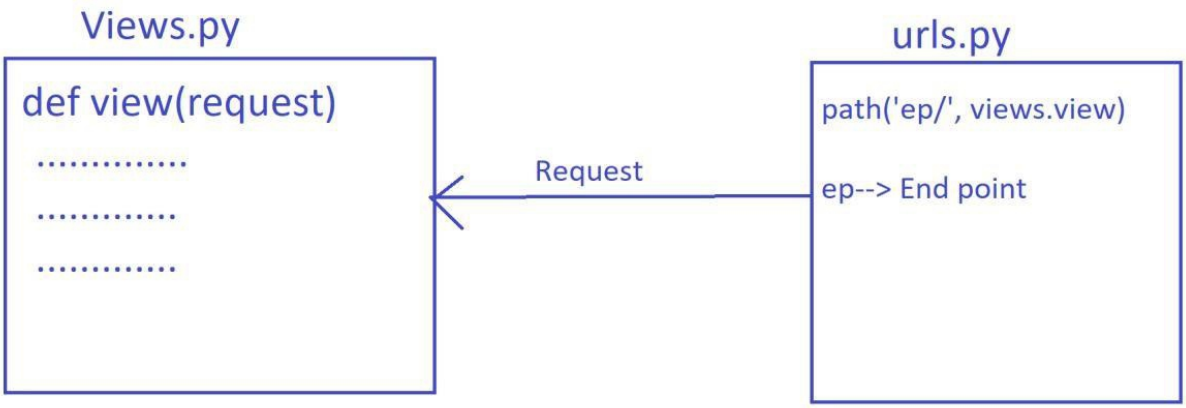
Creating views or Business Logic:

- Each view is a function that specifies business logic, used to perform a particular task.
- Each view should accept at-least a parameter called Request.
- Request specifies http request object.
- View accept request(http request) as a input and it will process the request, and response is generated.
- This Response can be directly sent to output.(html file).



Creating url pattern:

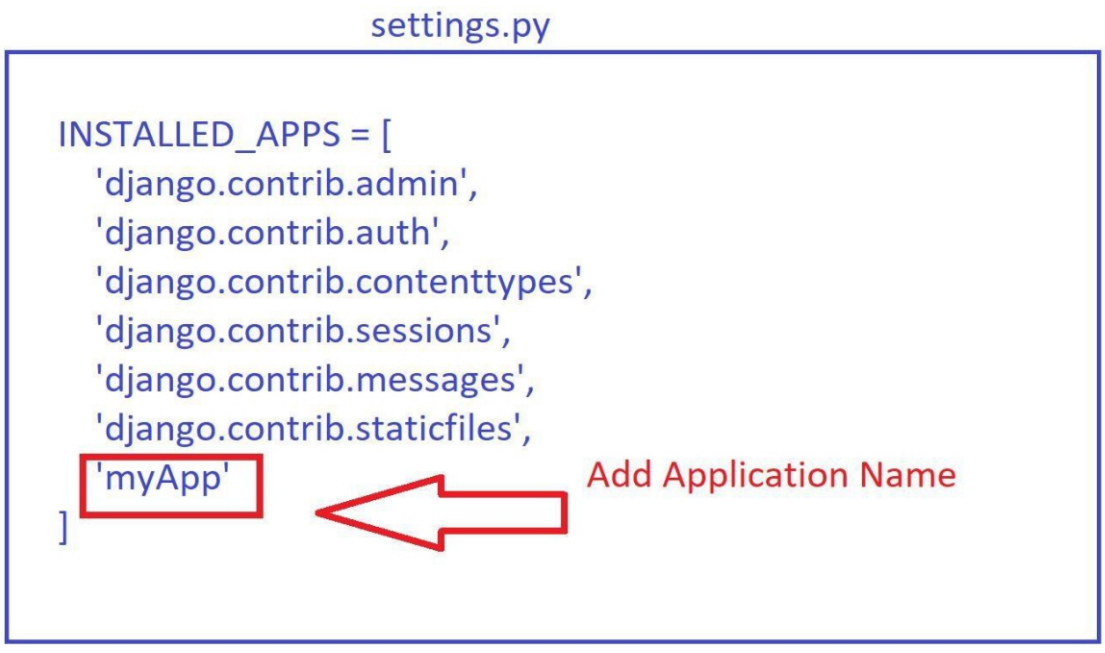
url pattern specifies End point. This end point is associated with corresponding view. Through the url request is accepted.



Manage.py: Command line utility to interact with Django project in different ways like to create an application, to run development server, to create migrations **etc**.

➔Go to sublime text, choose file

option→openfolder→fisrtProject(outer level)



Views.py

```
from django.shortcuts import render
from django.http import HttpResponse
# Create your views here.
def view1(request):
    s="Welcome to Python Webdevelopment"
    return HttpResponse(s)
```

View

urls.py

```
from django.contrib import admin
from django.urls import path
from myApp.views import view1

urlpatterns = [
    path('response1/', view1),
]
```

End Point

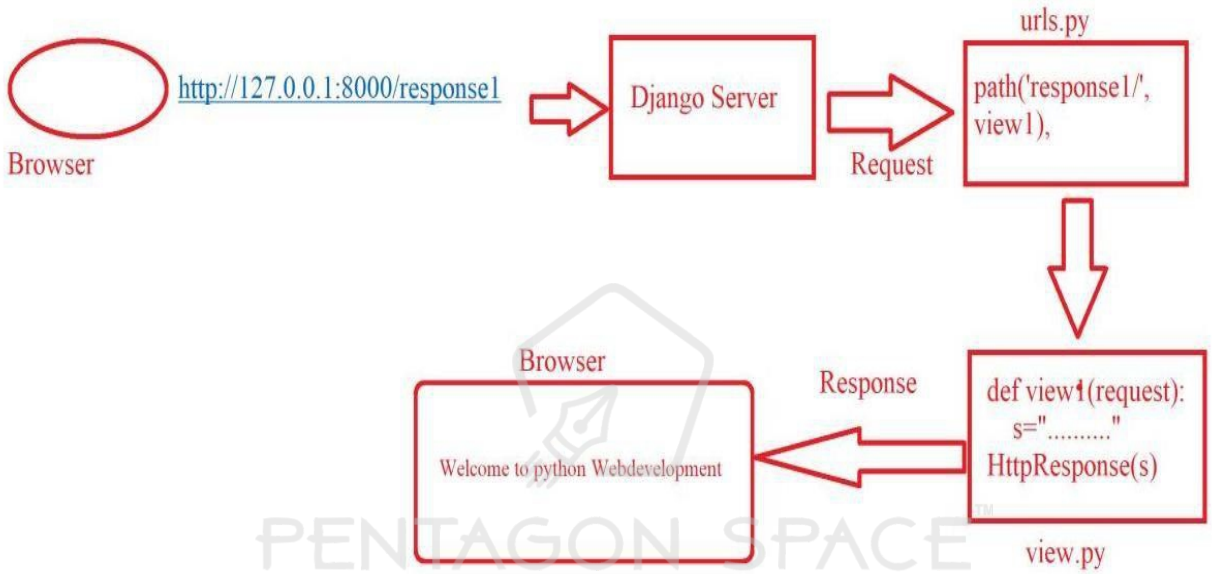
Class 3 :

Run the server by using command:

Copy the link http://127.0.0.1:8000/

Now load this link on the browser window along with the endpoint

http://127.0.0.1:8000/response1



We can write urls.py in the following way also:

```
from django.contrib import admin
from django.urls import path
from myApp import views
urlpatterns = [ path('response1/', views.view1), ]
```

Execution Steps:

Step 1: Create a folder to hold all Django projects by using mkdir

djangoProjects

Step2: change the directory to djangoProjects

Cd djangoProjects

Step3: Create a Django project

Django-admin startproject firstProject

Step 4: create an application inside firstProject

Cd firstProject py manage.py startapp myApp

step5: goto **se翻ngs.py** and add application to **INSTALLED_APPS list, edit views.py add pattern in urls.py** **step6:** Run the development server **py manage.py runserver**

step7: copy paste the http link by **appending the endpoint** used **urls.py**

Creating Multiple views inside a single Application:

1. Django-admin startrproject secondProject
2. cd secondProject
3. py manage.py startapp myApp

se翻ngs.py:

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

Views.py:

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

# Create your views here.
def view1(request):
s="<h1>This is 1st Response"
return HttpResponse(s) def
view2(request):
n1=int(input("Enter 1st num:"))
n2=int(input("Enter 2nd num:"))
n3=n1+n2 return
HttpResponse(str(n3))
```

urls.py:

```
from django.contrib import admin
from django.urls import path
from myApp.views import*
urlpatterns = [
    path('admin/', admin.site.urls),
    path('response1/',view1),
    path('response2/',view2),
    ]
```

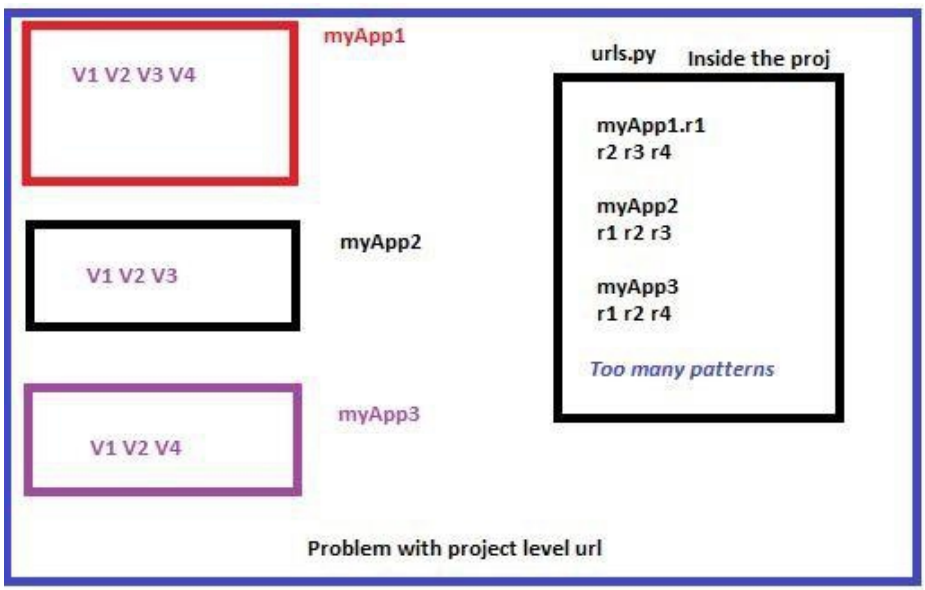
Run server: py manage.py runserver 8002

http://127.0.0.1:8002/response1/ → **This is 1st Response**

http://127.0.0.1:8002/response2/ → **500**

Creating url pattern at application level:

If we have multiple applications inside a project with multiple views,creating url patterns inside project level urls would be di□cult to maintain.To avoid this we can create at application level.



Step 1: django-admin startproject thirdProject

Step 2: cd thirdProject

Step 3: py manage.py startapp myApp

Step 4:Goto se翻ngs.py and add the application Step

4:Create a file in myApp,urls.py myApp/urls.py from

django.contrib import admin from django.urls import

path from myApp.views import * urlpatterns = [

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

path("response1/",view1), path("response2/",view2)

]

views.py

from django.shortcuts import render from

django.http import HttpResponse # Create

your views here. def view1(request):

s="<h1>This is 1st Response" return

HttpResponse(s)

def view2(request): s="<h1>This is 2nd Response" return

HttpResponse(s) Project level urls.py[main urls.py] from

django.contrib import admin from django.urls import

path,include urlpatterns = [

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

path('response/',include('myApp.urls'))

]

py manage.py runserver open browser and put request

http://127.0.0.1:8000/response/response1/

http://127.0.0.1:8000/response/response2/

Class 4 : A Project with Multiple Application

- cd djangoApplications
- django-admin startproject fourthProject
- py manage.py startapp myApp1
- py manage.py startapp myApp2 se翮⍰ngs.py:

```
INSTALLED_APPS = [
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'myApp1',
'myApp2'
]
```

myApp1/views.py:

```
from django.shortcuts import render
from django.http import HttpResponse
# Create your views here.
def view1(request):
s="Inside myApp1"
return HttpResponse(s)
```

myApp2/views.py:

```
from django.shortcuts import render from django.http import HttpResponse
# Create your views here. def view1(request):
```

s="<h1>Inside myApp2<h1>"

return HttpResponse(s)

urls.py:

from django.contrib import admin

from django.urls import path

from myApp2 import views from

myApp1 import views urlpatterns

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

path('response1/',views.view1),

path('response2/',views.view1)

]

py manage.py runserver error:

Because of name collision

Solution: Aliasing from

django.contrib import admin from

django.urls import path from

myApp1 import views as v1 from

myApp2 import views as v2

urlpatterns = [path('admin/',

admin.site.urls),

path('response1/',v1.view1), path('response2/',v2.view1)

]

http://127.0.0.1:8000/response1/ Inside myApp1

http://127.0.0.1:8000/response2/ Inside myApp2 **TEMPLATES IN DJANGO:**

Templates means an HTML File

Step 1: At project level, create a folder by name, templates

templateProject ----manage.py

----myApp.py

----templateProject

---- templates

Step 2: Inside templates, creates a folder by name myApp(with application name)

TemplateProject

----manage.py

----myApp

----templateProject

----templates myApp

Step 3: Inside myApp file we can create one more html files

C:\Users\Dell\DjangoApplications>django-admin startproject templateProject

C:\Users\Dell\DjangoApplications>cd templateProject

C:\Users\Dell\DjangoApplications\templateProject>py manage.py startapp myApp

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

```
from pathlib import Path
import os
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
TEMPLATE_DIR=os.path.join(BASE_DIR,'templates')
```

To the existing BASE_DIR path,join the path related to 'templates'

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplat
        'DIRS': [TEMPLATE_DIR],
        'APP_DIRS': True,
        'OPTIONS': {
```

views.py

```
from django.shortcuts import render
# Create your views here.
def template_view(request):
        return render(request,'myApp/1.html')
```

urls.py

```
from django.contrib import admin
from django.urls import path
from myApp.views import template_view
urlpatterns = [
        path('admin/',admin.site.urls),
        path('response/',template_view)
        ]
```

templates/myApp/1.html

```
<html>
	<center>
		<body bgcolor="yellow">
			<h1 style="color: blue;">First HTML Page of DJANGO</h1>
		</body>
	</center>
</html>
```

pymanage.pyrunserver

Class 5 :

```
from django.shortcuts import render
# Create your views here.
def template_view(request):
        return render(request,'myApp/1.html')
```

Render →is used to render the request to the given template[1.html]

render(request,template path,context)

context is a dictionary:contains key value pairs

```
from django.shortcuts import render #
Create your views here.
def template_view(request):
    name=”dhoni”
    id=101
    place=”B’lore”
    return render(request,'myApp/1.html',context={‘key1’:name,’key2’:id,’key3’:place})
    or
    return render(request,'myApp/1.html',{‘ key1’:name,’key2’:id,’key3’:place}) this
```

context would be sent to 1.html

1.html

To fetch the values, Jinja 2 syntax

{{key1}}→ name dhoni
{{key2}}→ id 101
{{key3}}→ place B’lore

---

views.py

```
from django.shortcuts import render

# Create your views here.
def template_view(request):
    name="Dhoni" id=101 place= "Blore "
    context={"key1":name,"key2":id,"key3" :place}
    return render(request,'myApp/1.html',context )
```

1. html

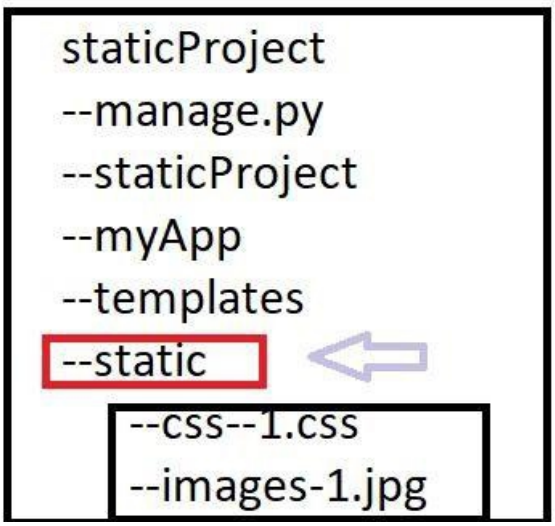
```
from django.shortcuts import render #
Create your views here.
def template_view(request):
        name=   "Dhoni   "   id=101   place="Blore"
        context={"key1":name,"key2":id,"key3"  :place}
        return render(request,'myApp/1.html',context )
```

Static Files In Django:

Note:Jinja syntax uses templating language that contains variables and programming logics.
The variables or programming logics are placed inside delimiters
Ex:To fetch any result or data we use {{data}},For expressions we use {%expression%}

{%for i in sequesnce%} Action1 Action2 {%endfor%}	{% if condition %} Action1 Action2 {%else%} Action3 Action4 {%endif%}

Step 1: At project level,create a folder static,Inside this folder create a folder by name css Inside css and images>



C:\Users\Dell\DjangoApplications>django-admin startproject staticProject

C:\Users\Dell\DjangoApplications>cd staticProject

C:\Users\Dell\DjangoApplications\staticProject>py manage.py startapp myApp

Step 2: se翮ngs.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myApp'
]
import os
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
TEMPLATE_DIR=os.path.join(BASE_DIR,'templates')
STATIC_DIR=os.path.join(BASE_DIR,'static')

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

At the end add the following line,

```
STATICFILES_DIRS=[STATIC_DIR] .....
.....
.....
```

1.html

```
{% load static %}
<head>
<link rel="stylesheet" href="{% static "css/1.css" %}"/>
<img src="{% static "images/1.jpg" %}" alt="....."/>
</head>
```

views.py

```
from django.shortcuts import render
# Create your views here.
def my_view(request):
        myName="Harish"
        favPlayer="Dhoni"
        favAnimal="Lion"
        favSub="Python"
        d={"name":myName,"player":favPlayer,"animal":favAnimal,"subject":"favSub"} return
        render(request,"myApp/1.html",d)
```

urls.py

```
from django.contrib import admin
from django.urls import path from
myApp.views import my_view
urlpatterns = [ path('admin/',
admin.site.urls),
        path('response/',my_view)
        ]
```

templates/myApp/1.html

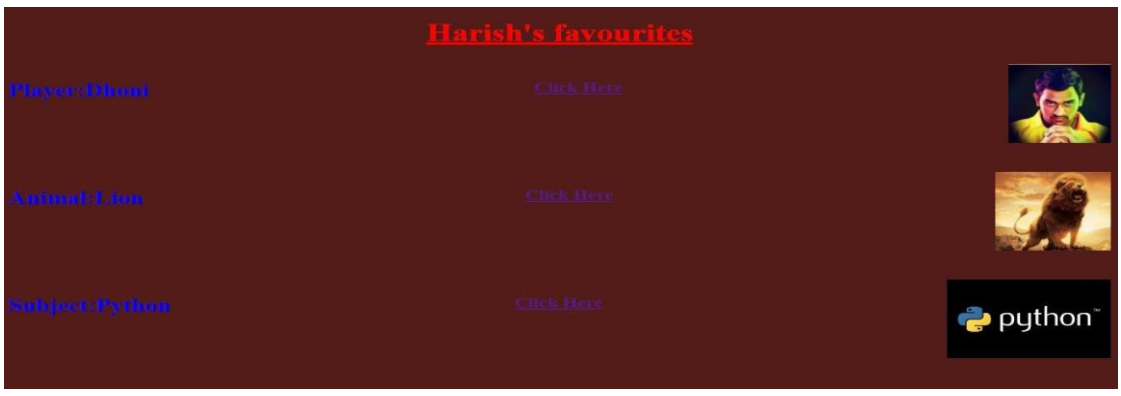
```
<!DOCTYPE html>
{%load static%}
<html lang="en" dir="ltr">
  <head> <link rel="stylesheet"href="{% static "css/1.css" %}">
  </head>
  <body bgcolor=#541c18>
    <center>
      <h1><u>{{name}}'s favourites</u></h1>
    </center>
    <div>
      <h2>Player:{{player}}</h2>
      <h3><a href="https://en.wikipedia.org/wiki/MS_Dhoni">Click Here</a></h3>
      <img src="{% static "images/dhoni.jpg" %}" alt="not available">
    </div>
```

```
    <br><br>
    <div>
      <h2>Animal:{{animal}}</h2>
      <h3><a href="https://en.wikipedia.org/wiki/Lion">Click Here</a></h3>
      <img src="{% static "images/lion.jpeg" %}" alt="not available">
    </div>
    <br><br>
    <div>
      <h2>Subject:{{subject}}</h2>
      <h3><a href="https://www.python.org/">Click Here</a></h3>
      <img src="{% static "images/python.jpg" %}" alt="not available">
    </div>
    <br><br>
  </body>
</html>
```

static/css/1.css

```
h1
{
color: red;
}
h2
{
color: blue;
}
img{
height:100px;
width:200px;
border:2px solid black;
}
h3
{
color:green;
} div{
display:flex;
justify-content: space-between;
}
```

py manage.py runserver



Class 6 : Models in Django

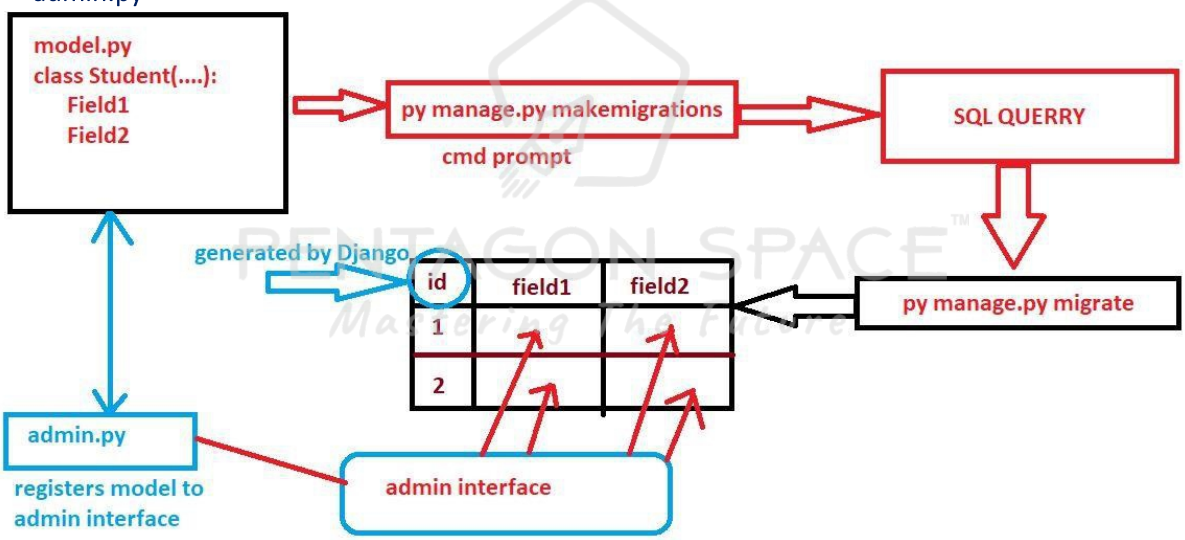
In Relational database we store the information in the form of a table, in which each row is referred to as record.

Each heading is called a field.

Name	Id(Primary Key)	Place
Rohan	1	Bangalore
Radha	2	Pune
Rosh an	3	Mumbai

Each record is identified with a unique number called primary key

model.py
admin.py



C:\Users\KITS>cd DjangoApplications

C:\Users\KITS\DjangoApplications>Django-admin startproject modelProject1

C:\Users\KITS\DjangoApplications>cd modelProject1

C:\Users\KITS\DjangoApplications\modelProject1>py manage.py startapp myApp

Se翻ngs.py

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

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
TEMPLATE_DIR=os.path.join(BASE_DIR,'templates')

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

Create templates/myApp/

In order to create a model class we must inherit the behaviors of Django.db.models.Model

Integer data→IntegerField(),

Float data→FloatField(),

Char data→CharField(max_lenght=30)

Max length→ max no of chars

models.py

```
from django.db import models

# Create your models here.
class Student(models.Model):
    number=models.IntegerField()
    name=models.CharField(max_length=40)
    marks=models.FloatField()
```

admin.py

This file is used to register our model to admin interface, here we specify the fields that created in models inside a class.

This class contains the fields, along with the class that we have created inside models.py should be registered.

Step1 : Import the required class from the models.py

Step2 : create an admin class which is the child of admin.ModelAdmin, Inside this class we should create a list containing all the fields of model class.

Step3 : register admin class and corresponding model class by using the function

admin.site.register(modelclassname,adminclassname)

admin.py

```
from django.contrib import admin
from myApp.models import Student
# Register your models here.
class StudentAdmin(admin.ModelAdmin):
l=['number','name','marks']
admin.site.register(Student,StudentAdmin)
```

py manage.py makemigrations

This step creates the following file

myApp\migrations\0001_initial.py

```
....
.... .... operations = [
migrations.CreateModel(
name='Student', fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
                serialize=False, verbose_name='ID')),#auto generated field
                ('number', models.IntegerField()),
                ('name', models.CharField(max_length=40)),
                ('marks', models.FloatField()),
            ],
        ),
    ]
```

py manage.py migrate

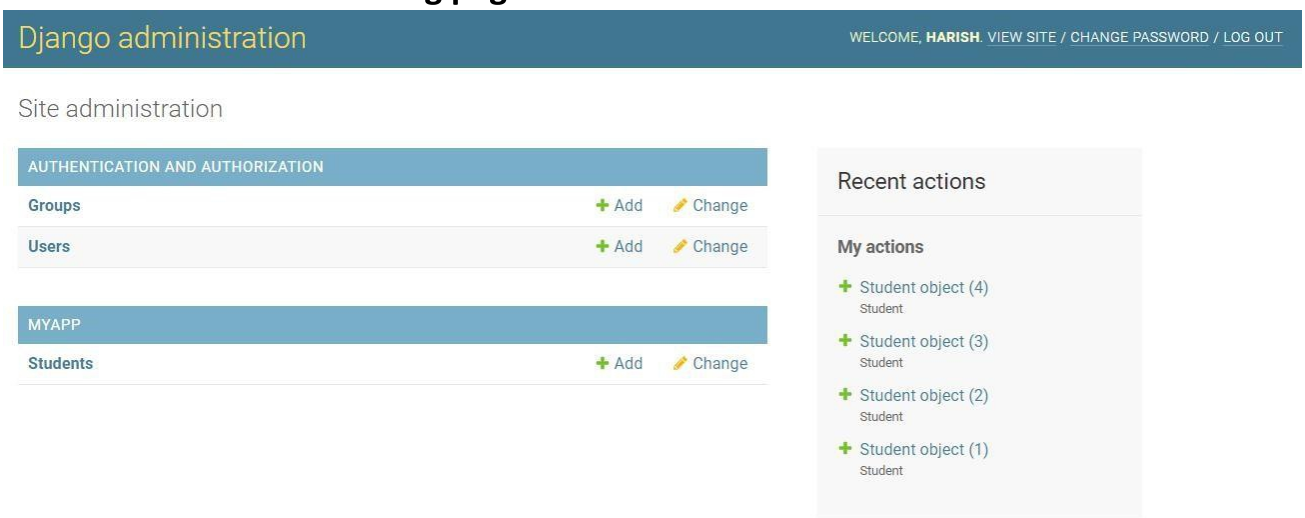
In order to have access to admin interface we must create superuser.by using the following cmd py manage.py createsuperuser

Username (leave blank to use 'dell'): shreenath
Email address: shreenath@gmail.com
Password:
Password (again):
Superuser created successfully.

Run server: py manage.py runserver

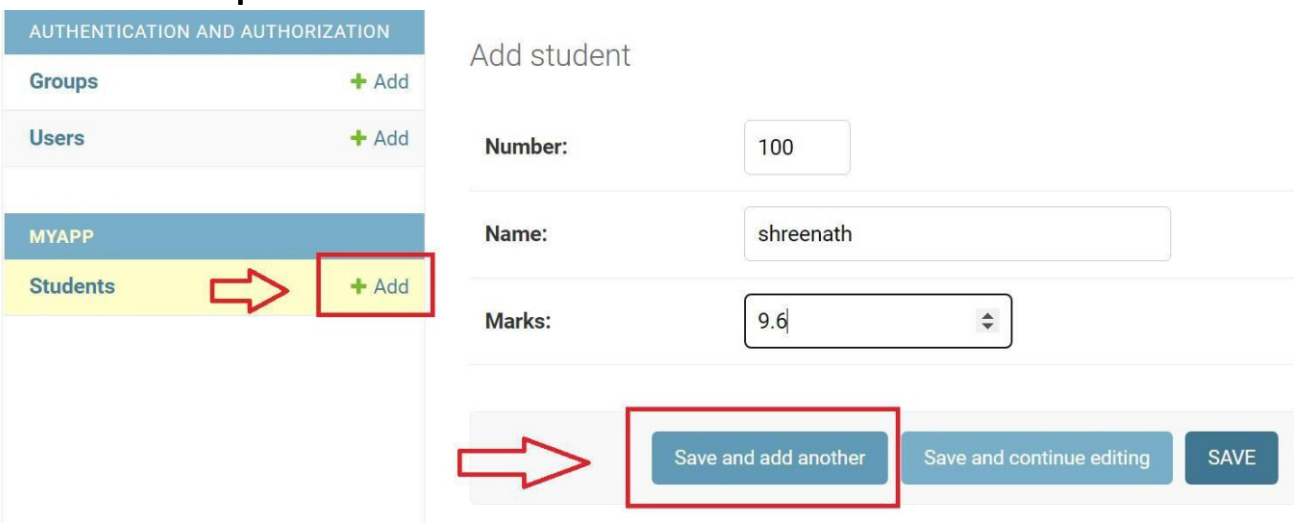
Send the request: http://127.0.0.1:8000/admin/

This will load the following page

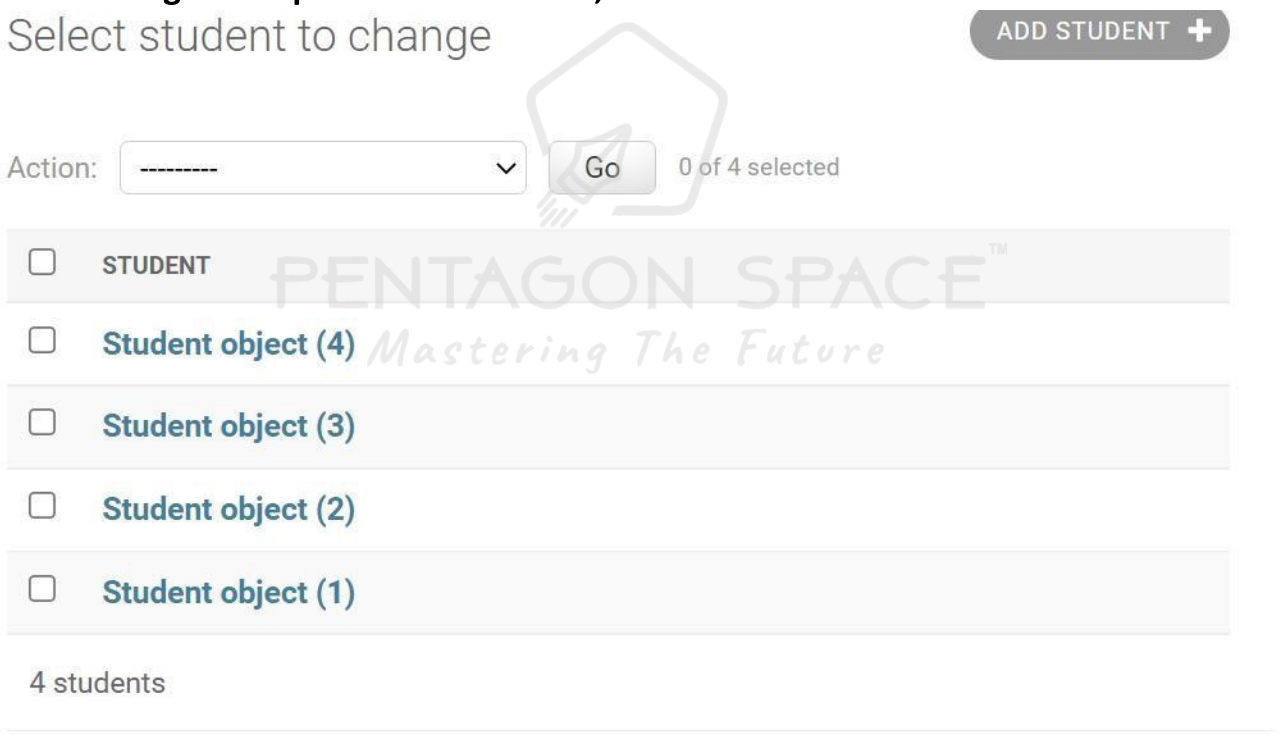


Students is the table name, if we click in students we get the following interface.

Click on Add option and enter the details



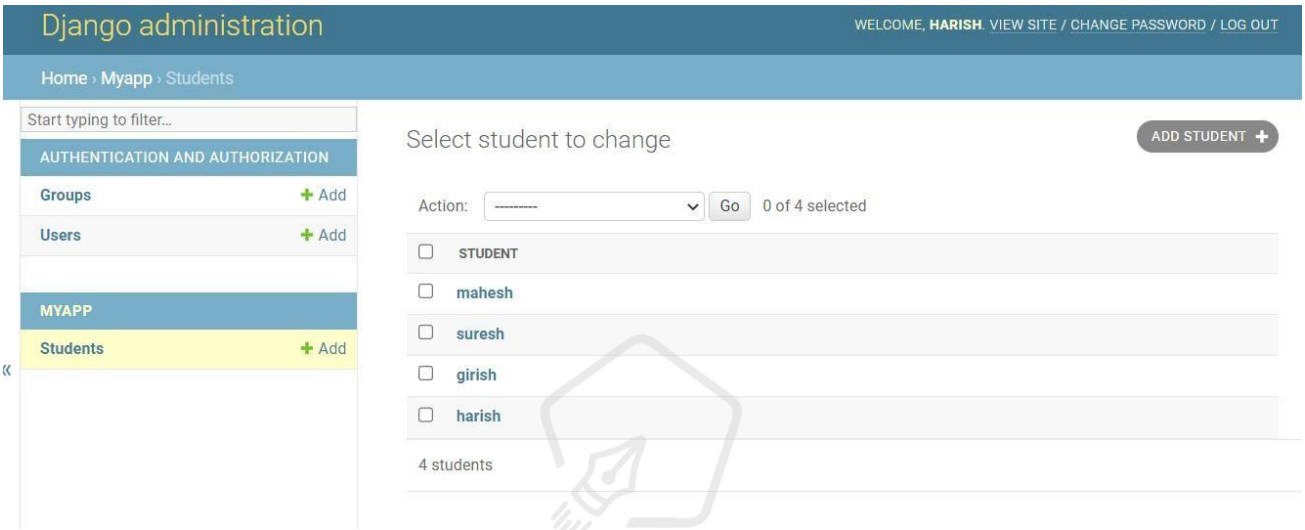
After adding the required no of records,interface looks like this



In order to convert object representation into name representation we use the method __str__ as shown below

models.py

```
from django.db import models class
Student(models.Model): #Model is a class
number=models.IntegerField()
name=models.CharField(max_length=40)
marks=models.FloatField() def __str__(self):
return self.name
```



Class 7 :ModelProjects

The data,that has been stored in admin interface should be displayed to the end user.

- **The data type associated with records is QuerySet.**
- **S=Classname.objects.all() returns all the records that have been created in admin interface.**
- **Here S is an object that refers to set of records returned from database.**
- **type(S)→QuerySet**
- **In order to pass the data from views to template,the data must be in the form of a context[dictionary]**
- **Hence we must convert the QuerySet→dictionary By using the following.**

statement d={'stud':s} stud→key s→value refers to multiple records

Now we can access the data by referring to key i.e stud inside html file

admin.py

```
from django.shortcuts import render
from myApp.models import Student
# Create your viewshere.
def myView(request):
    s=Student.objects.all()
    print(type(s))
    d={'stud':s} #qs to dict
    return render(request,'myApp/1.html',d)
```

urls.py

```
from django.contrib import admin
from django.urls import path from
myApp.views import myView
urlpatterns = [ path('admin/',
admin.site.urls),
    path('response/',myView)]
```

templates/myApp/1.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title></title>
    </head>
    <body>
        <h1>Student information</h1>
        <table border="2">
            <tr>
                <td>Number</td>
                <td>Name</td>
                <td>Place</td>
            </tr>
            {%for i in stud%}
            <tr>
                <td>{{i.number}}</td>
                <td>{{i.name}}</td>
                <td>{{i.marks}}</td>
            </tr>
            {%endfor%}
        </table>
    </body>
</html>
```

py manage.py runserver

send request http://127.0.0.1:8000/response/

1.django-admin startproject modelProject1
2.cd modelProject1
3.py manage.py startapp myApp
4.Goto settings.py add application and templates
5.Edit models.py and admin.py
6.py manage.py makemigrations
7.py manage.py migrate
8.py manage.py createsuperuser
9.py manage.py runserver
10.send request http://127.0.0.1:8000/admin/
11.Add the data in admin interface
12.Edit urls.py views.py,1.html
13.py manage.py runserver
14.send request http://127.0.0.1:8000/response/

Step 1 : django-admin startproject modelProject2

Step 2 : cd modelProject2

Step 3 : py manage.py startapp myApp

Step 4 : Create templates folder at project level inside that create myApp

Inside myApp create html files

Step 5 : Add application template configuration in se翮ngs.py

models.py

```
from django.db import models class
Employee(models.Model):
eid=models.IntegerField()
ename=models.CharField(max_length=30)
edesig=models.CharField(max_length=40)
edob=models.DateField()
eexp=models.FloatField()
esal=models.IntegerField() def __str__(self):
        return self.eid
```

admin.py

```
from django.contrib import admin
from myApp.models import Employee
# Register your models here.
class EmployeeAdmin(admin.ModelAdmin):
    l=['eid','ename','edesig','edob','eexp','esal']
admin.site.register(Employee,EmployeeAdmin)
```

py manage.py makemigrations py manage.py

migrate py manage.py createsuperuser py

manage.py runserver send req

http://127.0.0.1:8000/admin/ add data and

edit the following files

views.py

```
from django.shortcuts import render
from myApp.models import Employee
# Create your views here.
def EmployeeView(request):
    e=Employee.objects.all()
    d={'emp':e}
    return render(request,'myApp/1.html',d)
```

urls.py

```
from django.contrib import admin from
django.urls import path from
myApp.views import EmployeeView
urlpatterns = [
    path('admin/', admin.site.urls),
    path('response/',EmployeeView)
    ]
```

templates/myApp/1.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title></title>
    </head>
    <body>
        <h1>Employee info</h1>
        <table border="2">
            <tr>
                <td>Id</td>
                <td>Name</td>
```

```
            <td>Designation</td>
            <td>DOB</td>
            <td>EXPERIENCE</td>
            <td>SALARY</td>
        </tr>
        {%for i in emp%}
        <tr>
            <td> {{i.eid}}</td>
            <td> {{i.ename}}</td>
            <td> {{i.edesig}}</td>
            <td> {{i.edob}}</td>
            <td> {{i.eexp}}</td>
            <td> {{i.esal}}</td>
        </tr>
        {%endfor%}
    </table>
</body>
</html>
```

py manage.py runserver

send request http://127.0.0.1:8000/response/