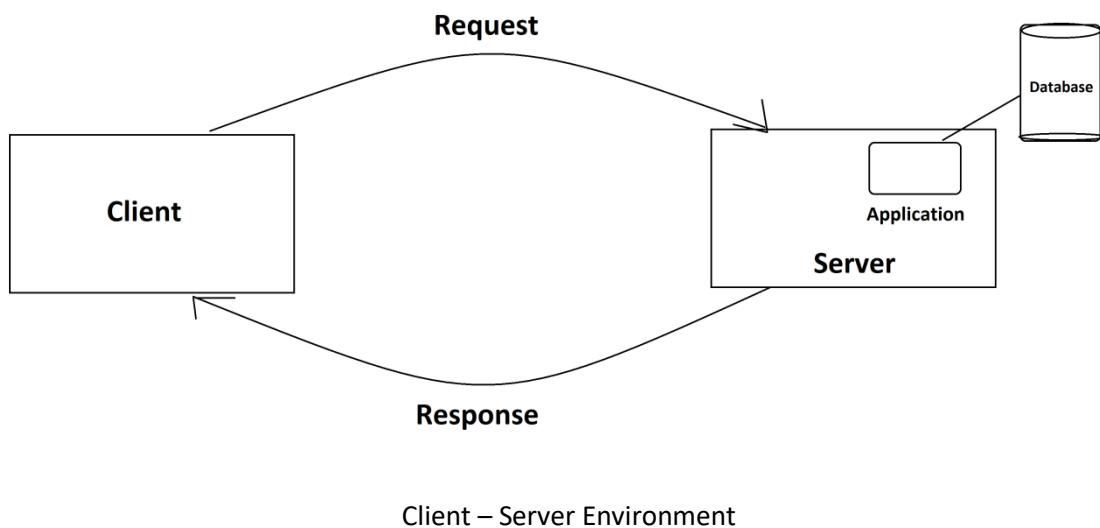


## PYTHON WEB DEVELOPMENT USING DJANGO

### INTRODUCTION

#### **Introduction to Web Applications (Client – Server Environment)**

Any application (Software) which can be accessed through web browsers (Chrome, Safari, Firefox, Opera, Edge etc.,) is said to be a web application. In this case Application will be stored at only one place i.e., in Server. All the client systems can access this application through web browsers. So, Client systems need to have only a web browser and internet connection (In case the server exists in remote place).



As shown in above figure, all the requests from clients will reach the server. The Web Application which is available in the server will receives the requests, processes it and prepares the response and sent back to the corresponding client.

We can make any system as server by installing any available web server software like apache, tomcat, WebLogic etc.,

#### **Web Designing vs Web Development:**

A web designer will make our webpages look good. They mainly focus on the look and feel of the websites. For this they may use software like Photoshop to customize the website's visual elements. They also use code like HTML and CSS (Cascading Style Sheets) to create their designs. Web designers are specialized as User Experience (UX) designer (to keep visitors hooked with our website), User Interface (UI) designer (To improve the user interactivity), Visual designer (combination of both UX & UI).

---

Web developer will build the web designer's concept. Web developers are specialized as Back-end developer (responsible for developing core structure using Java, Python, PHP, SQL etc., that what users can't see), Front-end developer (responsible for developing webpages using HTML, CSS, JavaScript that what users can see), Full-stack developer (Both Front and Back End).

### **What is Django?**

Django is a high-level python web framework that helps us in building and maintaining web applications. It helps us in developing applications very fast which in turns helpful in improving the productivity.

Django supports

- Object-Relational Mapping – ORM (provides a bridge between the data model and the database engine)
- Multilingual support (develop applications with various languages support), Framework support (Ajax etc.,)
- Administration GUI (ready to use user interface for administrative activities)
- Development Framework (supports end-to-end application development with an light weight web server)

## SOFTWARE INSTALLATIONS

### Software Installations

#### Python

Download and install latest version of python from <https://www.python.org> website.

To check version python version command is :      `python --version`

#### PIP

Once you install latest version of python (after version 3.x), pip will automatically be installed. To install pip manually visit <https://pip.pypa.io/en/stable/installing/> and follow the steps.

#### Django

To Install Django:                  `pip install django`

To Check Django Version:        `python -m django --version`

#### Install Atom IDE

To Install it visit <https://atom.io/> and download the IDE.

After installed Atom IDE do the following settings.

1) Goto File ->Settings (MAC O/S : preferences)

click on install -> search “autocomplete-python” and install it. Search for “atom-django” and install it.

Click on editor -> update the tab length to any highest number. Let us say 5.

#### Install MySql Software

Search in google as “mysql server download” or visit <https://dev.mysql.com/downloads/installer/> and install mysql server from it.

If mysql-workbench not came with above download search in google and install it also

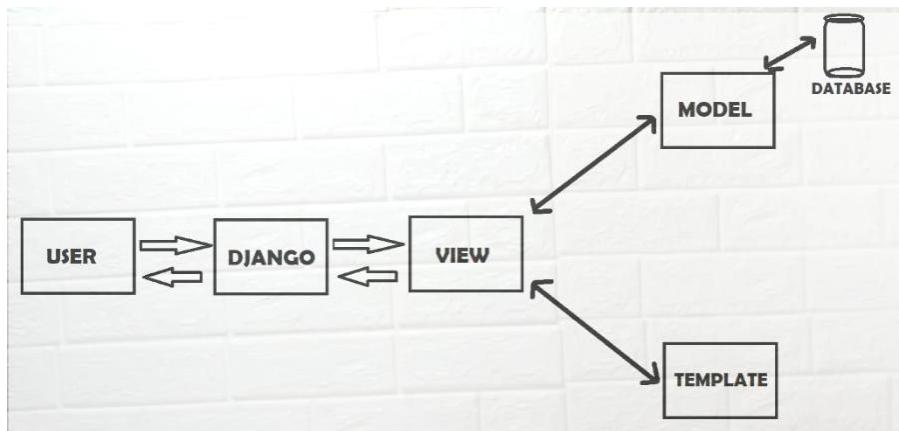
#### Install mysql client for our django applications :

Download mysqlclient from the below link

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#mysqlclient>

## MVT (MODEL – VIEW – TEMPLATE) ARCHITECTURE

### What is MVT (Model-View-Template)?



Django MVT (Model View Template) is a software design pattern used by Django framework in the process of developing web applications. It is collection of three important components Model, View and Template.

#### Model:

It helps to handle the database. We generally create a model for every table in the database. Whenever we want to access a table we will connect to the corresponding model which in turn prepares the queries and access the database and perform the requested action.

#### Template:

It is a presentation layer which handles user interface part. Generally, we create multiple templates where these are used as responses to the user actions.

#### View:

It is used to execute the business logic. It interacts with both model and template. We develop multiple views where each view is responsible for a specific action. While performing the action it will connect to model for database accessing related operations and after performing database related actions it will show a template as response to the client.

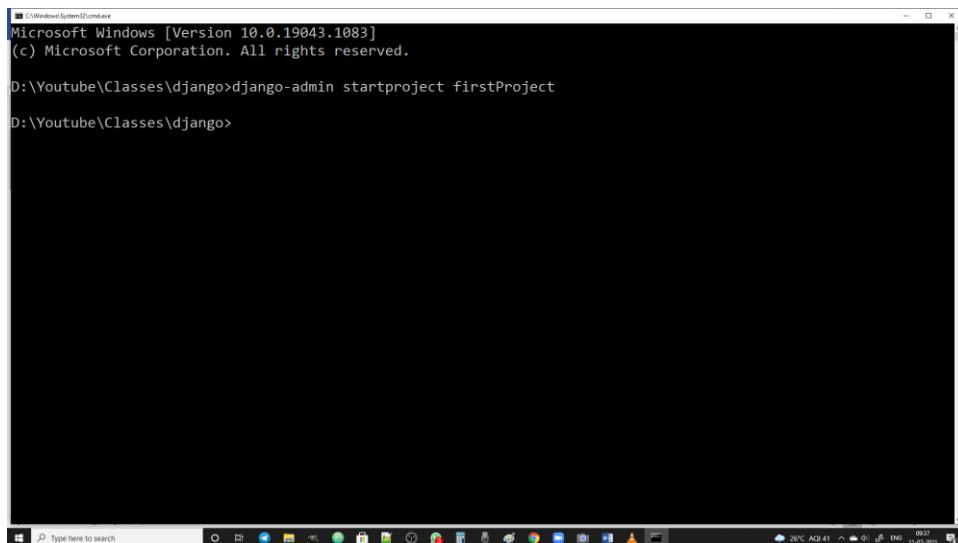
## CREATING OUR FIRST DJANGO PROJECT

Django allows us to add multiple applications under same project. With the help of this feature we can easily use one application's modules in other applications.

### Creating First Project:

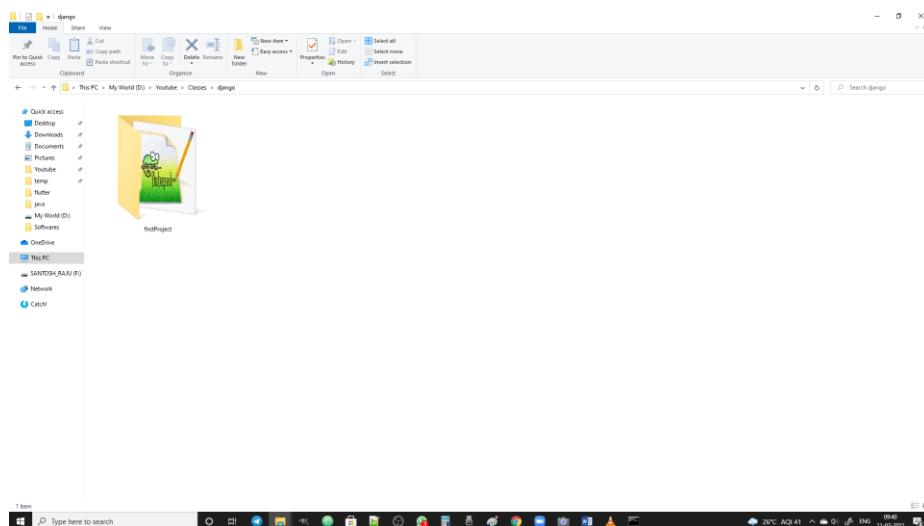
To create our first project, we need to run the below command

```
django-admin startproject projectname
```

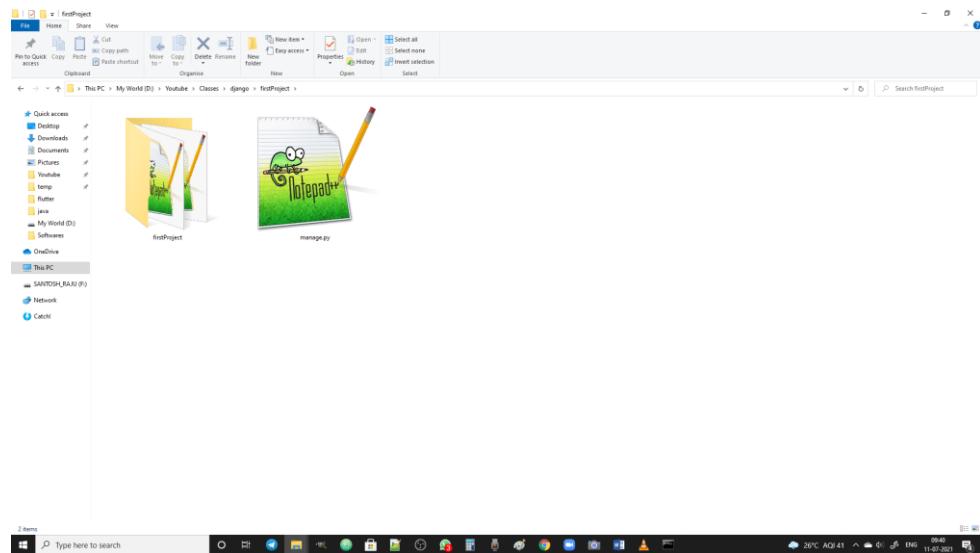


Microsoft Windows [Version 10.0.19043.1083]  
(c) Microsoft Corporation. All rights reserved.  
D:\Youtube\Classes\django>django-admin startproject firstProject  
D:\Youtube\Classes\django>

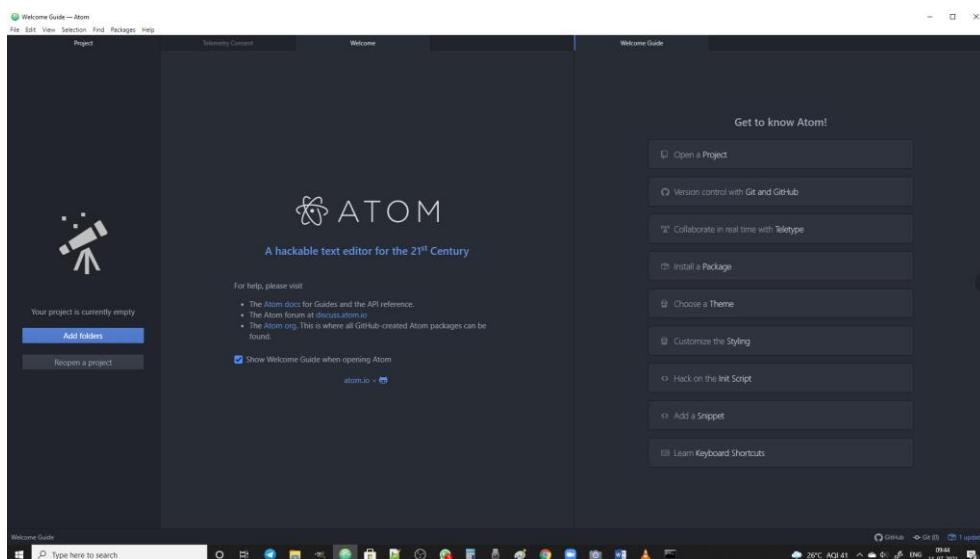
After running this command, it will creates a project folder with required files automatically.



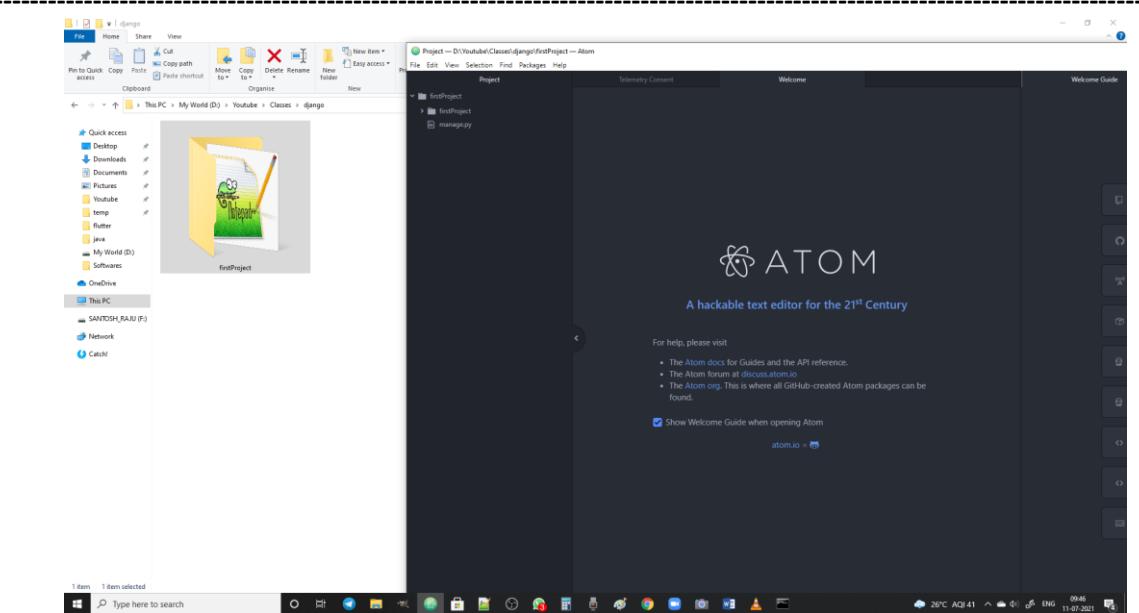
Inside this project folder, it creates a folder and a file(`manage.py`) as shown below.



Now let us open our IDE and load the recently created project into our IDE.

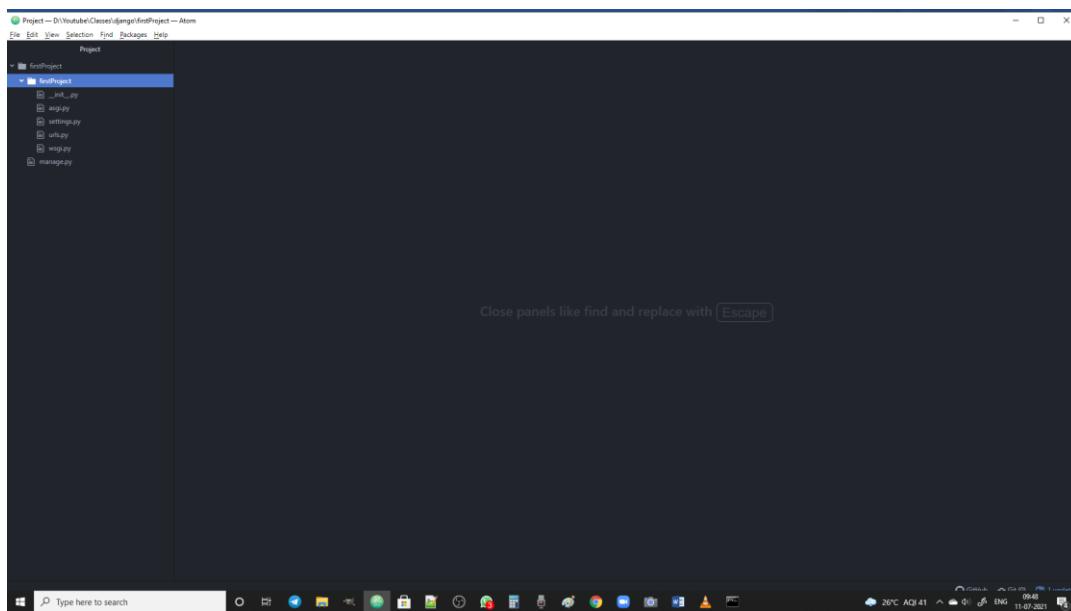


To load the project into atom, we can simply drag and drop the project folder into our ATOM.



Our project is successfully loaded into our ATOM.

Now let us observe the folder structure of our project



**manage.py** : We take the help of manage.py while we run our project every time

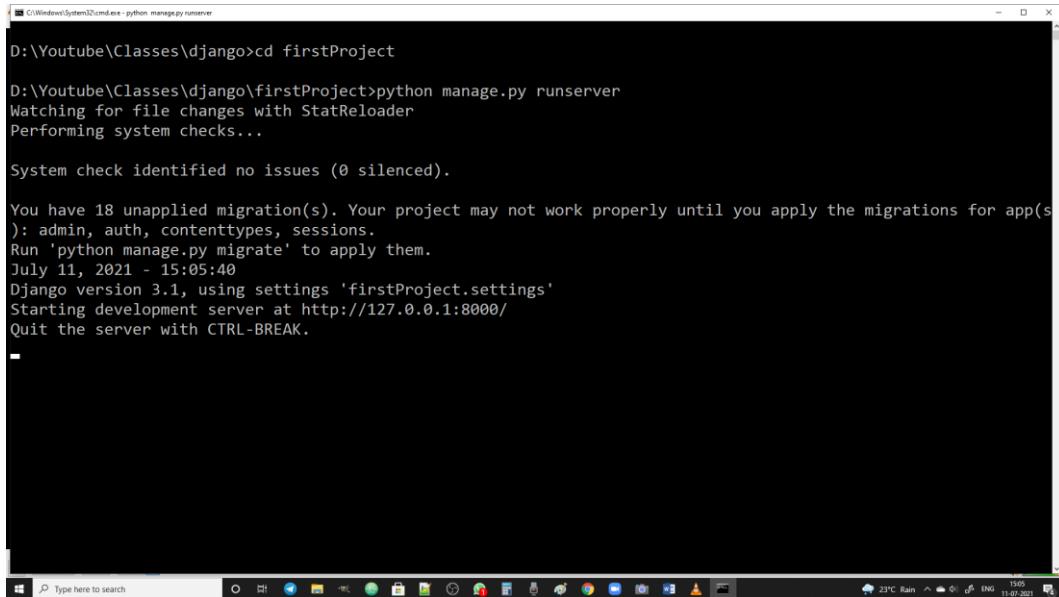
**init.py** : Indicates this is the python based project

**settings.py** : complete project settings can be configured here

**urls.py** : views and url mappings can be performed in this file.

To Run our project we need to execute the below command

```
python manage.py runserver
```



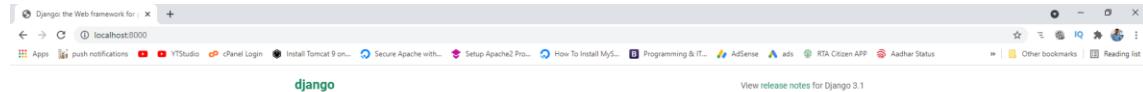
```
C:\Windows\System32\cmd.exe - python manage.py runserver
D:\Youtube\Classes\django>cd firstProject
D:\Youtube\Classes\django\firstProject>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s):
admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 11, 2021 - 15:05:40
Django version 3.1, using settings 'firstProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Now we can run our project in web browser by using the url

<http://127.0.0.1:8000> or <http://localhost:8000>



It is default output page provided by django. whenever we add applications then we may see our configured screens as output.

We can stop our server by using **ctrl + c**

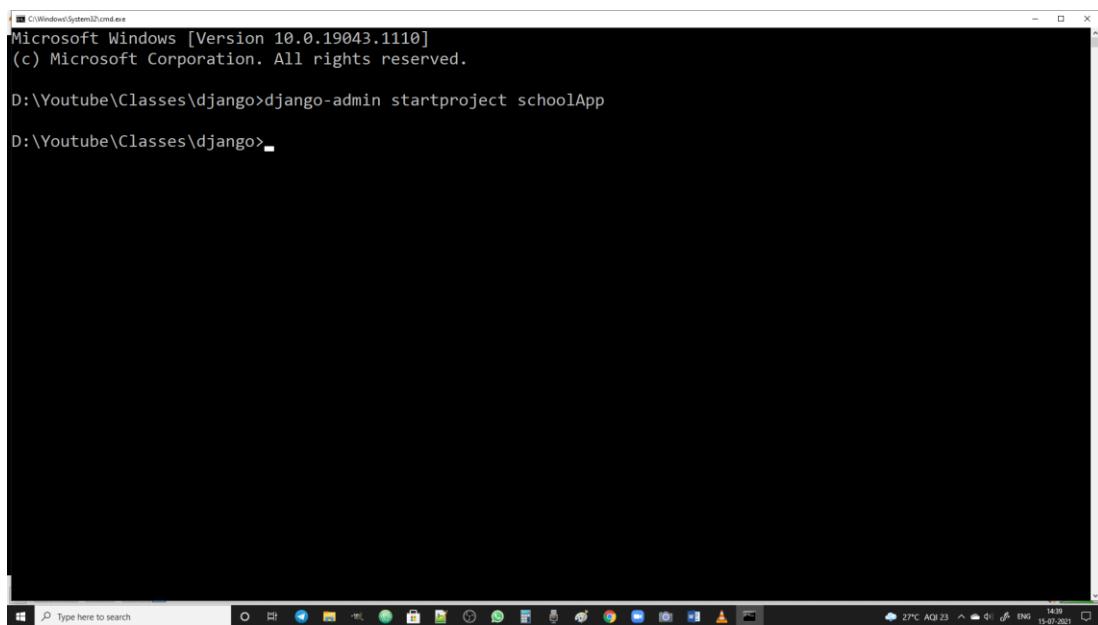
## CREATING APPLICATIONS &amp; VIEWS IN OUR DJANGO PROJECT

Now Let's learn how to add applications to our django project.

**Steps to add application to the existing django project:**

- 1) Create an application
- 2) Add this application's entry into settings.py
- 3) create views under the project
- 4) link views with urls in urls.py file
- 5) Run the project

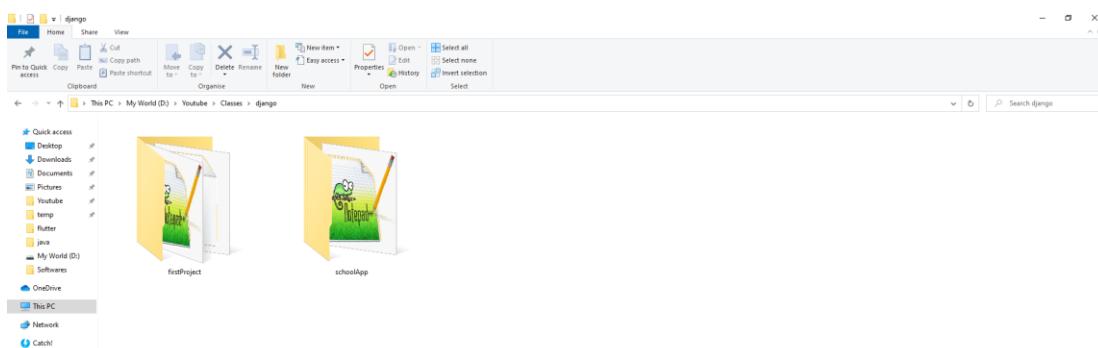
First let's create a fresh project called schoolApp



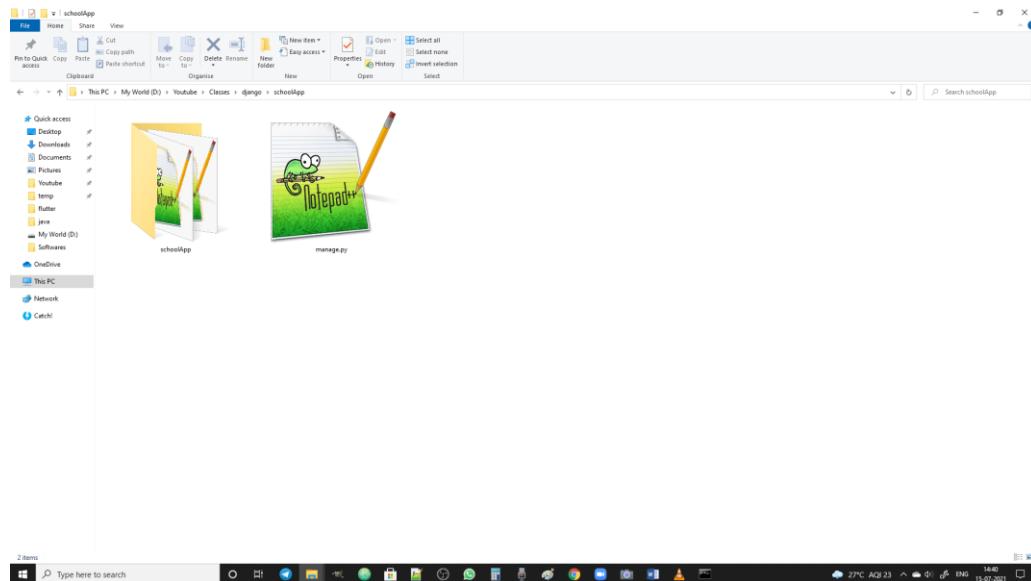
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1110]
(c) Microsoft Corporation. All rights reserved.

D:\Youtube\Classes\django>django-admin startproject schoolApp

D:\Youtube\Classes\django>
```



Now schoolApp project is created and its folder is as shown below



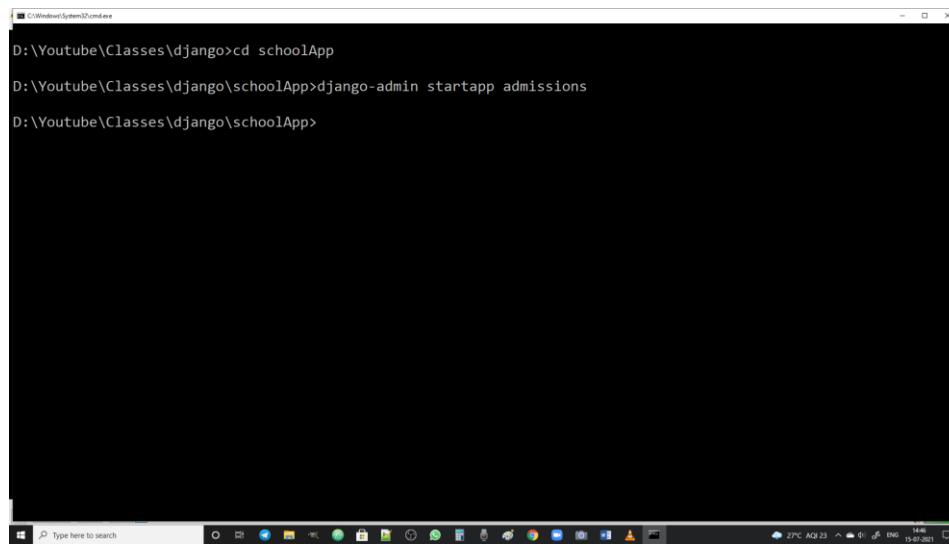
Let's load this project into atom IDE

Now let us create 'admissions' application into our schoolApp project

### Step-1: Creating an Application

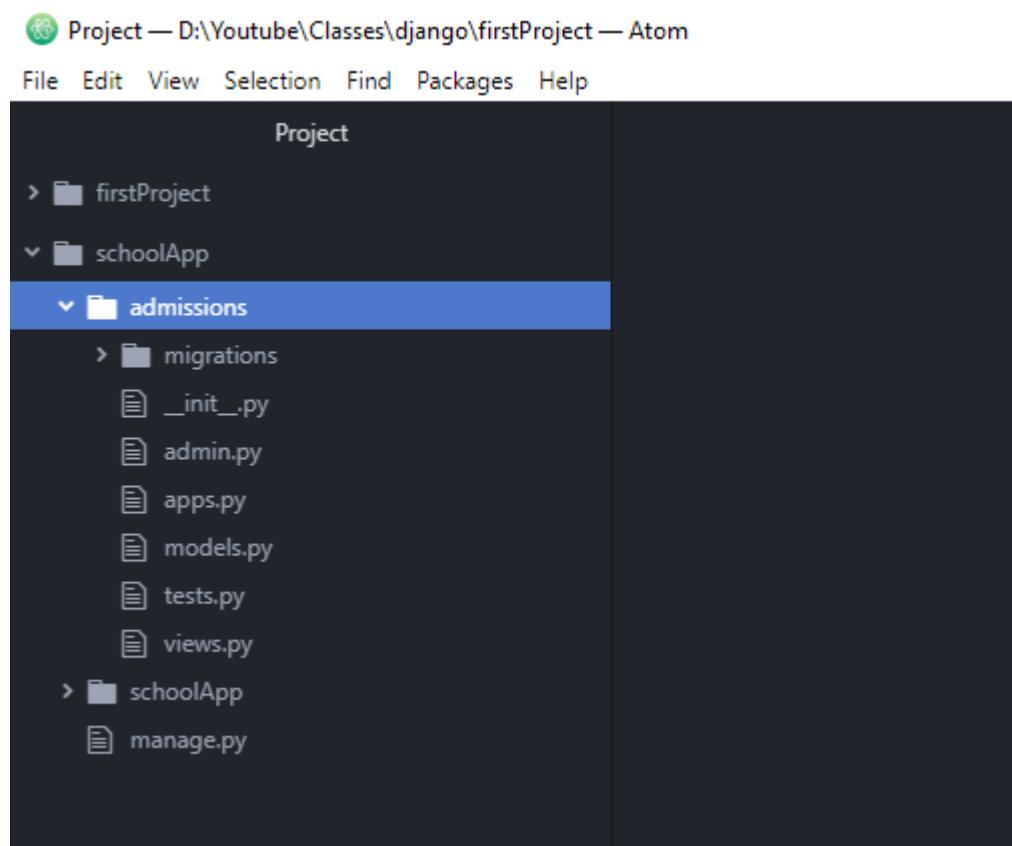
We can create an application by using startapp command as follows

```
django-admin startapp admissions
```



```
D:\Youtube\Classes\django>cd schoolApp
D:\Youtube\Classes\django\schoolApp>django-admin startapp admissions
D:\Youtube\Classes\django\schoolApp>
```

Now observe the admissions application's inner folder structure that is shown below



In this structure we create views in views.py file.

views – urls mappings can be done in urls.py file.

admin.py is related with administration related activities which will be discussed in further topics.

### Step-2: Add this application's entry into settings.py

After creating the application we need to add it in settings.py so that it is available to the entire project so that one application's elements can be accessed by other applications.

To add this we need to open settings.py file

and in this file, under INSTALLED\_APPS we need to add our application as follows.

```
settings.py — D:\YouTube\Classes\django\schoolApp — Atom
File Edit View Selector Find Packages Help
Project
viewsp.py urls.py settings.py
> fmProject
> schoolApp
> admissions
> schoolApp
    <__init__.py
    __init__.py
    settings.py
    urls.py
    wsgi.py
    manage.py
23 SECRET_KEY = 'p#a!2(o9e2u*a(89+$9i#0t1*78a+b5-7wr+-rpf83s&kn5f5'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'admissions',
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
99
```

### **Step-3:Create views under the project**

We can create the views by using functions (Function based views) or classes (Class Based Views) in views.py file. First let us discuss function-based views and then in further classes we will discuss about class-based views.

A function based view can be written as follows:

```
def viewname():
```

[view code](#)

The screenshot shows the Atom IDE interface with the following details:

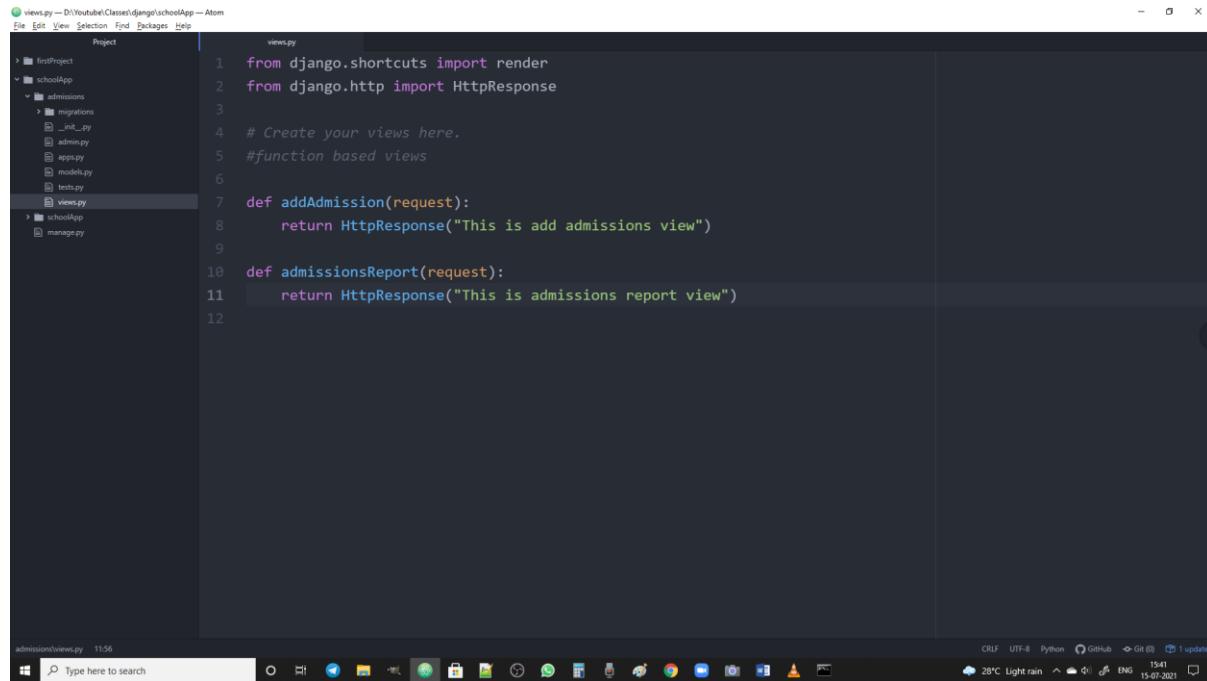
- File Path:** views.py — D:\Youtube\Classes\django\schoolApp — Atom
- Project Structure:** The left sidebar shows a project named "firstProject" containing a "schoolApp" folder. Inside "schoolApp", there are "admissions", "migrations", and "views" subfolders, each with its own Python files (e.g., migrations.py, admissions.py, views.py).
- Code Editor:** The main editor area displays the contents of "views.py". The code includes imports for django.shortcuts.render, comments for creating function-based views, and two defined functions: addAdmission(request) and admissionsReport(request), both of which contain placeholder code.
- Bottom Status Bar:** Shows file statistics (admissions/views.py, 829), system information (CPU: 100%, RAM: 1537), and a search bar.
- System Tray:** Shows weather (28°C, Light rain), battery status, and system icons.

```
views.py
1 from django.shortcuts import render
2
3 # Create your views here.
4 #function based views
5
6 def addAdmission(request):
7     code
8
9 def admissionsReport(request):
10    code
11
```

Let's fill the code later. Here request corresponds to the request which is coming from clients.

Servers generally calls the corresponding views based on request from clients so that the particular view will be executed. All the parameters that are forwarded with request are available with this request argument.

To send response back to the client, Django uses `HttpResponse`.



```

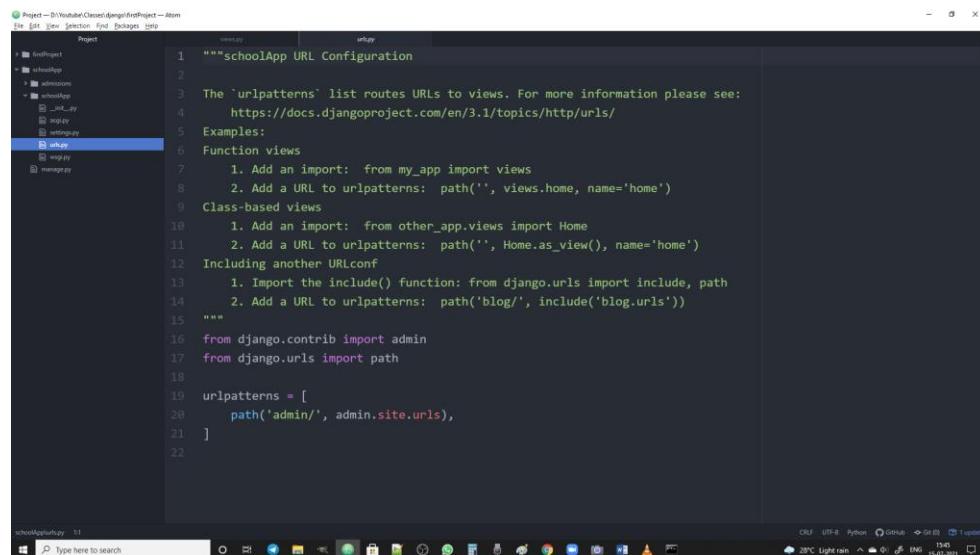
views.py
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 # Create your views here.
5 #function based views
6
7 def addAdmission(request):
8     return HttpResponse("This is add admissions view")
9
10 def admissionsReport(request):
11     return HttpResponse("This is admissions report view")
12

```

Now whenever client calls `addAdmission`, then “This is add admissions view” will be sent as response. In the same way whenever client calls `admissionsReport` then “This is admissions report view” will be displayed as response.

Client send the request to the server via urls. so we need to map each url with a view. These mappings can be done in `urls.py` file.

#### Step-4: link views with urls in `urls.py` file



```

urls.py
1 """schoolApp URL Configuration
2
3 The `urlpatterns` list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/3.1/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import: from my_app import views
8     2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21 ]
22

```

here under `url-patterns`, we will write the mappings between views and urls as follows

```

urls.py -- D:\Youtube\Classes\django\schoolApp - Atom
File Edit View Selection Find Packages Help
Project
firstProject
schoolApp
admissions
schoolApp
__init__.py
admissions.py
settings.py
urls.py
views.py
manage.py
views.py      urls.py
1 """schoolApp URL Configuration
2
3 The `urlpatterns` list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/3.1/topics/http/urls/
5 Examples:
6     1. Add an import: from my_app import views
7         2. Add a URL to urlpatterns: path('', views.home, name='home')
8 Class-based views
9     1. Add an import: from other_app.views import Home
10        2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
11 Including another URLconf
12     1. Import the include() function: from django.urls import include, path
13         2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
14 """
15 """
16 from django.contrib import admin
17 from django.urls import path
18 from admissions import views
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('newadm/',views.addAdmission),
23     path('admreport/',views.admissionsReport),
24 ]
25

```

Windows Taskbar at the bottom: Type here to search, Start button, Taskbar icons, Date/Time: 28°C AQI 23, 15:47, 15-07-2021.

Observe here that

url newadm/ is mapped with addAdmissions view &

url admreport/ is mapped with admissionsReport

### Step-5: Run the project

Now we can run the server and execute the project

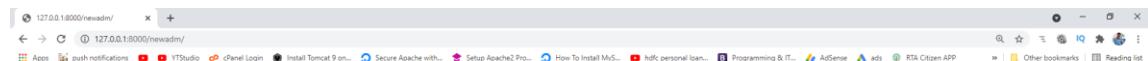
```

C:\Windows\System32\cmd.exe : python manage.py runserver
D:\Youtube\Classes\django\schoolApp>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s):
admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 15, 2021 - 16:15:45
Django version 3.1, using settings 'schoolApp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

```

Windows Taskbar at the bottom: Type here to search, Start button, Taskbar icons, Date/Time: 28°C AQI 21, 15:49, 15-07-2021.



This is add admissions view



We can run our application as

**http://127.0.0.1:8000/our\_url**

or

**http://localhost:8000/our\_url**

please observe when we request /newadm then it checks in the urls.py with which view this particular view is mapped. then addAdmission view will be executed.



This is admissions report view

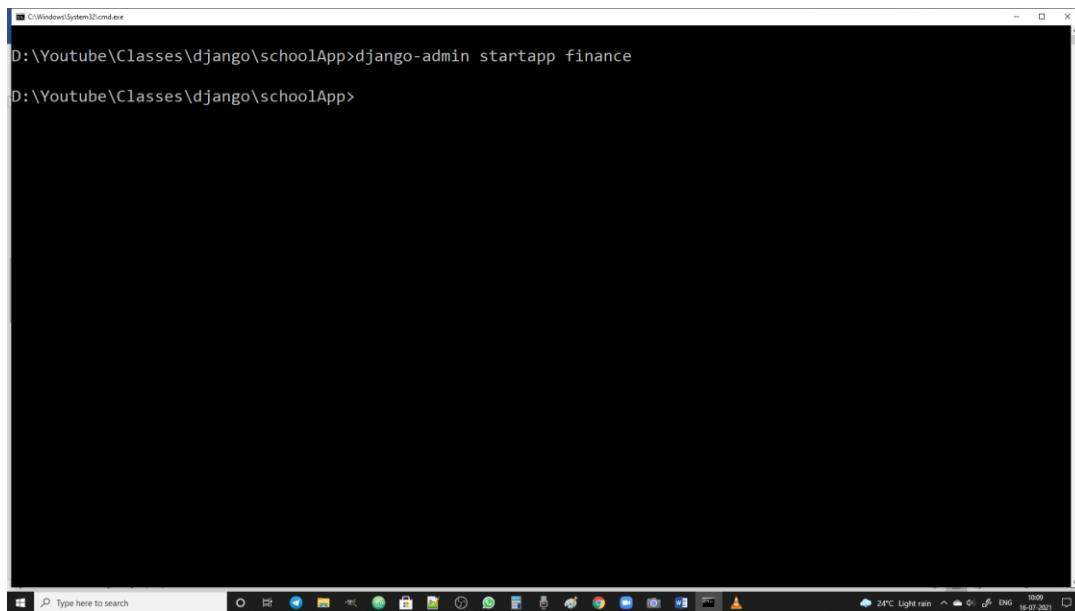
In the same way when client request with admreport/ url, then admissionsReport view will be executed.

Observe here that instead of ip address 127.0.0.1 we used localhost. (Both are same)

## ADDING MULTIPLE APPLICATIONS TO OUR DJANGO PROJECT

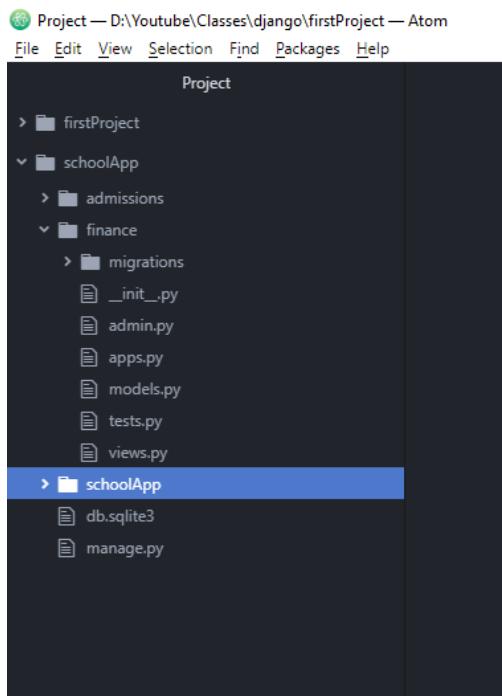
Adding multiple applications to the existing project, is same as just discussed.

Now let us add finance project to our schoolApp project

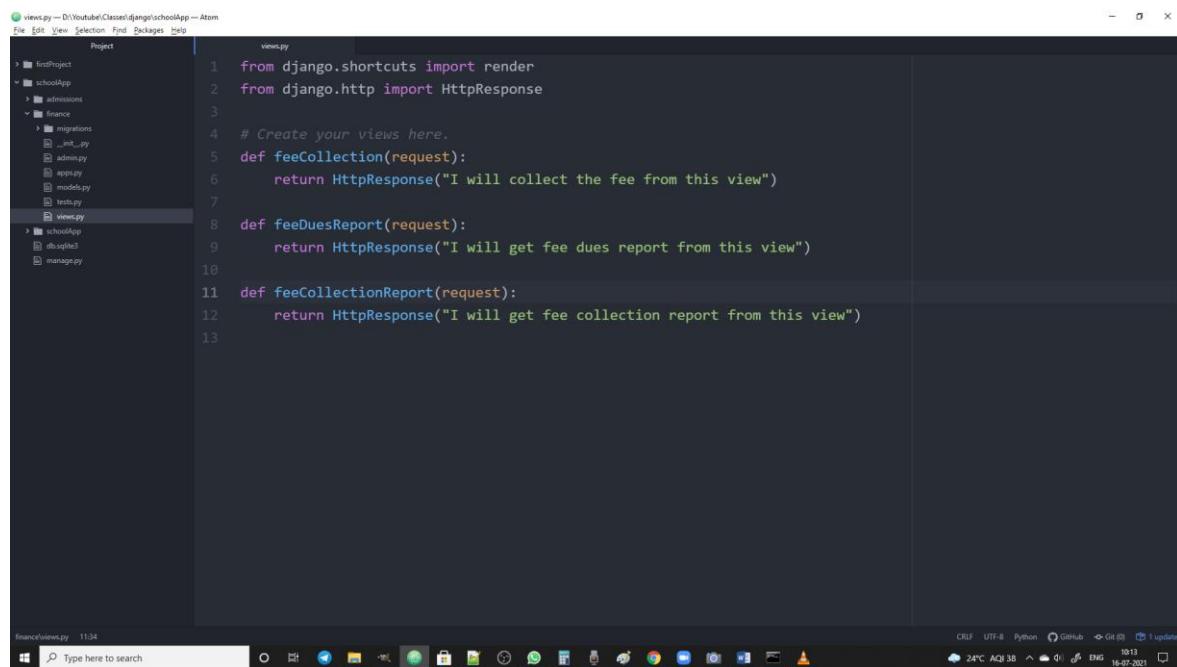


```
C:\Windows\System32\cmd.exe
D:\Youtube\Classes\django\schoolApp>django-admin startapp finance
D:\Youtube\Classes\django\schoolApp>
```

finance app is created now.



Now let us add views to the finance application. These will be written in views.py of finance folder



```

views.py — D:\Youtube\Classes\django\schoolApp — Atom
File Edit View Selection Find Packages Help
Project
> firstProject
- schoolApp
  > admissions
  - finance
    > migrations
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
    views.py
  > schoolApp
    __init__.py
    manage.py
  finance
  views.py

views.py
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 # Create your views here.
5 def feeCollection(request):
6     return HttpResponse("I will collect the fee from this view")
7
8 def feeDuesReport(request):
9     return HttpResponse("I will get fee dues report from this view")
10
11 def feeCollectionReport(request):
12     return HttpResponse("I will get fee collection report from this view")
13

```

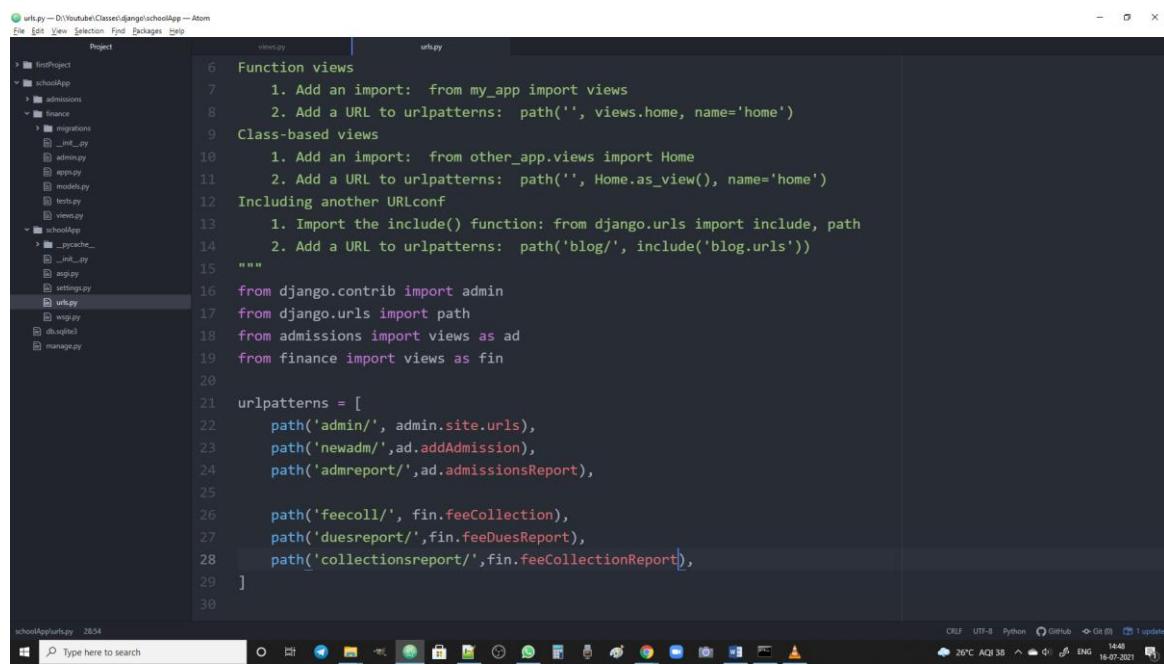
finance\views.py 11:34

Type here to search

CR LF UTF-8 Python GitHub Git (S) 1 update

24°C AQI 38 ⚡ ENG 10:13 16-07-2021

Now let us map the urls



```

urls.py — D:\Youtube\Classes\django\schoolApp — Atom
File Edit View Selection Find Packages Help
Project
> firstProject
- schoolApp
  > admissions
  - finance
    > migrations
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
    views.py
  > schoolApp
    __init__.py
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  db.sqlite3
  manage.py

views.py
url.py

6 Function views
7   1. Add an import: from my_app import views
8   2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10  1. Add an import: from other_app.views import Home
11  2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13  1. Import the include() function: from django.urls import include, path
14  2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18 from admissions import views as ad
19 from finance import views as fin
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('newadm/', ad.addAdmission),
24     path('admreport/', ad.admissionsReport),
25
26     path('feecoll/', fin.feeCollection),
27     path('duesreport/', fin.feeDuesReport),
28     path('collectionsreport/', fin.feeCollectionReport),
29 ]
30

schoolApp\urls.py 28:54
Type here to search
CR LF UTF-8 Python GitHub Git (S) 1 update
26°C AQI 38 ⚡ ENG 14:48 16-07-2021

```

Observe here that, while importing views in finance, we introduced alias names ad,fin which corresponds to admissions, finance respectively. Otherwise both views (admissions, finance) will collided with each other



Now let us start our server and run the program.



I will collect the fee from this view



I will get fee dues report from this view



I will get fee collection report from this view

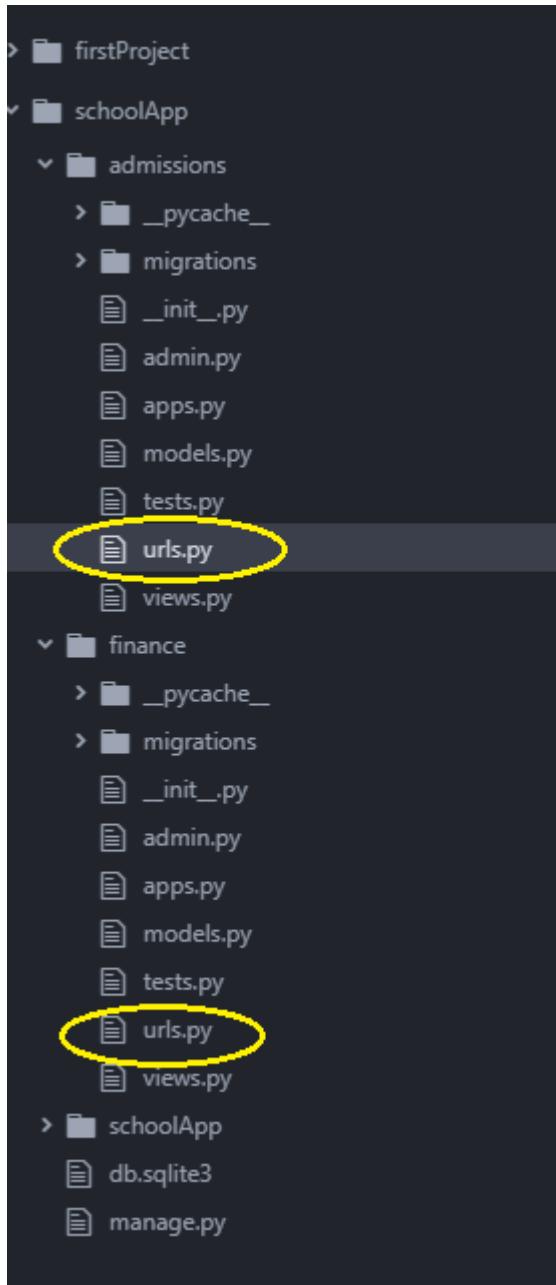


## APPLICATION LEVEL URL MAPPINGS

Instead of writing all url mappings in the project level url mappings file (urls.py) we can do mappings in application level urls.py files also.

This will reduce the burden of using alias names (ad, fin which we used in previous topic).

First let us create urls.py files under each application



now let us implement application level mappings

The following is the admissions application's url mappings file

The screenshot shows the Atom IDE interface with the following details:

- Project Tree:** On the left, under the "Project" tab, the file structure is displayed:
  - firstProject
  - schoolApp
    - admissions
    - finance
  - migrations
  - utils
  - views
- Code Editor:** The main area shows a Python file named `admissionsurls.py`. The code defines URL patterns for adding admissions and generating reports.

```
from django.urls import path
from admissions.views import addAdmission
from admissions.views import admissionsReport

urlpatterns = [
    path('newadm/',addAdmission),
    path('admreport/',admissionsReport),
]
```
- Status Bar:** At the bottom, it shows the file name `admissionsurls.py`, line number `12:1`, and system information including temperature (`25°C`), weather (`Heavy I-storms`), and date/time (`17-07-2021`).

Following is the finance application's mappings

The screenshot shows the Atom IDE interface with the following details:

- Project View:** On the left, the project structure is displayed under "Project". It includes a "firstProject" folder containing "schoolApp" and "finance" subfolders. "schoolApp" contains "admissions", "migrations", and "tests.py". "finance" contains "models.py" and "tests.py".
- Code Editor:** The main area shows the "urls.py" file for the "finance" app. The code defines URL patterns for three views: "feeCollection", "feeDuesReport", and "feeCollectionReport".

```
from django.urls import path
from finance.views import feeCollection
from finance.views import feeDuesReport
from finance.views import feeCollectionReport

urlpatterns = [
    path('feecoll/', feeCollection),
    path('duesreport/',feeDuesReport),
    path('collectionsreport/',feeCollectionReport),
]
```
- Status Bar:** At the bottom, the status bar shows "financials.py 4:14", "CRU UTF-8 Python GitHub Git 1 update", and system icons for weather, battery, and network.

observe here that in each url mapping file, we individually imported that application's views and implemented the corresponding mappings.

now let us see the project level url mappings file.

The screenshot shows the Atom code editor with the following file structure:

```

schoolApp/
  - __pycache__
  - migrations
  - admisions
  - models.py
  - tests.py
  - urls.py
  - views.py
  - __init__.py
  - admin.py
  - apps.py
  - schoolApp/
    - __pycache__
    - migrations
    - admisions/
      - __init__.py
      - models.py
      - tests.py
      - urls.py
      - views.py
    - finance/
      - __init__.py
      - migrations
      - admisions/
        - __init__.py
        - models.py
        - tests.py
      - finance/
        - __init__.py
        - migrations
        - admisions/
          - __init__.py
          - models.py
          - tests.py
        - finance/
          - __init__.py
          - migrations
          - admisions/
            - __init__.py
            - models.py
            - tests.py
        - urls.py
      - urls.py
      - views.py
    - __init__.py
    - db.sqlite3
    - manage.py
  - __init__.py
  - urls.py

```

The `urls.py` file content is as follows:

```

1  """schoolApp URL Configuration
2
3  The `urlpatterns` list routes URLs to views. For more information please see:
4      https://docs.djangoproject.com/en/3.1/topics/http/urls/
5  Examples:
6      Function views
7          1. Add an import: from my_app import views
8              2. Add a URL to urlpatterns: path('', views.home, name='home')
9      Class-based views
10         1. Add an import: from other_app.views import Home
11             2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12     Including another URLconf
13         1. Import the include() function: from django.urls import include, path
14             2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16
17 from django.contrib import admin
18 from django.urls import path
19 from django.conf.urls import include
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('ad/', include('admisions.urls')),
24     path('fin/', include('finance.urls')),
25 ]

```

Above pic shows project level url mappings.

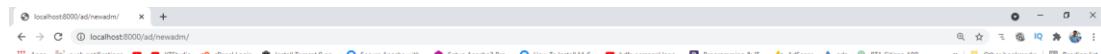
As application level mappings are already separated, we just need to include those url mappings into our project level `urls.py` by using `include` function.

`include` is available in `django.conf.urls`

Now we will run the application as follows

whenever we want to run admissions application's url `newadm` we need to add these project level mapping `/ad/` to the actual url as follows.

<http://localhost:8000/ad/newadm>



This is add admissions view



In the same way, whenever we want to run finance application's url feecoll we need to add these project level mapping /fin/ to the actual url as follows.

<http://localhost:8000/fin/feecoll>



I will collect the fee from this view



## CREATING TEMPLATES IN DJANGO

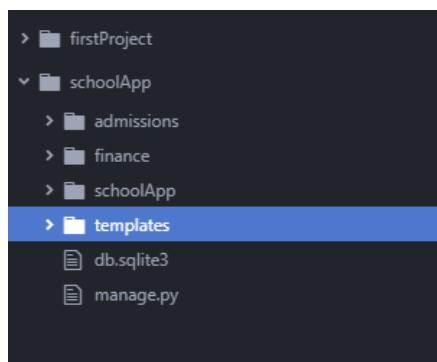
We use templates to generate dynamic responses.

Steps involved in creating templates

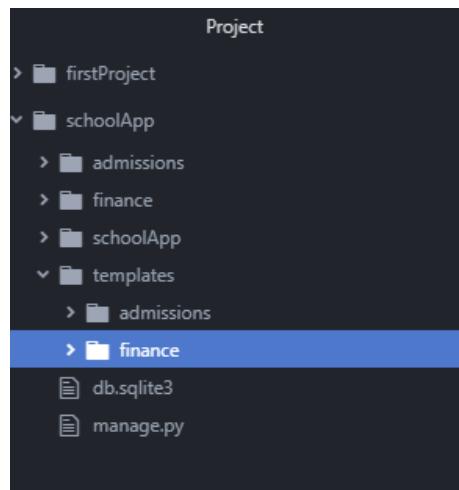
- 1) Setup the templates folder
- 2) setup DIRS in settings.py
- 3) Create a template
- 4) Use template in views
- 5) execute the project

**Setup the templates folder:**

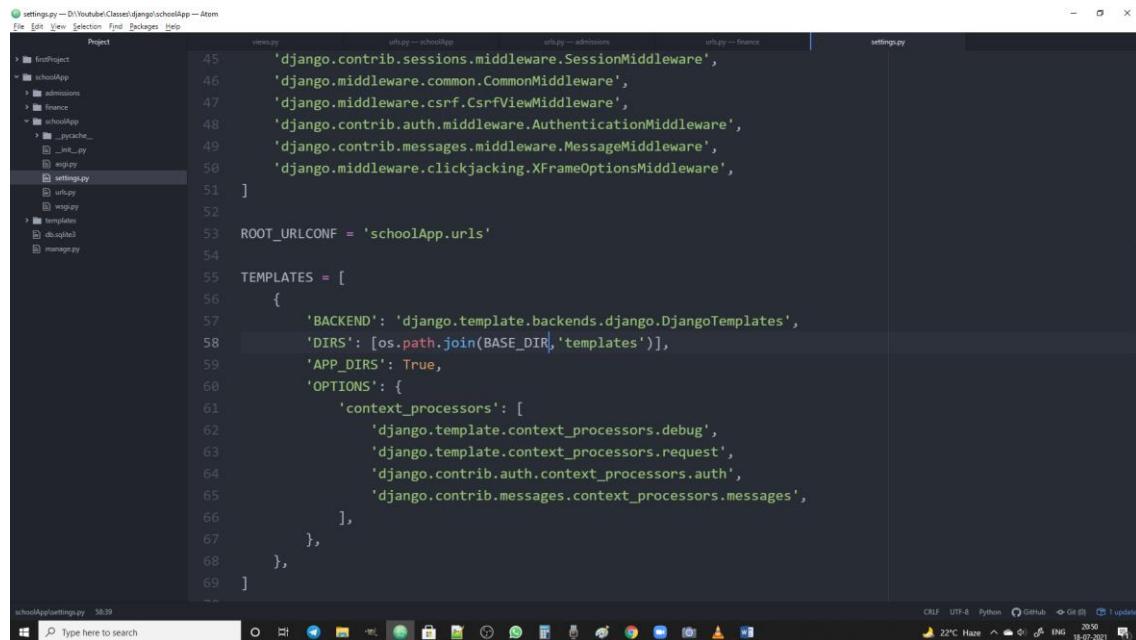
Create templates folder under project folder 'schoolApp'



To maintain application wise templates separately, let us create two subfolders under templates with the names admissions, finance



### Setup DIRS in settings.py:



```

settings.py — D:\Youtube\Class\django\schoolApp — Atom
File Edit View Selection Find Packages Help
Project schoolApp admissions finance schoolApp _init_.py __init__.py settings.py urls.py wsgi.py templates db.sqlite3 manage.py
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
51 ]
52
53 ROOT_URLCONF = 'schoolApp.urls'
54
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates',
58         'DIRS': [os.path.join(BASE_DIR,'templates')],  

59         'APP_DIRS': True,
60         'OPTIONS': {
61             'context_processors': [
62                 'django.template.context_processors.debug',
63                 'django.template.context_processors.request',
64                 'django.contrib.auth.context_processors.auth',
65                 'django.contrib.messages.context_processors.messages',
66             ],
67         },
68     },
69 ]

```

The screenshot shows the Atom code editor with the file 'settings.py' open. The code is configuring the Django template system. It includes middleware like SessionMiddleware, CommonMiddleware, CsrfViewMiddleware, AuthenticationMiddleware, MessageMiddleware, and XFrameOptionsMiddleware. The 'TEMPLATES' section is defined with 'DIRS' set to [os.path.join(BASE\_DIR,'templates')]. The 'APP\_DIRS' setting is set to True, which means Django will look for templates directly within each app's directory. The 'OPTIONS' section contains a 'context\_processors' list with debug, request, auth, and messages processors.

As shown above, we need to set the DIRS of TEMPALTES in settings.py as

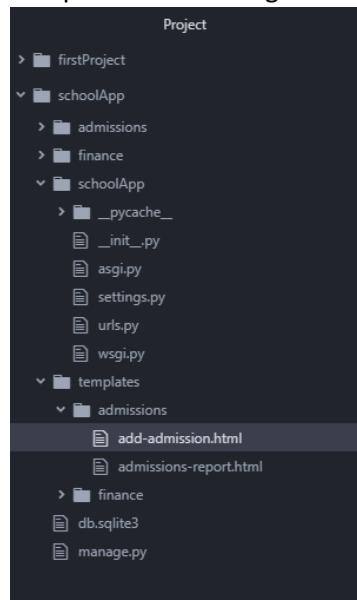
'DIRS': [os.path.join(BASE\_DIR,'templates')],

which means that we are informing to the system that we will place all our templates under folder called 'templates' which is available in project directory (BASE\_DIR indicates project directory).

Before doing above setting, we need to import os module in order to set the templates.

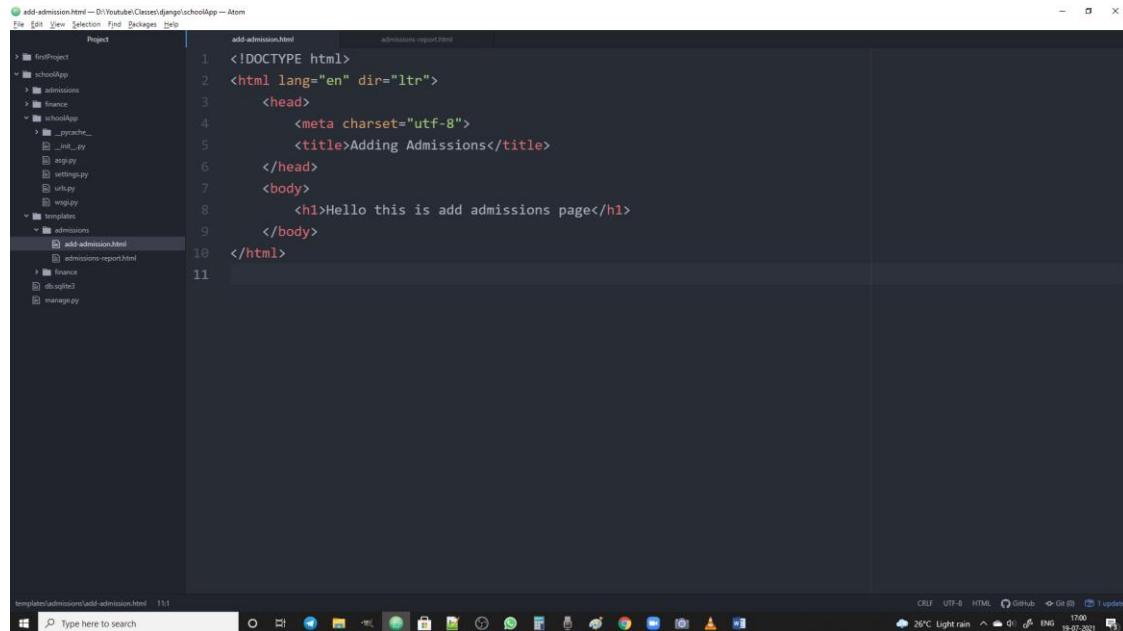
### Creating Templates:

Templates are nothing but html files. Now let us create our templates under templates folder.

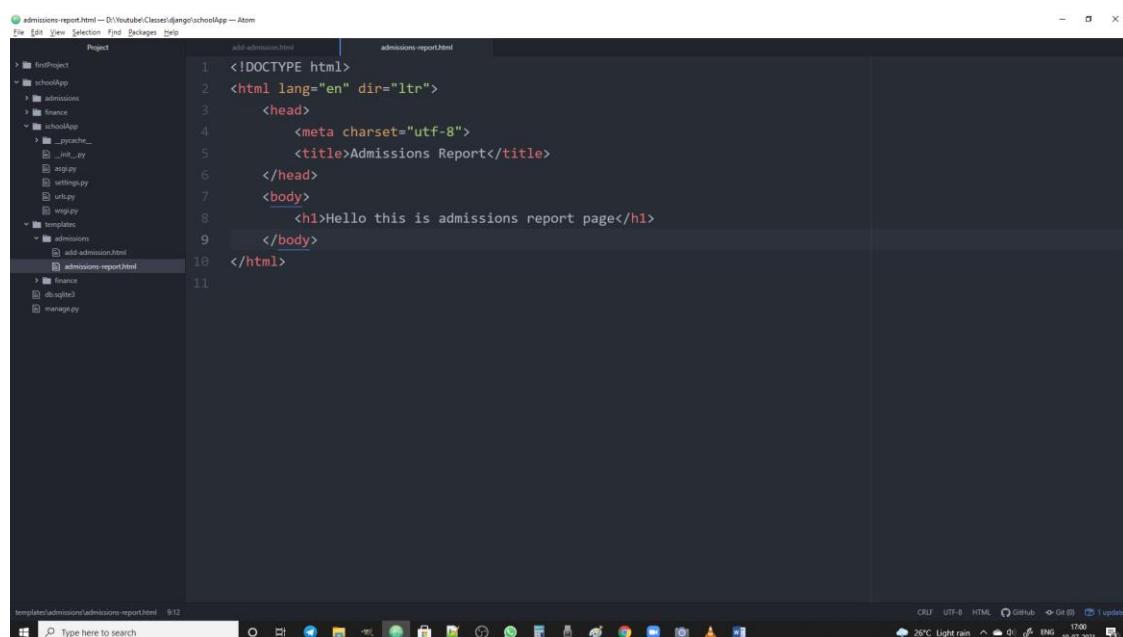


We created two template files (add-admission.html, admissions-report.html) under admissions folder of templates.

Below codes shows how these two html files were Implemented.



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title>Adding Admissions</title>
    </head>
    <body>
        <h1>Hello this is add admissions page</h1>
    </body>
</html>
```



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title>Admissions Report</title>
    </head>
    <body>
        <h1>Hello this is admissions report page</h1>
    </body>
</html>
```

### Using templates in views:

By using render, we can link templates and views.

`render (request, template)`

The screenshot shows the Atom code editor interface. On the left is a sidebar titled 'Project' listing several Django applications: firstProject, schoolApp, admissions, finance, and templates. The main editor area displays Python code in 'views.py':

```

1 from django.shortcuts import render
2
3
4 # Create your views here.
5 #function based views
6
7 def addAdmission(request):
8     return render(request,'admissions/add-admission.html')
9
10 def admissionsReport(request):
11     return render(request,'admissions/admissions-report.html')
12

```

The status bar at the bottom shows the file path 'admissions/views.py' and the line count '11:81'. The taskbar at the bottom includes icons for various applications like File Explorer, Task Manager, and a browser.

### Execute the project:

The screenshot shows a Windows Command Prompt window. The user has navigated to the directory 'D:\Youtube\Classes\django\schoolApp' and run the command 'python manage.py runserver'. The output shows the server is running on port 8000.

```

C:\Windows\system32\cmd.exe - python manage.py runserver
D:\Youtube\Classes\django\schoolApp>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s)
): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 19, 2021 - 17:06:53
Django version 3.1, using settings 'schoolApp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

```

The taskbar at the bottom shows the date and time as '25°C Light rain 17:05 19-07-2021'.

Now let us run the application and check the output is taken from html templates or not.



## Hello this is add admissions page



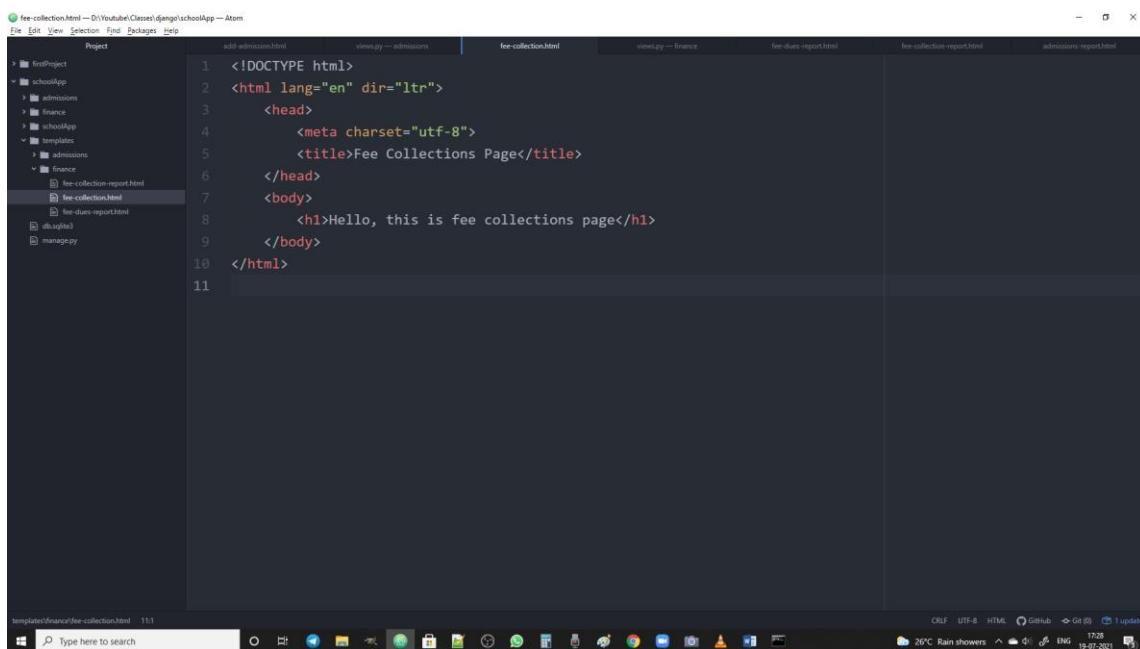
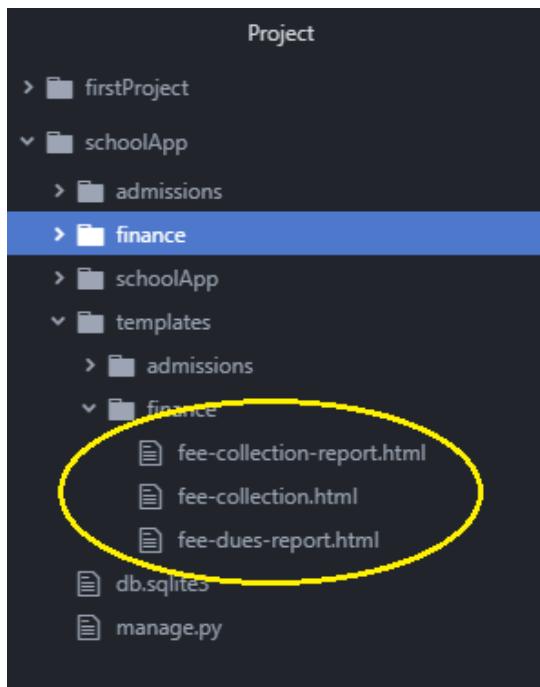
## Hello this is admissions report page



Observe here that views are linked with templates successfully.

Whenever a view is activated, corresponding template is displayed as output.

In the same way we can create three templates under finance folder of templates, and link those templates with views in views.py under finance application.



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
    <meta charset="utf-8">
    <title>Fee Collections Page</title>
</head>
<body>
    <h1>Hello, this is fee collections page</h1>
</body>
</html>
```



fee-dues-report.html — D:\Youtube\Classes\django\schoolApp — Atom

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Fee Dues Page</title>
  </head>
  <body>
    <h1>Hello, this is fee dues page</h1>
  </body>
</html>
```

File Edit View Selection Fnd Packages Help

Project add-admission.html views.py — admissions fee-collection.html views.py — finance fee-dues-report.html fee-collection-report.html admissions-report.html

fee-collection-report.html — D:\Youtube\Classes\django\schoolApp — Atom

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Fee Collection Report Page</title>
  </head>
  <body>
    <h1>Hello, this is fee collections report page</h1>
  </body>
</html>
```

File Edit View Selection Fnd Packages Help

Project add-admission.html views.py — admissions fee-collection.html views.py — finance fee-dues-report.html fee-collection-report.html admissions-report.html

The screenshot shows the Atom code editor with the file `views.py` open. The code defines three view functions: `feeCollection`, `feeDuesReport`, and `feeCollectionReport`. The `feeCollection` function returns a render to the template `'finance/fee-collection.html'`. The `feeDuesReport` function returns a render to the template `'finance/fee-dues-report.html'`. The `feeCollectionReport` function returns a render to the template `'finance/fee-collection-report.html'`.

```

from django.shortcuts import render

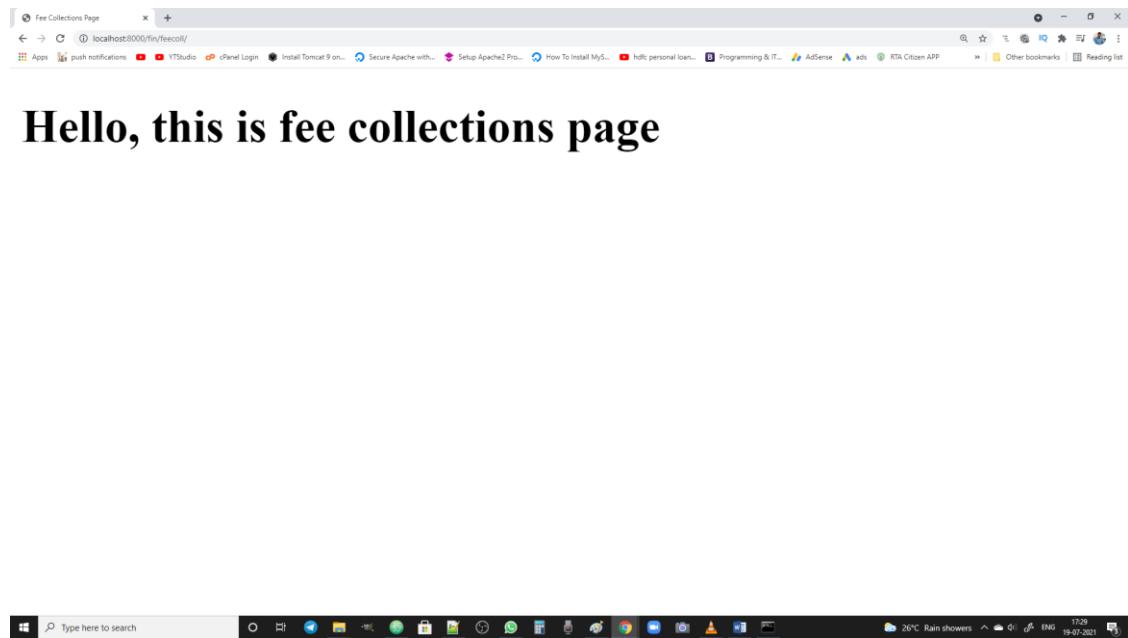
# Create your views here.
def feeCollection(request):
    return render(request, 'finance/fee-collection.html')

def feeDuesReport(request):
    return render(request, 'finance/fee-dues-report.html')

def feeCollectionReport(request):
    return render(request, 'finance/fee-collection-report.html')

```

Now we can see the finance template files as output when we run finance views (same as admissions part)



## SENDING DATA FROM VIEW TO TEMPLATES

We can send data from view to templates so that we will generate the dynamic output by filling these values in the template file.

We generally use dictionaries in order to send the data to the templates as follows.

```
def addAdmission(request):
    values = {"name": "Santosh", "age": 36, "Address": "Visakhapatnam"}
    return render(request, 'admissions/add-admission.html', values)
```

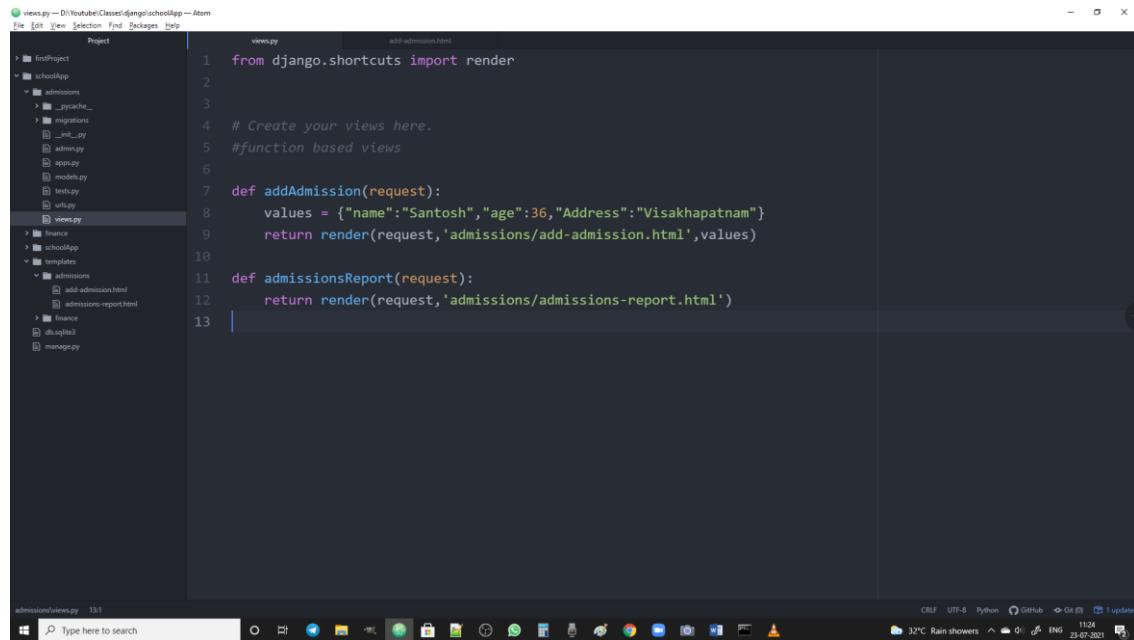
Observe here that a dictionary called 'values' is created and passed to the template through render method.

Now this dictionary values will be used in the template add-admission.html and dynamic response will be generated with the values in this dictionary as follows.

for example we can print the value of age in template as

```
{% age %}
```

Observe the full example below



The screenshot shows the Atom code editor interface. On the left, the project structure is visible, showing a 'firstProject' folder containing 'schoolApp' and 'views.py'. The 'views.py' file is open in the main editor area, displaying Python code for defining two views: 'addAdmission' and 'admissionsReport'. The 'add-admission.html' template file is shown in the preview pane on the right. The status bar at the bottom indicates the file is 13.5 lines long and shows system information like weather (32°C Rain showers), battery level (11:24), and network connection.

```
views.py — D:\Youtubel-Classes\djangoSchoolApp — Atom
File Edit View Selection Find Packages Help
Project
firstProject
  schoolApp
    admissions
      migrations
        __pycache__
        migrations
          __init__.py
          adm.py
          apps.py
          models.py
          tests.py
          urls.py
    views.py
  finance
  schoolApp
  templates
    admissions
      add-admission.html
      admissions-report.html
    finance
    db.sqlite3
    manage.py

views.py
admissions\views.py 13.5
from django.shortcuts import render
# Create your views here.
#function based views
def addAdmission(request):
    values = {"name": "Santosh", "age": 36, "Address": "Visakhapatnam"}
    return render(request, 'admissions/add-admission.html', values)
def admissionsReport(request):
    return render(request, 'admissions/admissions-report.html')

```



**Hello Santosh, welcome to add admissions page**

**Your age is 36**

**Your address is**

**Your score is**



Here observe that response is created dynamically by filling the values received from view.

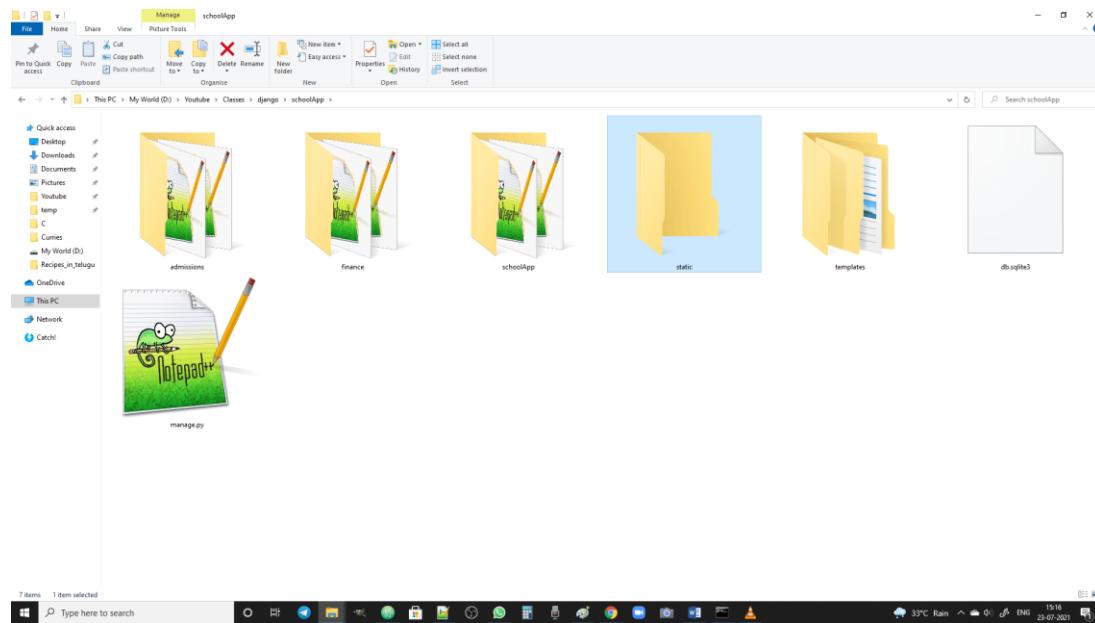
And also observe that score field is not available in the dictionary which is created in views.

So, in template, score is shown as empty as it is not received through dictionary.

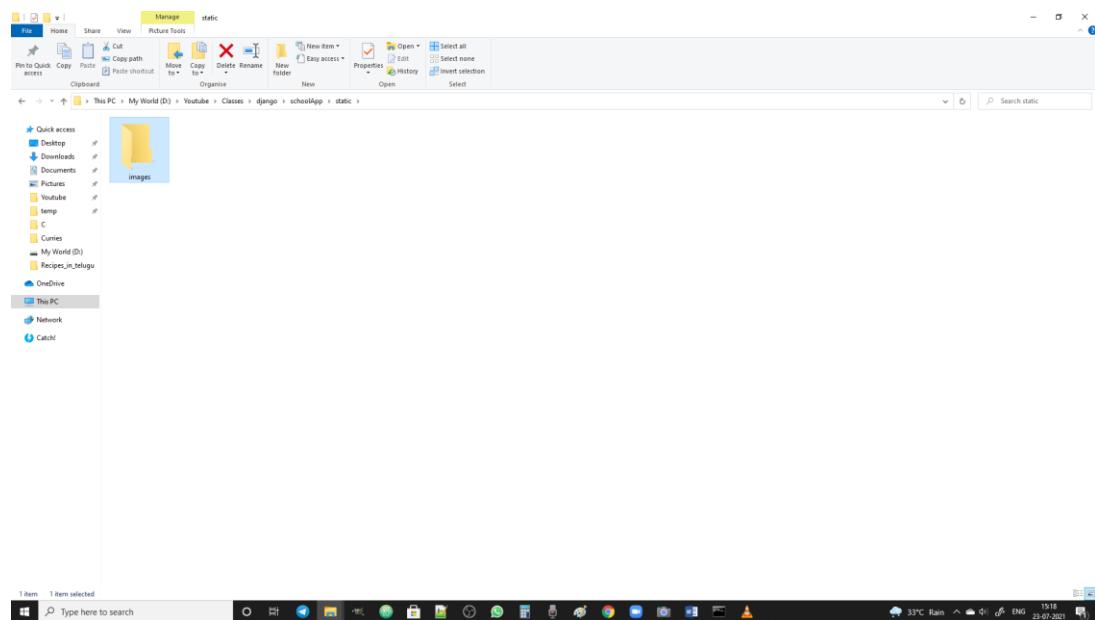
### INSERTING IMAGES INTO TEMPLATES

To insert images into our templates,

Step – 1: We need to create a sub folder called ‘static’ in our project’s folder.

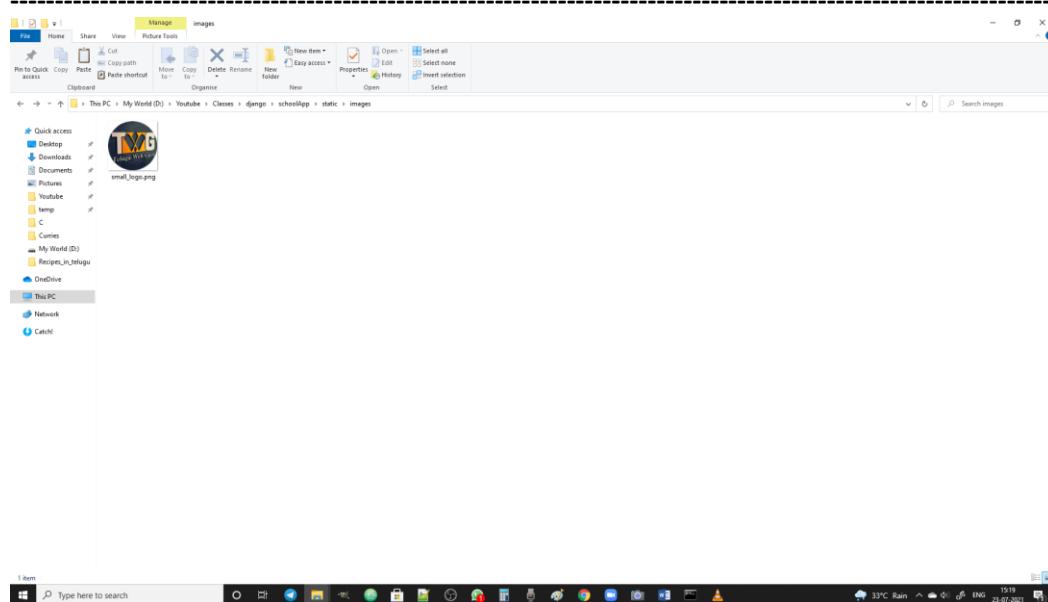


We may maintain separate folders (for images , logos etc., ) inside the static folder



We may store the images in this images folder which are need to be inserted into our templates.

For example I stored one image called small\_logo.png as follows.



## Step-2: Set the STATICFILES\_DIRS setting in settings.py

```

settings.py -- D:\YouTube\Classes\django\schoolapp -- Atom
File Edit View Selection Find Packages Help
Project schoolapp
  - schoolapp
    - migrations
      - __init__.py
      - admin.py
      - apps.py
      - models.py
      - tests.py
      - tests.py
      - views.py
    - static
    - templates
      - admissions
        - add-admission.html
        - admissions-report.html
      - finance
      - db.sqlite3
      - manage.py
    - settings.py
      - __init__.py
      - wsgi.py
    - urls.py
  - static
  - templates
    - admissions
      - add-admission.html
      - admissions-report.html
    - finance
    - db.sqlite3
    - manage.py

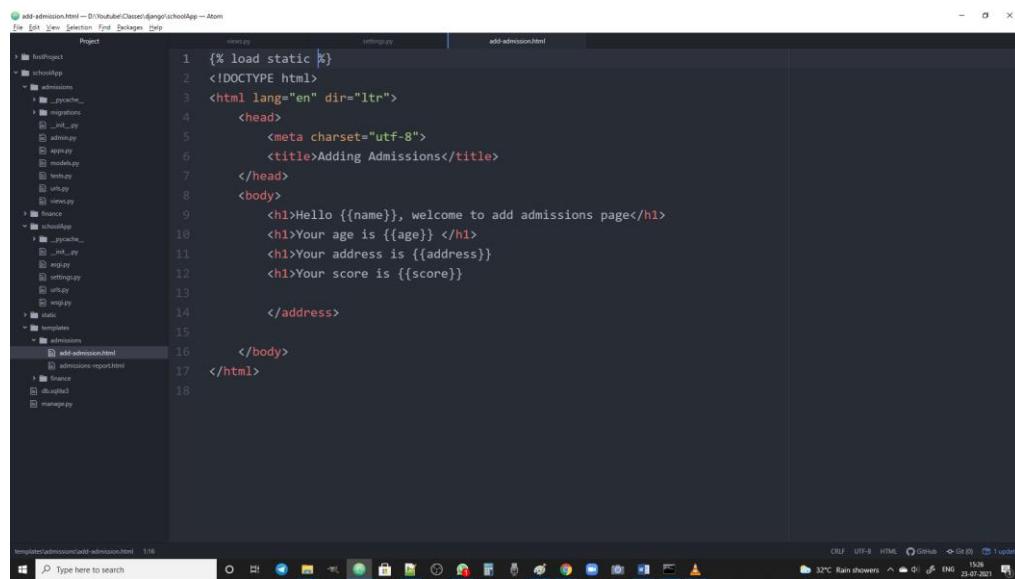
settings.py 123:19
102
103
104 # Internationalization
105 # https://docs.djangoproject.com/en/3.1/topics/i18n/
106
107 LANGUAGE_CODE = 'en-us'
108
109 TIME_ZONE = 'UTC'
110
111 USE_I18N = True
112
113 USE_L10N = True
114
115 USE_TZ = True
116
117
118 # Static files (CSS, JavaScript, Images)
119 # https://docs.djangoproject.com/en/3.1/howto/static-files/
120
121 STATIC_URL = '/static/'
122
123 STATICFILES_DIRS = [
124     os.path.join(BASE_DIR, 'static')
125 ]
126

```

The screenshot shows the PyCharm IDE with the 'settings.py' file open. The code editor highlights the 'STATICFILES\_DIRS' setting. The desktop background is visible at the bottom.

## Step-3: Load the static files into templates.

To achieve this, we need to use template tags {%. . . . %}

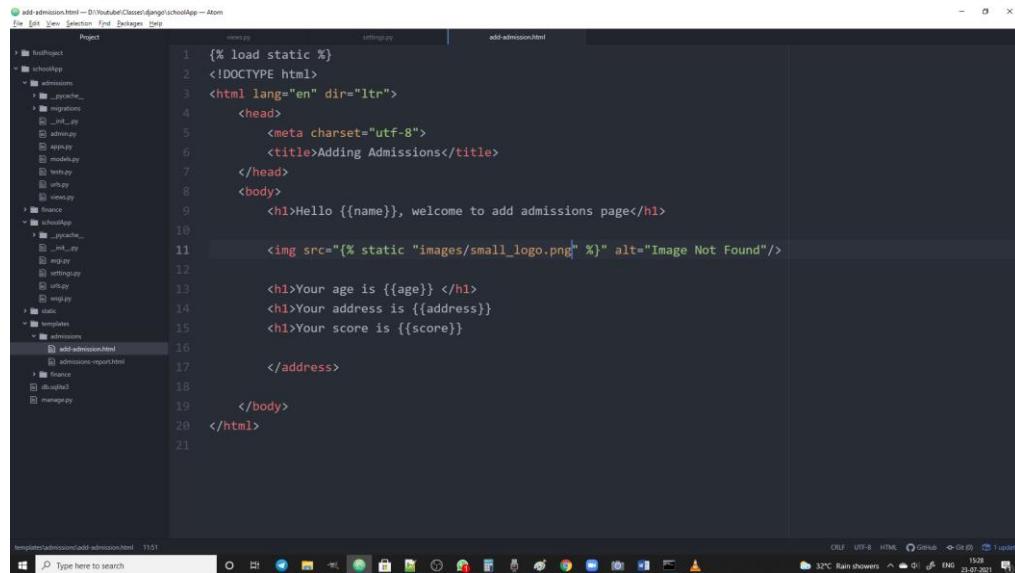


```

1  {% load static %} 
2  <!DOCTYPE html>
3  <html lang="en" dir="ltr">
4      <head>
5          <meta charset="utf-8">
6          <title>Adding Admissions</title>
7      </head>
8      <body>
9          <h1>Hello {{name}}, welcome to add admissions page</h1>
10         <h1>Your age is {{age}}</h1>
11         <h1>Your address is {{address}}</h1>
12         <h1>Your score is {{score}}</h1>
13     </body>
14 </html>
15
16
17
18

```

Step-4: Insert images by using img tag as shown below.

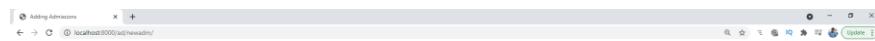


```

1  {% load static %} 
2  <!DOCTYPE html>
3  <html lang="en" dir="ltr">
4      <head>
5          <meta charset="utf-8">
6          <title>Adding Admissions</title>
7      </head>
8      <body>
9          <h1>Hello {{name}}, welcome to add admissions page</h1>
10         
11
12         <h1>Your age is {{age}}</h1>
13         <h1>Your address is {{address}}</h1>
14         <h1>Your score is {{score}}</h1>
15
16
17
18
19
20 </body>
21 </html>

```

After inserting images using img tag, we can run the server and start our project so that our templates are loaded with images.



Hello Santosh, welcome to add admissions page



Your age is 36

Your address is

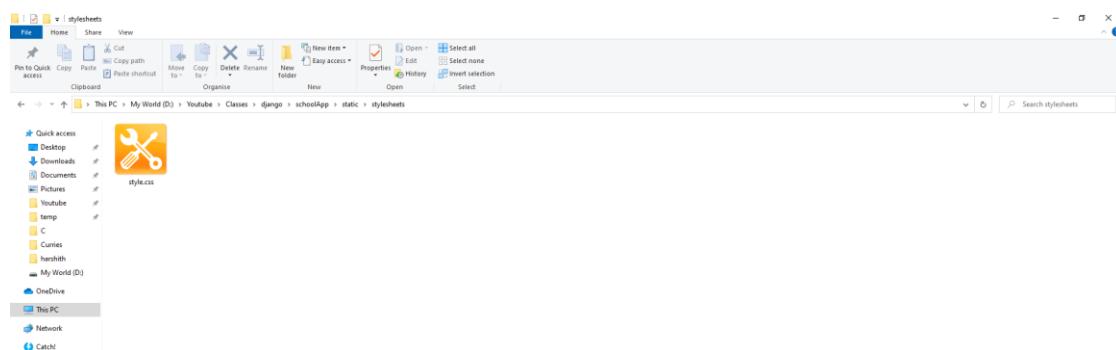
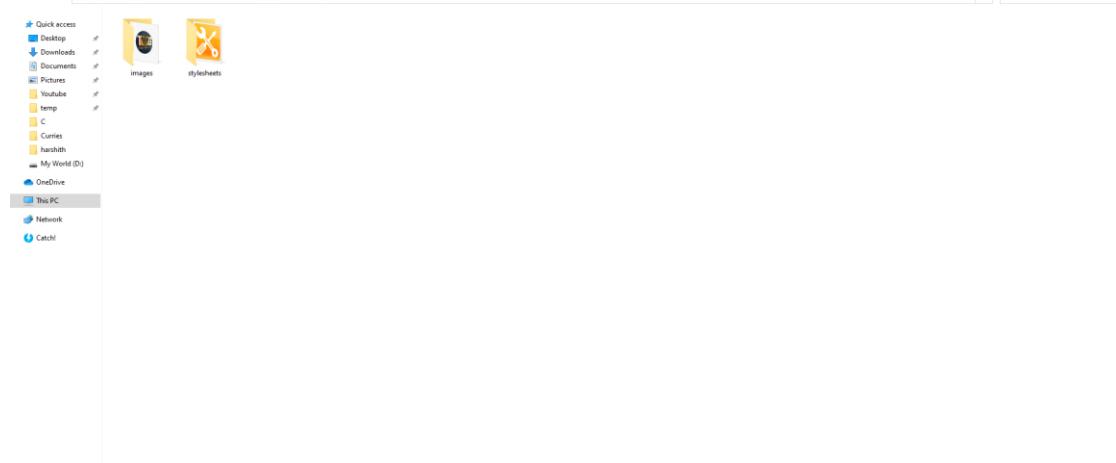
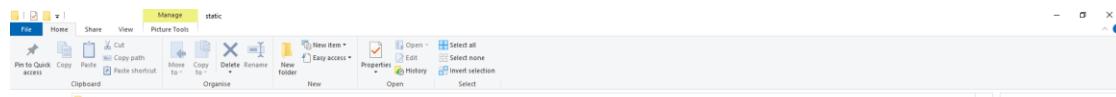
Your score is



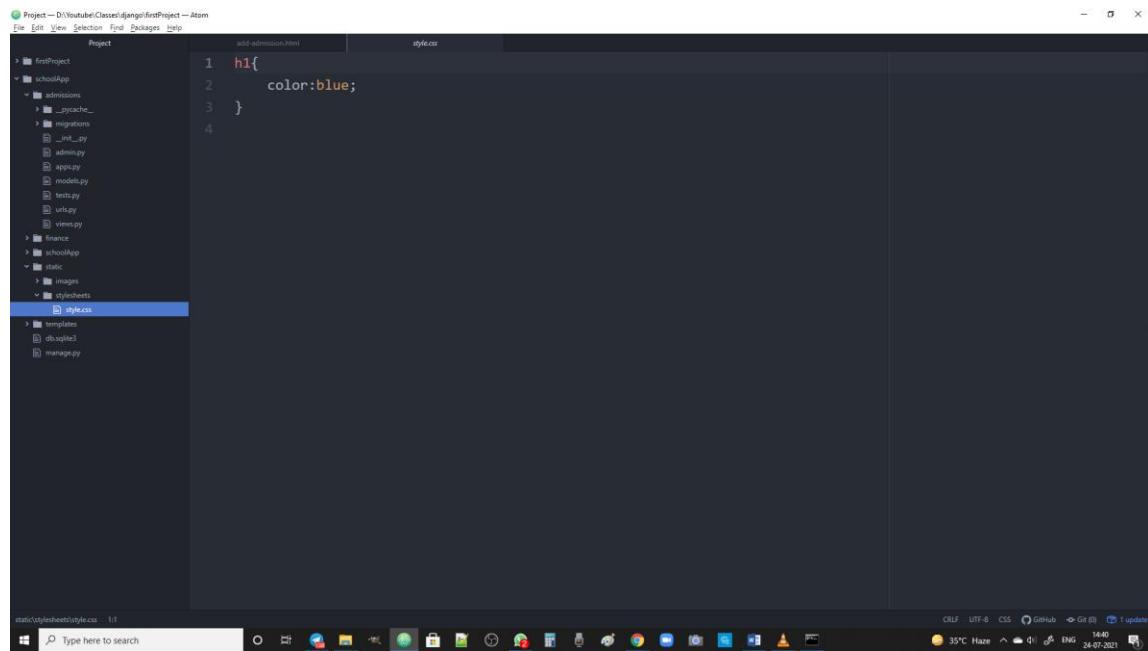
## LINKING STYLESHEETS WITH TEMPLATES

Linking stylesheets follows the same procedure, which we followed to insert images into template.

Let us create a folder stylesheet and create a style.css file into it



Now let us write styles code in style sheet file.



The screenshot shows the Atom code editor interface. The left sidebar displays a project structure for a Django application named 'firstProject'. The 'style.css' file is open in the main editor area, containing the following CSS code:

```

1 h1{
2     color:blue;
3 }
4

```

The status bar at the bottom indicates the file is 1:1, has 1 update, and shows system information like battery level, temperature (35°C), and date (24-07-2021).

Now let us link this css file with our templates so that all the content within h1 tag will be displayed in blue colour.



Hello Santosh, welcome to add admissions page



Your age is 36

Your address is

Your score is



This is how we can link our stylesheets with templates and create beautiful webpages.

### SETTING HOME PAGE

Now let us create a home page template with different navigations and set it as homepage

First let us create index.html template with all the available links as buttons.

```

index.html
1 <html lang="en" dir="ltr">
2   <head>
3     <meta charset="utf-8">
4     <title>Homepage</title>
5     <link rel="stylesheet" href="{% static "stylesheets/style.css" %}" />
6   </head>
7   <body>
8     <br/>
9     <br/>
10    <h2>
11      
12      Telugu Web Guru Academy
13    </h2><br/>
14    <div class="module">
15      <h3>Admissions</h3><br/>
16      <a href="/ad/newadm"><button type="button" name="button">Add Admission</button></a><br/><br/>
17      <a href="/ad/admreport"><button type="button" name="button">Admission Report</button></a><br/><br/>
18    </div>
19
20    <div class="module">
21      <h3>Finance</h3><br/>
22      <a href="/fin/feecoll"><button type="button" name="button">Fee Collection</button></a><br/><br/>
23      <a href="/fin/duesreport"><button type="button" name="button">Fee Collection Report</button></a><br/><br/>
24      <a href="/fin/collectionsreport"><button type="button" name="button">Fee Dues Report</button></a><br/><br/>
25    </div>
26
27  </body>
28
29 </html>
30

```

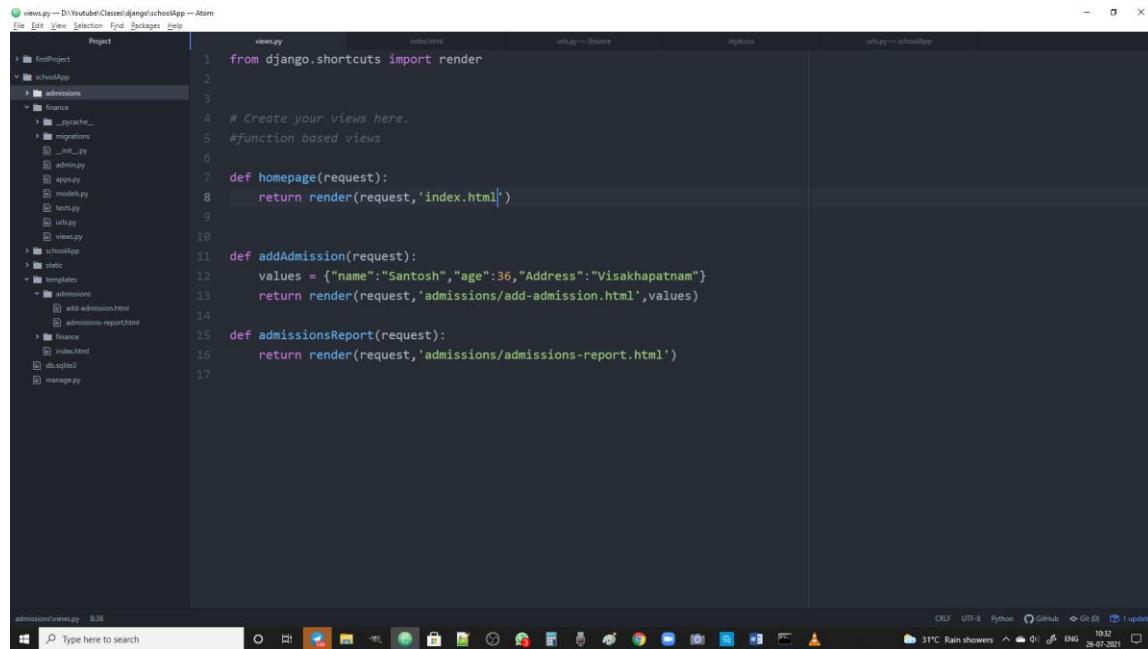
and write various styles to present the above web page beautiful way.

```

style.css
1 *{
2   margin:0;
3   padding: 0;
4 }
5 div{
6   border: 1px solid grey;
7   width:45%;
8   text-align: center;
9   float: left;
10  margin: 20px;
11  height:200px;
12  background-color: #F1F1F1;
13 }
14 h3{
15   color:red;
16   background-color: yellow;
17 }
18 img{
19   width: 25px;
20   height: 25px;
21 }
22 h2{
23   text-align: center;
24 }
25 button{
26   width:150px;
27   background-color: grey;
28   color: white;
29 }
30 button:hover{
31   background-color: white;
32   color:grey;
33 }
34

```

Now let us create a view and link above template index.html with this view.

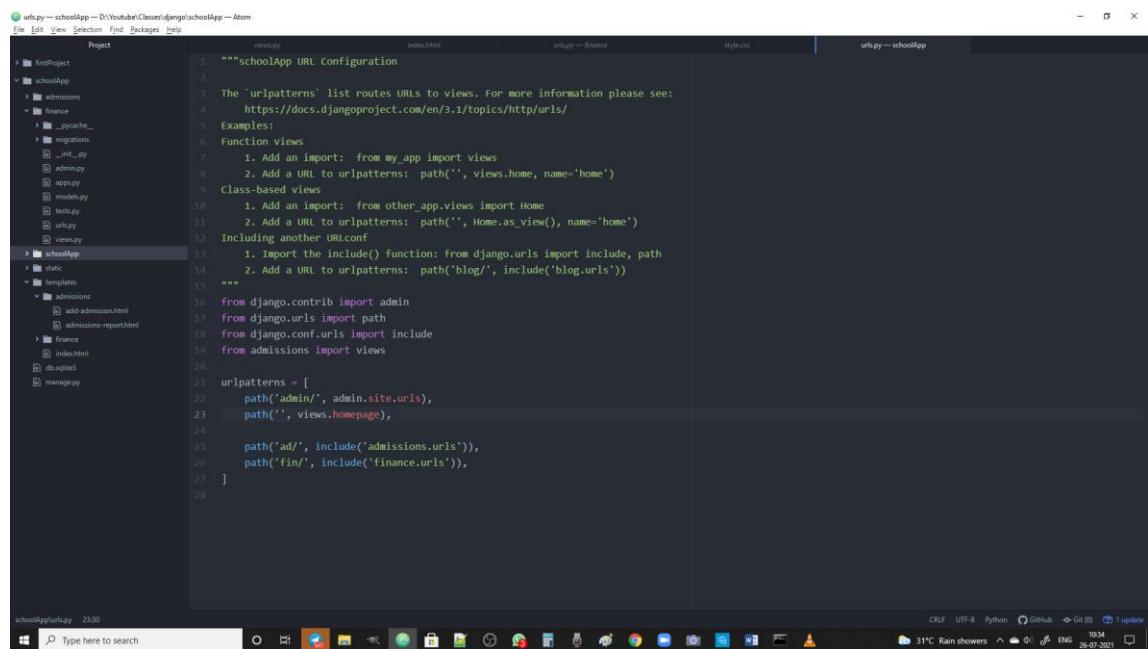


```

views.py
1 from django.shortcuts import render
2
3
4 # Create your views here.
5 #function based views
6
7 def homepage(request):
8     return render(request,'index.html')
9
10
11 def addAdmission(request):
12     values = {"name":"Santosh","age":36,"Address":"Visakhapatnam"}
13     return render(request,'admissions/add-admission.html',values)
14
15 def admissionsReport(request):
16     return render(request, 'admissions/admissions-report.html')
17

```

Let us map the view homepage with an url in urls.py so that it will be set as homepage.



```

urls.py
1 """SchoolApp URL Configuration
2
3 The `urlpatterns` list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/3.1/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import: from my_app import views
8     2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns: path("blog/", include('blog.urls'))
15 """
16
17 from django.contrib import admin
18 from django.urls import path
19 from django.conf.urls import include
20 from admissions import views
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('', views.homepage),
25
26     path('ad/', include('admissions.urls')),
27     path('fin/', include('finance.urls')),
28 ]

```

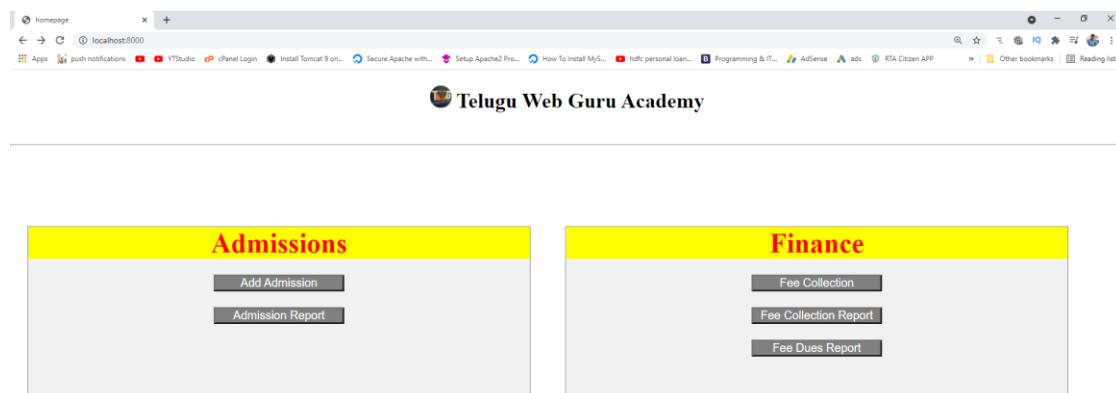
Now everything set.

Let us run the server and check whether the above page is set as homepage or not

```
C:\Windows\System32\cmd.exe : python manage.py runserver
D:\Youtube\Classes\django\schoolApp>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s)
): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 26, 2021 - 10:35:24
Django version 3.1, using settings 'schoolApp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

and the output is:



The screenshot shows a web browser window with the address bar set to 'localhost:8000'. The page content is divided into two main sections: 'Admissions' and 'Finance'. The 'Admissions' section contains two buttons: 'Add Admission' and 'Admission Report'. The 'Finance' section contains three buttons: 'Fee Collection', 'Fee Collection Report', and 'Fee Dues Report'.

## INTRODUCTION TO MODELS

Model represents the data base tables.

So, for each table in the database we will create a separate model with table fields are declared as variables so that django will automatically create tables and prepares all the queries related to that table.

To create a model, we need to create a class that extends `django.db.models.Model`

once a model is created, we can create the sql code by using makemigrations

create the tables automatically by using migrate

Along with our tables these migrations will create other tables related to administration, security, session management,

## How to create a Model?

### 1) Configure database in settings.py

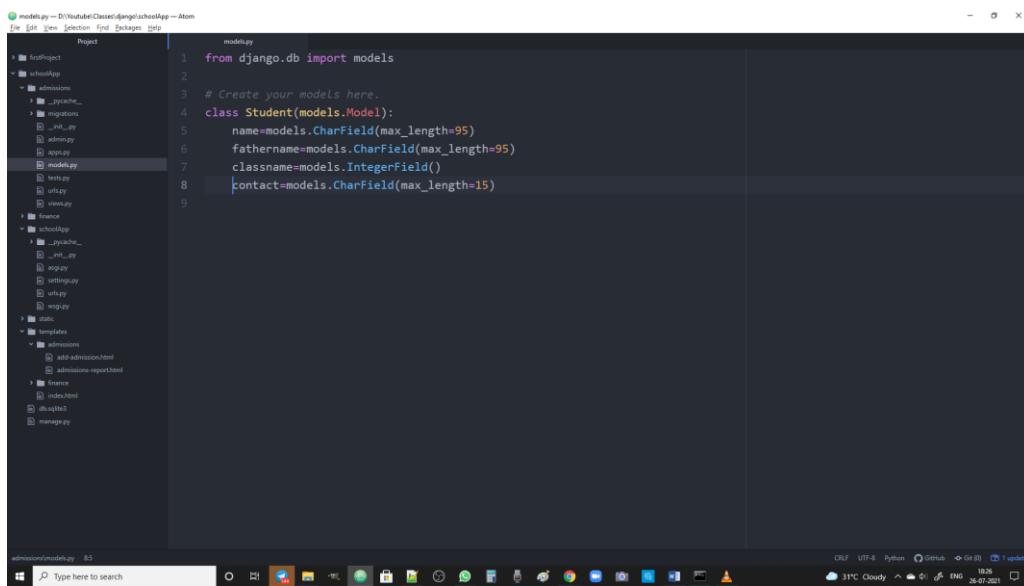
The screenshot shows the Atom code editor interface with the following details:

- File Path:** settings.py — D:\Youtube\Class1\django\schoolApp — Atom
- Project Structure:** The sidebar shows a project structure with several packages and files:
  - firstProject
  - schooldApp
    - admissions
    - finance
    - migrations
      - \_\_init\_\_.py
      - admin.py
      - apps.py
      - models.py
      - tests.py
      - utils.py
    - views.py
  - schoolApp
    - \_\_pycache\_\_
      - \_\_init\_\_.py
      - asgi.py
      - settings.py
      - urls.py
      - wsgi.py
    - static
    - templates
      - admissions
        - add-admission.html
        - address-report.html
      - base.html
      - index.html
      - style.css
    - management

- Code View:** The main pane displays the `settings.py` file content, which includes database configurations, password validation rules, and internationalization settings.
- Status Bar:** The bottom status bar shows the file path as "schoolApp/settings.py", the language as "Python", and the current commit status as "1 update".
- System Taskbar:** The Windows taskbar at the bottom shows various pinned icons and the system clock indicating it's 10:02 AM on a cloudy day.

we are using default database sqlite which came with django installation.

- 2) To create a model, open models.py file in our application folder
  - 3) Each model is a Python class that subclasses django.db.models.Model.
  - 4) Declare each attribute of the model represents a database field.



The screenshot shows the Atom code editor with a Python file named `models.py` open. The code defines a `Student` model with fields for name, fathername, classname, and contact.

```

from django.db import models

# Create your models here.

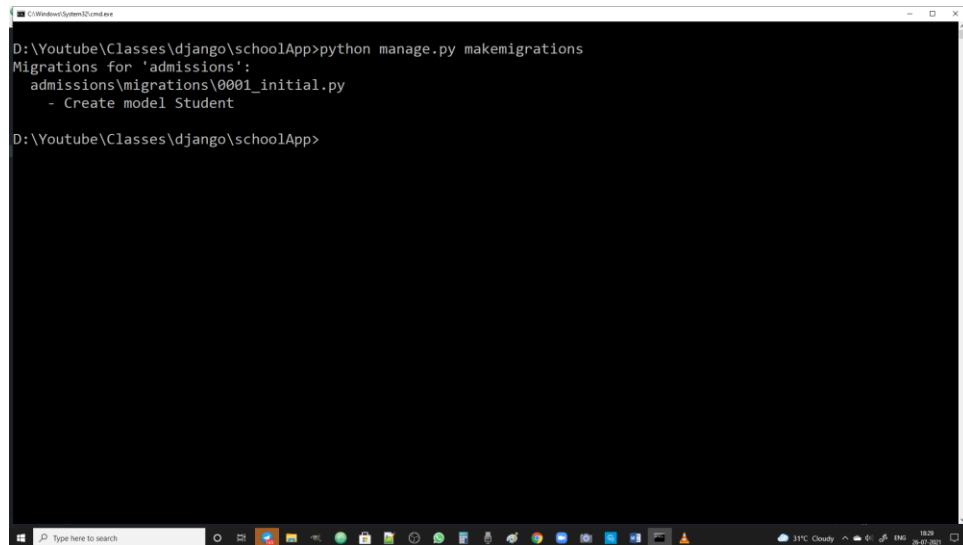
class Student(models.Model):
    name=models.CharField(max_length=95)
    fathername=models.CharField(max_length=95)
    classname=models.IntegerField()
    contact=models.CharField(max_length=15)

```

- 5) run the following command in command prompt (you must be in project folder)

```
python manage.py makemigrations
```

this will prepare all the required sql queries from the model defined.



The screenshot shows a Windows Command Prompt window. The user has navigated to the directory `D:\Youtube\Classes\django\schoolApp` and run the command `python manage.py makemigrations`. The output shows the creation of a migration file `0001_initial.py` for the `admissions` app.

```

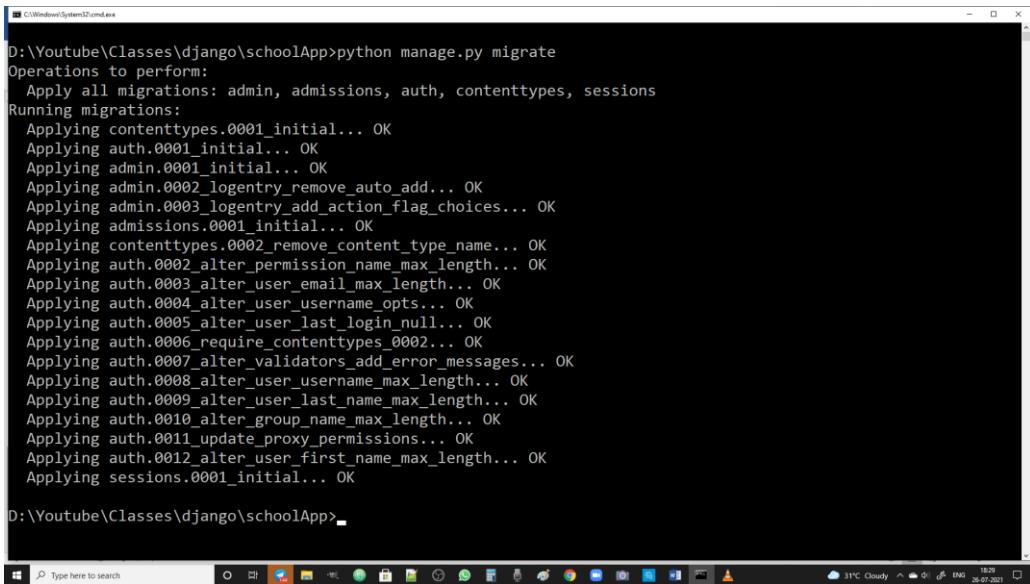
D:\Youtube\Classes\django\schoolApp>python manage.py makemigrations
Migrations for 'admissions':
  admissions\migrations\0001_initial.py
    - Create model Student

D:\Youtube\Classes\django\schoolApp>

```

- 6) run the following command to execute the created queries

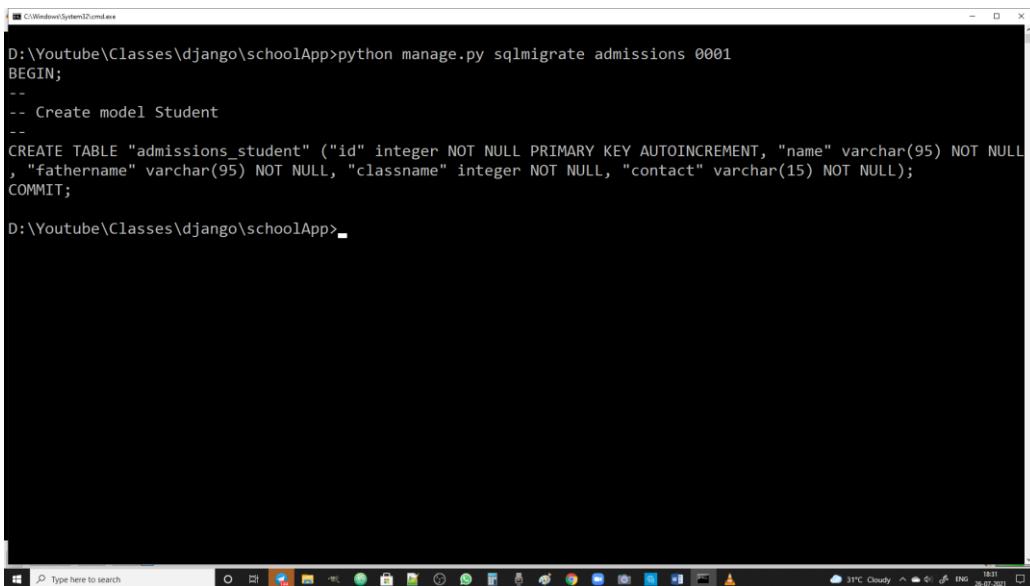
```
python manage.py migrate
```



```
D:\Youtube\Classes\django\schoolApp>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, admissions, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying admissions.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

D:\Youtube\Classes\django\schoolApp>
```

We can check how queries are prepared in the backend by django as follows



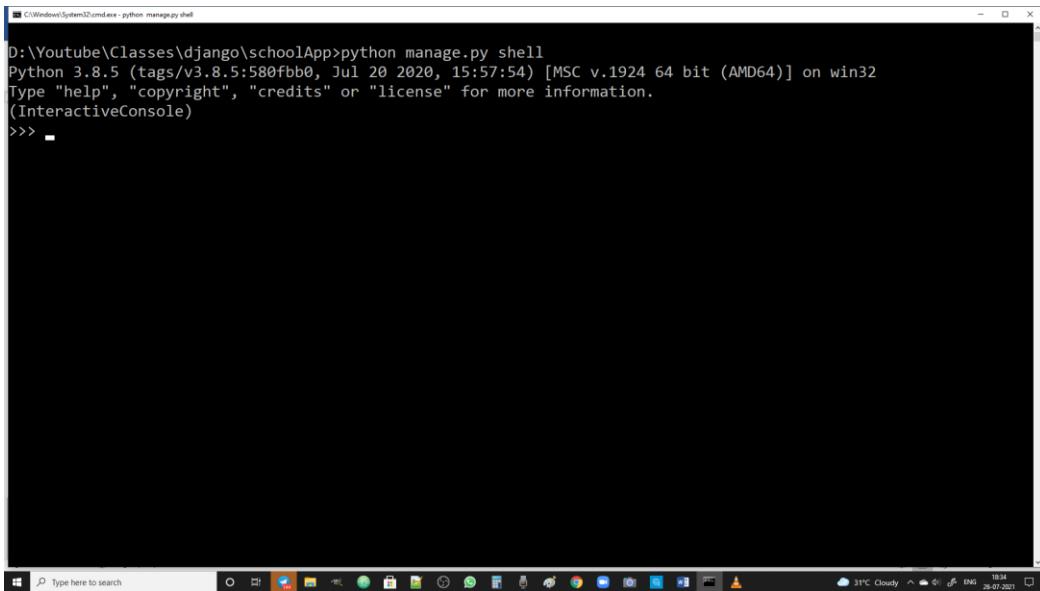
```
D:\Youtube\Classes\django\schoolApp>python manage.py sqlmigrate admissions 0001
BEGIN;
-- 
-- Create model Student
-- 
CREATE TABLE "admissions_student" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(95) NOT NULL
, "fathername" varchar(95) NOT NULL, "classname" integer NOT NULL, "contact" varchar(15) NOT NULL);
COMMIT;

D:\Youtube\Classes\django\schoolApp>
```

Now let us open sqlite and check how database and tables are created.

We can enter into sqlite by using following command

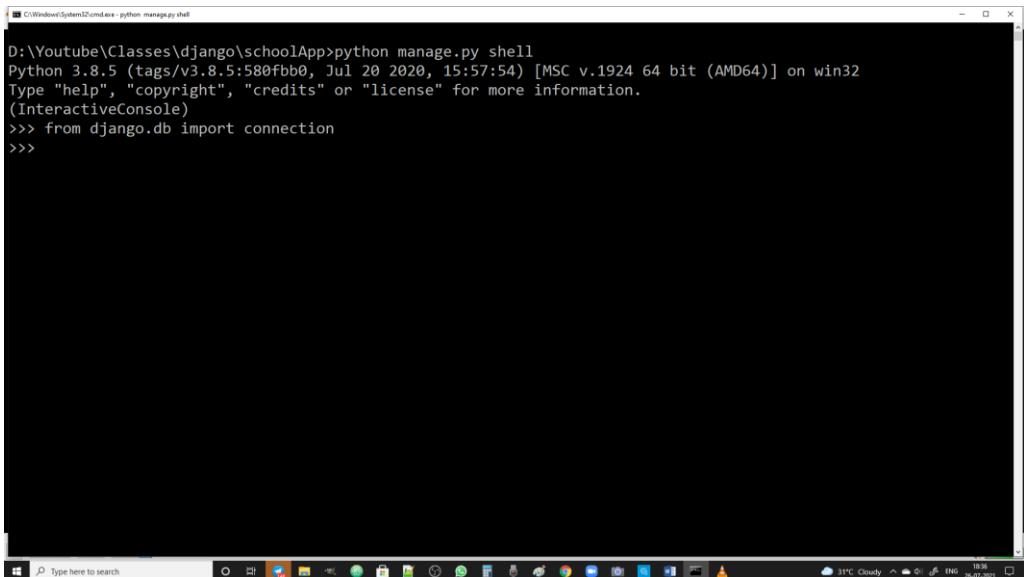
`python manage.py shell`



```
C:\Windows\System32\cmd.exe - python manage.py shell
D:\Youtube\Classes\django\schoolApp>python manage.py shell
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> -
```

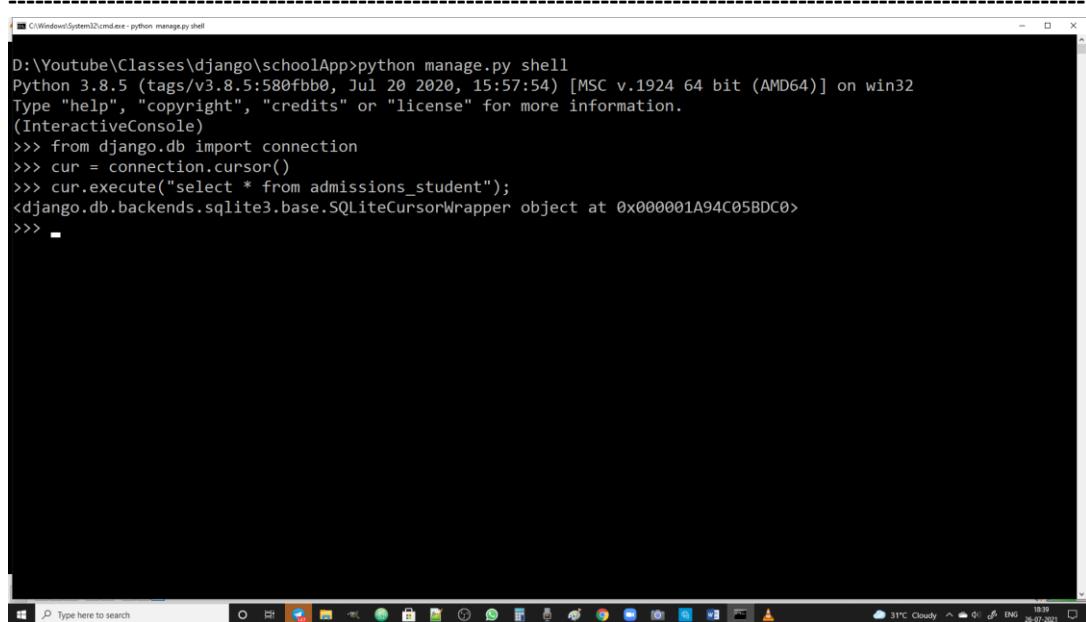
We can create a connection to the database by using following command

As we opened this command from schoolApp and in schoolApp settings we set the database settings for sqlite, so this command will open a connection with sqlite database.



```
C:\Windows\System32\cmd.exe - python manage.py shell
D:\Youtube\Classes\django\schoolApp>python manage.py shell
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.db import connection
>>>
```

Now as we learned in python database connections topic, we need to open cursor, and execute the required query by using execute() method.

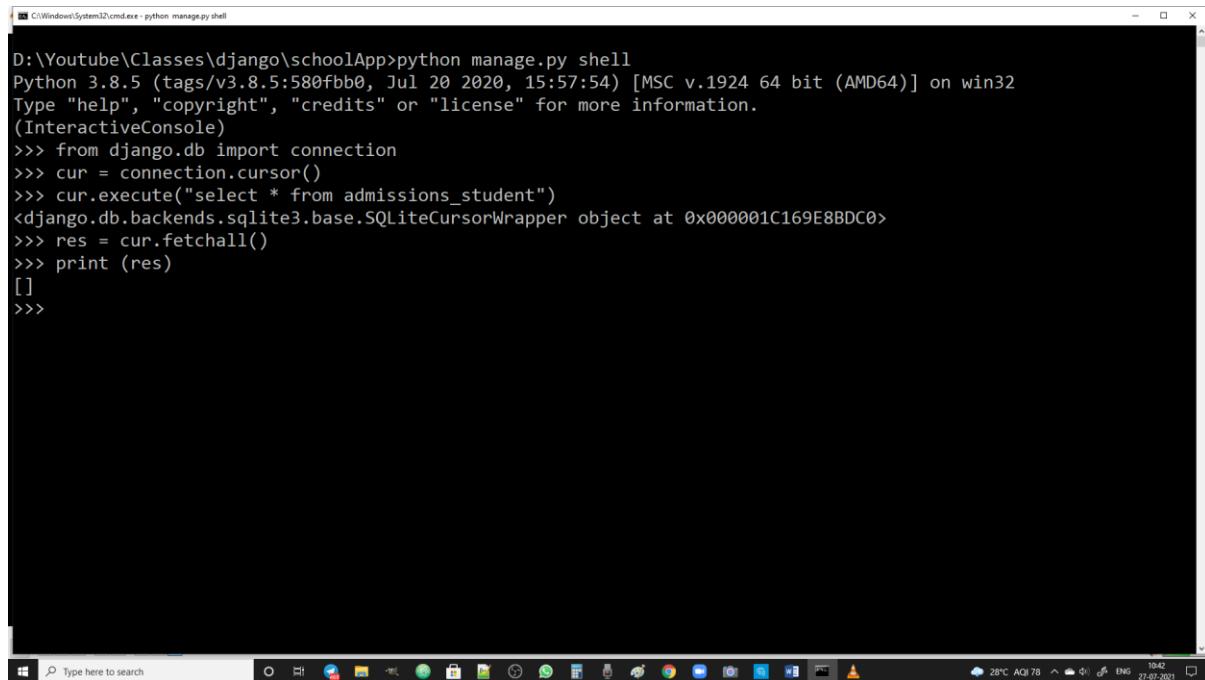


```
D:\Youtube\Classes\django\schoolApp>python manage.py shell
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.db import connection
>>> cur = connection.cursor()
>>> cur.execute("select * from admissions_student");
<django.db.backends.sqlite3.base.SQLiteCursorWrapper object at 0x000001A94C05BDC0>
>>>
```

observe here that django creates tables as applicationname\_modelname

that's why we tried to retrieve the rows from the table admissions\_student

we can see the result by using fetch operations.



```
D:\Youtube\Classes\django\schoolApp>python manage.py shell
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.db import connection
>>> cur = connection.cursor()
>>> cur.execute("select * from admissions_student")
<django.db.backends.sqlite3.base.SQLiteCursorWrapper object at 0x000001C169E8BDC0>
>>> res = cur.fetchall()
>>> print (res)
[]
>>>
```

As students are not yet added, It returned empty list.

## USING MODELS IN VIEWS

Now let us see how to connect our model Student with the view so that we can do database operations and display the results in template.

**Steps:**

**1) import the model Student**

```
from admissions.models import Student
```

**2) get all the records from database using model**

```
result = Student.objects.all()
```

**3) create a dictionary and store above records in this dictionary**

```
students = {'allstudents' : result}
```

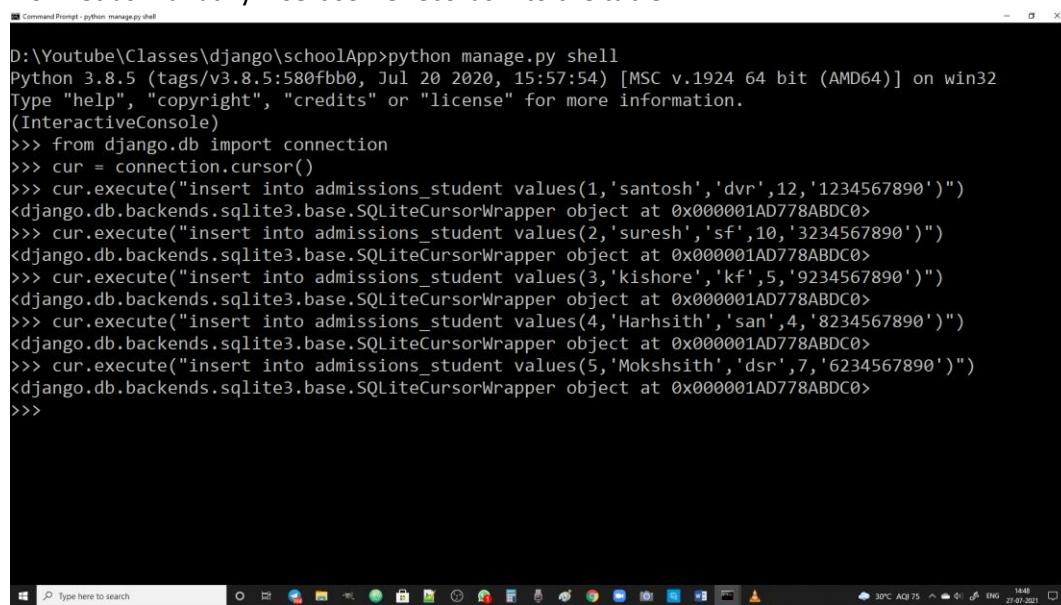
**4) pass the dictionary to template by using render method**

```
return render(request,'admissions/admissions-report.html' , students)
```

**5) Retrieve the values and display them in the template by using a for loop as follows**

```
{% for s in allstudents %}  
    {{s.name}} - {{s.fathername}}  
    <br/>  
{% endfor %}
```

Now let us manually insert some records into the table



```
D:\Youtube\Classes\django\schoolApp>python manage.py shell
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.db import connection
>>> cur = connection.cursor()
>>> cur.execute("insert into admissions_student values(1,'santosh','dvr',12,'1234567890')")
<django.db.backends.sqlite3.base.SQLiteCursorWrapper object at 0x000001AD778ABDC0>
>>> cur.execute("insert into admissions_student values(2,'suresh','sf',10,'3234567890')")
<django.db.backends.sqlite3.base.SQLiteCursorWrapper object at 0x000001AD778ABDC0>
>>> cur.execute("insert into admissions_student values(3,'kishore','kf',5,'9234567890')")
<django.db.backends.sqlite3.base.SQLiteCursorWrapper object at 0x000001AD778ABDC0>
>>> cur.execute("insert into admissions_student values(4,'Harhsith','san',4,'8234567890')")
<django.db.backends.sqlite3.base.SQLiteCursorWrapper object at 0x000001AD778ABDC0>
>>> cur.execute("insert into admissions_student values(5,'Mokshsith','dsr',7,'6234567890')")
<django.db.backends.sqlite3.base.SQLiteCursorWrapper object at 0x000001AD778ABDC0>
>>>
```

full code of views.py and template are as follows

The screenshot shows the Atom code editor interface with the following details:

- Project Structure:** On the left, the project tree shows a directory structure:
  - admission
  - admission/urls.py
  - admission/views.py
  - admission/templates
  - admission/templates/admissions
  - admission/templates/admissions/add-admission.html
  - admission/templates/admissions/admissions-report.html
  - admission/management
  - admission/management/commands
  - admission/management/commands/admissions\_report.py
  - admission/migrations
  - admission/migrations/0001\_initial.py
  - admission/admin.py
  - admission/apps.py
  - admission/models.py
  - admission/tests.py
  - admission/urls.py
  - admission/views.py
- Code Editor:** The main area displays the `views.py` file content:

```
from django.shortcuts import render
from admissions.models import Student

# Create your views here.
#function based views

def homepage(request):
    return render(request,'index.html')

def addAdmission(request):
    values = {"name":"Santosh","age":36,"Address":"Visakhapatnam"}
    return render(request,'admissions/add-admission.html',values)

def admissionsReport(request):
    #get all records from admissions_student table
    result = Student.objects.all() # this is equal to select * from student query
    #Create a dictionary and store above result in it
    students = {'allstudents':result}
    return render(request,'admissions/admissions-report.html',students)
```
- Bottom Status Bar:** Shows file status (1:08), encoding (UTF-8), Python, GitHub, Git status (green), and system info (CPU: 29°C, Light rain).

The screenshot shows a Windows desktop environment with a Python Django project named 'admissions-report' open in a code editor. The project structure is as follows:

- admissions-report.html**: The main template file containing the HTML structure for the admissions report.
- models.py**: The models file defining the database structure.
- views.py**: The views file containing the logic for rendering the template.

The code in **admissions-report.html** is as follows:

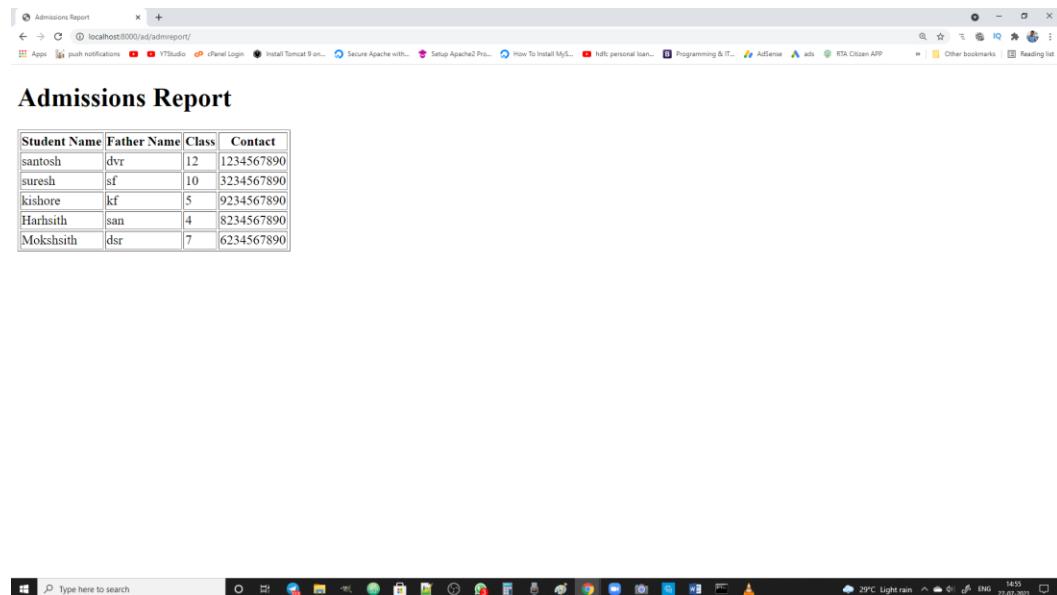
```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title>Admissions Report</title>
    </head>
    <body>
        <h1>Admissions Report</h1>
        <table border="1">
            <tr>
                <th>Student Name</th>
                <th>Father Name</th>
                <th>Class</th>
                <th>Contact</th>
            </tr>
            {% for s in allstudents %}
            <tr>
                <td>{{s.name}}</td>
                <td>{{s.fathername}}</td>
                <td>{{s.classname}}</td>
                <td>{{s.contact}}</td>
            </tr>
            {% endfor %}
        </table>
    </body>
</html>
```

The code editor also shows the file path: **admissions/report.html**.

At the bottom of the screen, the taskbar displays various application icons, and the system tray shows the date and time as **29°C Light rain**.

Now let us start the server and run the program

---



The screenshot shows a web browser window with the title "Admissions Report". The URL in the address bar is "localhost:8000/ad/admireport/". The page content displays a table with the following data:

Student Name	Father Name	Class	Contact
santosh	dvr	12	1234567890
suresh	sf	10	3234567890
kishore	kf	5	9234567890
Hansith	san	4	8234567890
Mokshith	dsr	7	6234567890

The browser's toolbar and taskbar are visible at the bottom.

This is how we can link models with the views and retrieve the records and display those results

## DJANGO – MYSQL CONNECTIVITY

The First step that we need to do in order to implement django-mysql connectivity is setting the mysql values in settings.py

Before setting the values let us create a user in mysql with mysql\_native\_password option as follows.

```
mysql> create database schoolapp;
Query OK, 1 row affected (0.01 sec)

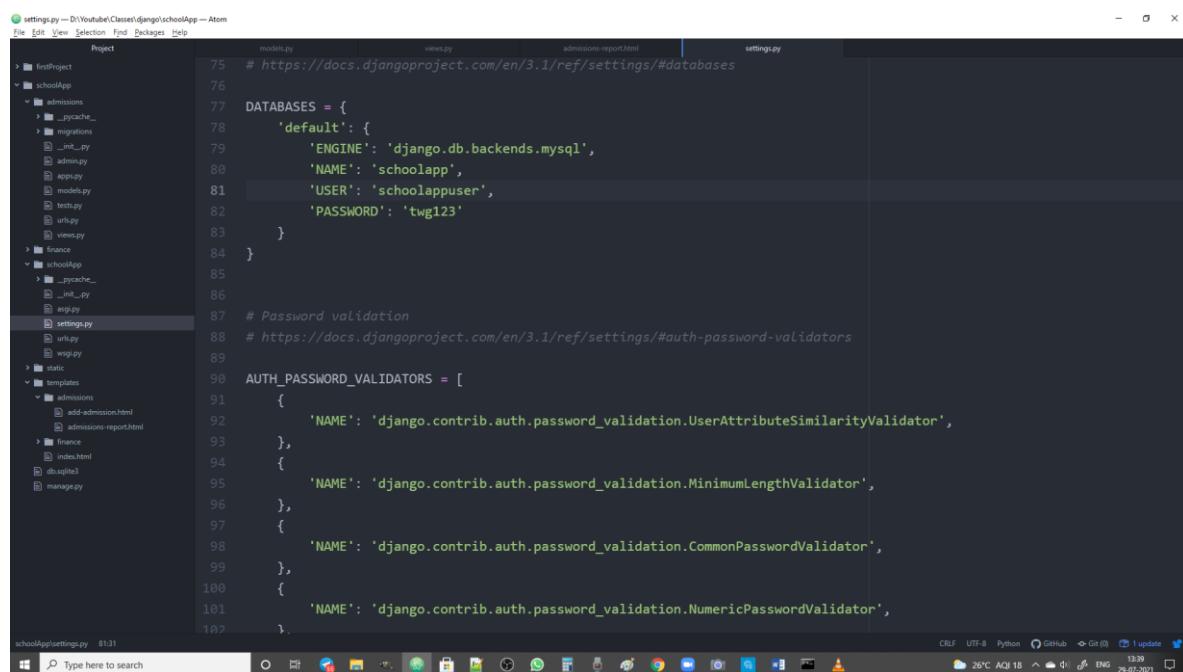
mysql> create user 'schoolappuser'@'localhost' identified with mysql_native_password by 'twg123';
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Now grant the permission to the user schoolappuser to access schoolapp database.

```
mysql> grant all on schoolapp.* to 'schoolappuser'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

Now let's set the database setting in settings.py



The screenshot shows the Atom code editor with the settings.py file open. The code defines the DATABASES section with the following details:

```
75 # https://docs.djangoproject.com/en/3.1/ref/settings/#databases
76
77 DATABASES = {
78     'default': {
79         'ENGINE': 'django.db.backends.mysql',
80         'NAME': 'schoolapp',
81         'USER': 'schoolappuser',
82         'PASSWORD': 'twg123'
83     }
84 }
85
86
87 # Password validation
88 # https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validation
89
90 AUTH_PASSWORD_VALIDATORS = [
91     {
92         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
93     },
94     {
95         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
96     },
97     {
98         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
99     },
100    {
101        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
102    }
103]
```

Now let us run makemigrations and migrate commands to create the tables in mysql instead of sqlite.

```
python manage.py makemigrations
```

```
python manage.py migrate
```

```
C:\Windows\System32\cmd.exe - mysql -u root -p

mysql> use schoolapp;
Database changed
mysql> show tables;
+-----+
| Tables_in_schoolapp |
+-----+
| admissions_student |
| admissions_teacher |
| auth_group |
| auth_group_permissions |
| auth_permission |
| auth_user |
| auth_user_groups |
| auth_user_user_permissions |
| django_admin_log |
| django_content_type |
| django_migrations |
| django_session |
+-----+
12 rows in set (0.00 sec)

mysql> desc admissions_student;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id | int | NO | PRI | NULL | auto_increment |
| name | varchar(1000) | NO | | NULL |
| fathername | varchar(1000) | NO | | NULL |
| classname | int | NO | | NULL |
| contact | varchar(1000) | NO | | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)


```

Now let us insert some student data into the above table manually

```
C:\Windows\System32\cmd.exe - mysql -u root -p

mysql> desc admissions_student;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id | int | NO | PRI | NULL | auto_increment |
| name | varchar(1000) | NO | | NULL |
| fathername | varchar(1000) | NO | | NULL |
| classname | int | NO | | NULL |
| contact | varchar(1000) | NO | | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0.04 sec)

mysql> insert into admissions_student values(1,'Santosh','dvr',4,'9999999999');
Query OK, 1 row affected (0.01 sec)

mysql> insert into admissions_student values(2,'Harshith','dsr',3,'8888899999');
Query OK, 1 row affected (0.01 sec)

mysql> insert into admissions_student values(3,'Mokshith','dsr',1,'7777799999');
Query OK, 1 row affected (0.01 sec)

mysql> select * from admissions_student;
+-----+-----+-----+-----+
| id | name | fathername | classname | contact |
+-----+-----+-----+-----+
| 1 | Santosh | dvr | 4 | 9999999999 |
| 2 | Harshith | dsr | 3 | 8888899999 |
| 3 | Mokshith | dsr | 1 | 7777799999 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)


```

Now let us run the program and check whether our django application can retrieve the data from the mysql table `admissions_student` or not



## Admissions Report

Student Name	Father Name	Class	Contact
Santosh	dvr	4	9999999999
Harshith	dsr	3	8888899999
Mokshith	dsr	1	7777799999



## DJANGO – ADMIN USER INTERFACE

Django provides Administrator user interface for projects automatically.

In urls.py one url mapping is by default provided by django i.e., admin/

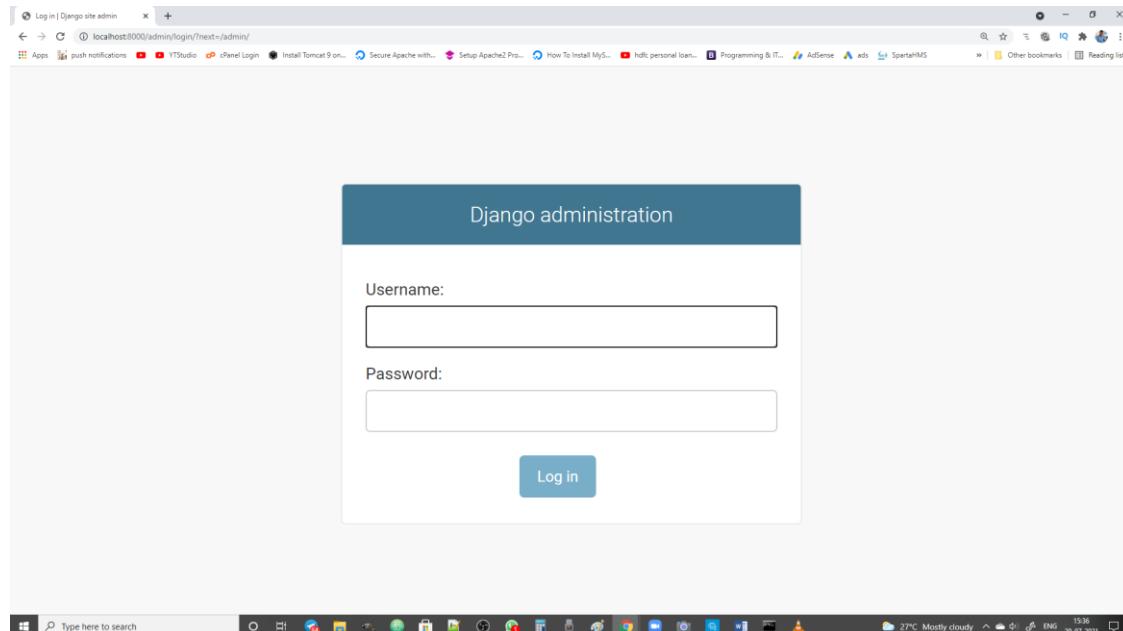


The screenshot shows the Atom code editor with the following details:

- Project:** schoolApp
- File:** urls.py
- Content:** The code defines the URL patterns for the schoolApp project. It includes imports for django.contrib.admin, django.urls, and django.conf.urls, as well as local imports for views, admissions, finance, and admissions-report.html. The urlpatterns list contains paths for the admin interface, a homepage, admissions, financial reports, and class-based views for admissions and finance.

```
2
3     The `urlpatterns` list routes URLs to views. For more information please see:
4         https://docs.djangoproject.com/en/3.1/topics/http/urls/
5     Examples:
6         Function views
7             1. Add an import: from my_app import views
8             2. Add a URL to urlpatterns: path('', views.home, name='home')
9     Class-based views
10        1. Add an import: from other_app.views import Home
11        2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13     1. Import the include() function: from django.urls import include, path
14     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18 from django.conf.urls import include
19 from admissions import views
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', views.homepage),
24
25     path('ad/', include('admissions.urls')),
26     path('fin/', include('finance.urls')),
27 ]
28
```

If we run this url we will get administrator login page automatically.



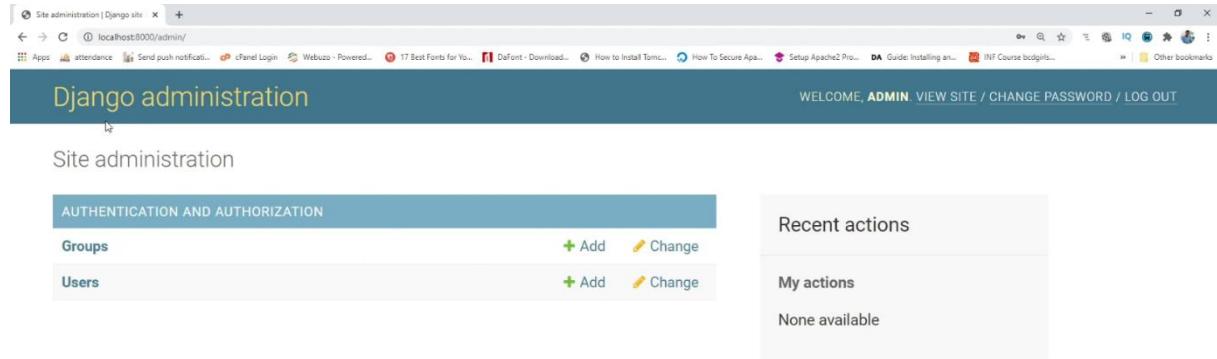
To login into this, first we need to create a super user so that we can use those credentials.

```
C:\WINDOWS\system32\cmd.exe - python manage.py createsuperuser

D:\Youtube\Classes\django\schoolApp>python manage.py createsuperuser
Username (leave blank to use 'santosh'): admin
Email address: teluguwebguru@gmail.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]:
```

Now we successfully created a superuser with the login details admin,admin

Now let us try to enter into the admin ui by using these credentials



Observe how the admin user interface is automatically created by django.

Now let us add Student model to the admin UI

open admin.py in admission application

The screenshot shows the Atom IDE interface with a Django project structure on the left and a code editor on the right.

**Project Tree:**

- Project
- firstProject
- schoolApp
  - admissions
    - \_\_pycache\_\_
    - migrations
      - \_\_init\_\_.py
    - admin.py
    - apps.py
    - models.py
    - tests.py
    - urls.py
    - views.py
  - finance
  - schoolApp
    - \_\_pycache\_\_
      - \_\_init\_\_.py
      - asgi.py
      - settings.py
      - urls.py
      - wsgi.py
    - static
    - templates
      - admissions
        - add-admission.html
        - admissions-report.html
    - finance
      - index.html
  - db.sqlite3
  - manage.py

**Code Editor (admin.py):**

```
from django.contrib import admin

# Register your models here.
```

**System Tray:**

- CR LF
- UTF-8
- Python
- Github
- Git (S)
- Update
- Cloud
- 27°C AQI 18
- 16:20
- 29-07-2021

The screenshot shows the PyCharm IDE interface with a Django project named 'schoolApp'. The left sidebar displays the project structure, including 'admissions' and 'schoolApp' packages, and various files like 'admin.py', 'models.py', and 'views.py'. The right pane shows the code for 'admissions/admin.py', which contains the following Python code:

```
from django.contrib import admin
from admissions.models import Student

# Register your models here.
admin.site.register(Student)
```

As shown above, We need to first import the model and register it into admin ui using

```
admin.site.register(modelname)
```

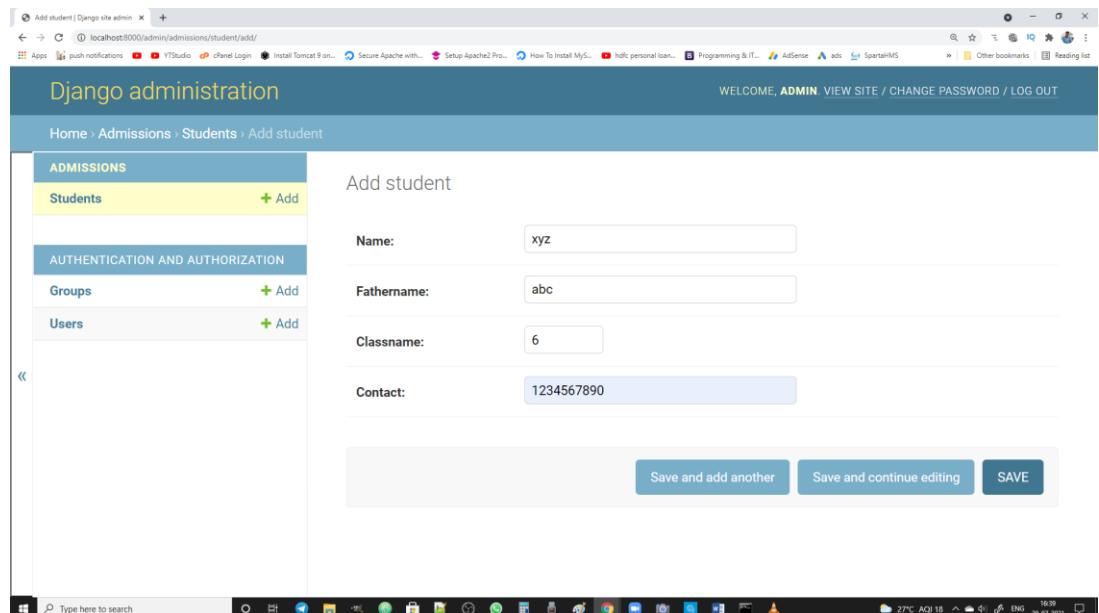
After this process, if we run our admin page again we can see the Student model in admin User interface.

The screenshot shows the Django administration interface. On the left, there's a sidebar with sections for 'ADMISSIONS' (Students) and 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users). On the right, there are two boxes: 'Recent actions' (empty) and 'My actions' (also empty). At the top right, it says 'WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT'.

Observe, the registered model is displayed in admin ui interface

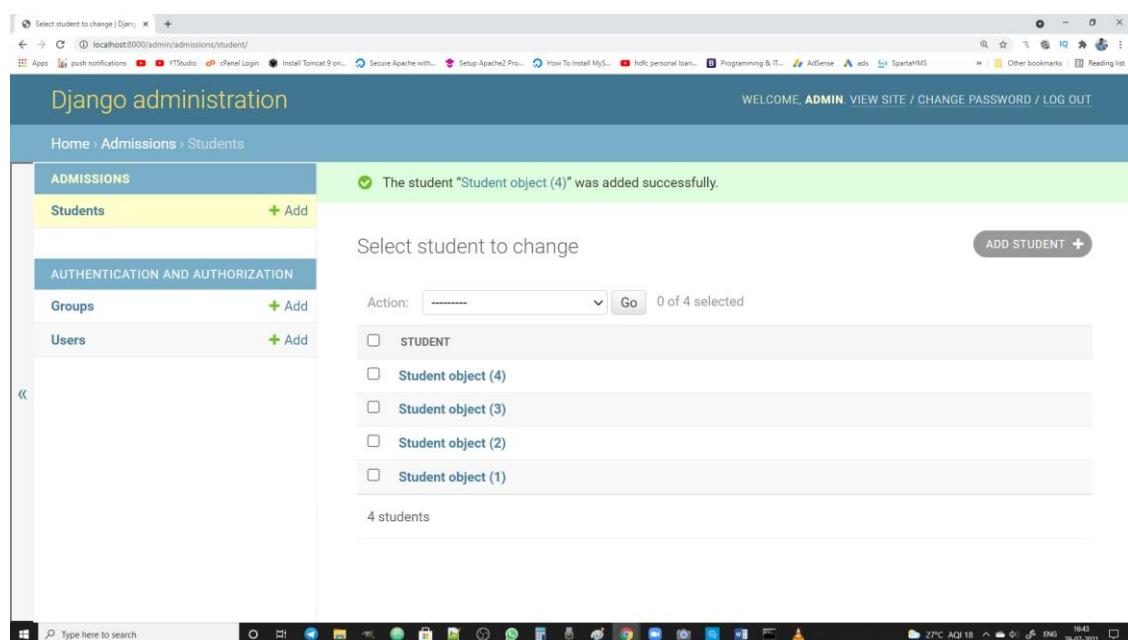
If we click on add button it will automatically creates a form and display it to enter new students data.

The screenshot shows the 'Add student' form in the Django administration interface. The sidebar on the left shows the 'Students' model selected under 'ADMISSIONS'. The main area has a form with fields: 'Name' (with an input field), 'Fathername' (with an input field), 'Classname' (with an input field), and 'Contact' (with an input field). Below the form are three buttons: 'Save and add another', 'Save and continue editing', and a large blue 'SAVE' button. The status bar at the bottom shows system information like temperature, battery level, and network status.



The screenshot shows the Django Admin interface for adding a student. The left sidebar has 'ADMISSIONS' and 'Students' selected. The main form has fields for Name, Fathername, Classname, and Contact. At the bottom are three buttons: 'Save and add another', 'Save and continue editing', and a larger blue 'SAVE' button.

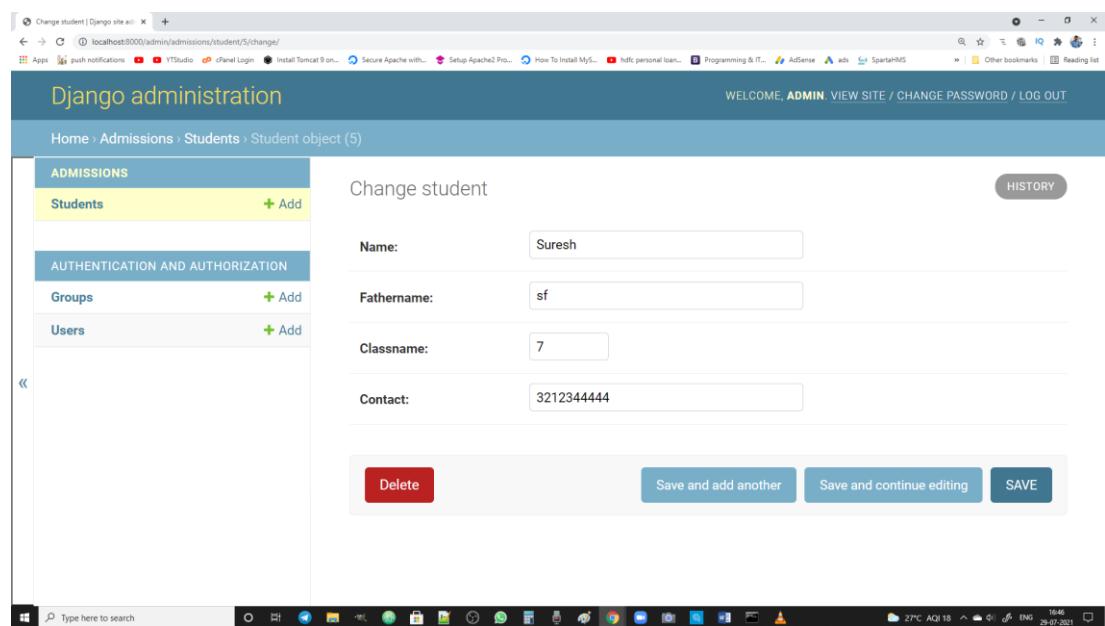
Once we fill data and click on save, it will automatically add it to the database



The screenshot shows the Django Admin interface after saving a new student. A green success message says 'The student "Student object (4)" was added successfully.' To the right, there's a list titled 'Select student to change' with four entries: 'STUDENT', 'Student object (4)', 'Student object (3)', 'Student object (2)', and 'Student object (1)'. At the top right is a blue 'ADD STUDENT +' button.

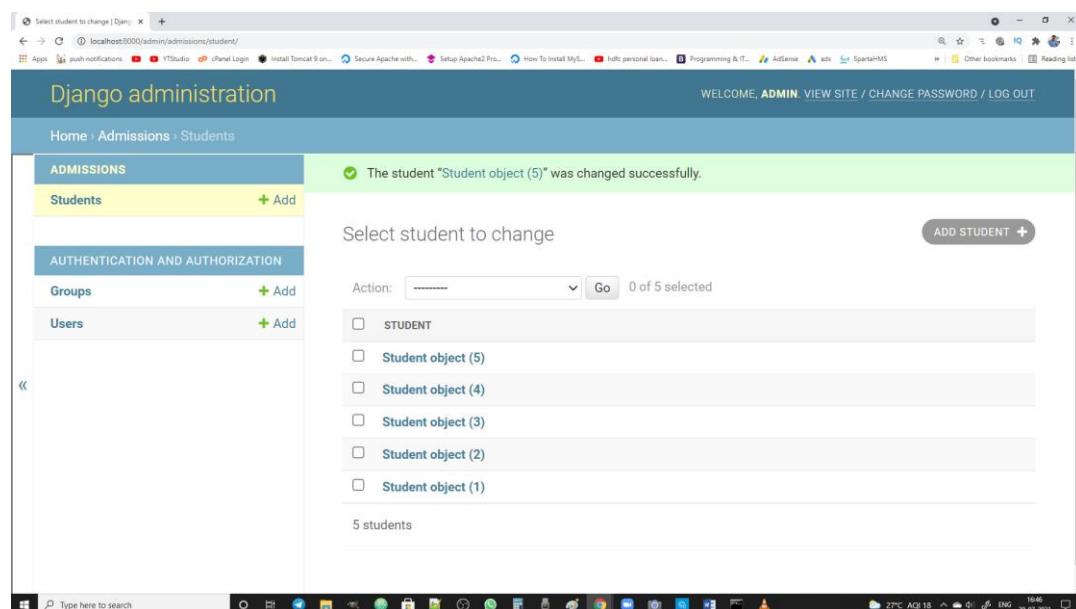
By default, it will display each student as an object. We added three students in above topic and now we added the forth student xyz, so it is showing 4 students.

Let us try to add one more student now.



The screenshot shows the Django admin interface for a 'Student' object. The left sidebar has 'ADMISSIONS' and 'AUTHENTICATION AND AUTHORIZATION' sections. The 'Students' section is selected. The main area is titled 'Change student' and contains fields for Name (Suresh), Fathername (sf), Classname (7), and Contact (3212344444). At the bottom are four buttons: 'Delete', 'Save and add another', 'Save and continue editing', and a large 'SAVE' button.

after click on save, it will now show 5 student objects



The screenshot shows the Django admin interface after saving a student. A green success message says 'The student "Student object (5)" was changed successfully.' Below this, a list of students is displayed with checkboxes. The list includes 'STUDENT' and five individual student objects: 'Student object (5)', 'Student object (4)', 'Student object (3)', 'Student object (2)', and 'Student object (1)'. There are also buttons for 'Select student to change' and 'ADD STUDENT'.

If we want to display these student objects as tables then we need to define an extra class in admin.py file (instead of registering the model directly).

```
from django.contrib import admin
from admissions.models import Student

class StudentAdmin(admin.ModelAdmin):
    list_display=['name','fathername']

# Register your models here.
admin.site.register(Student,StudentAdmin)
```

Instead of directly registering our model with the admin, we need to create a class that inherits from ModelAdmin and we need to specify all the list of columns to be displayed in the admin ui by using list\_display as shown above.

Now let us run the admin ui and check whether displaying student objects are replaced with these columns.

Select student to change | Djenv: +

localhost:8000/admin/admissions/student/

Apps push notifications V7audio chen Login Install Tomcat 9 on... Secure Apache with... Setup Apache2 Pro... How To Install MySQL personal loan... Programming & IT... AdSense Ads SportsHMS Other bookmarks Reading list

## Django administration

WELCOME, ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Admissions › Students

**ADMISSIONS**

**Students** [+ Add](#)

**AUTHENTICATION AND AUTHORIZATION**

**Groups** [+ Add](#)

**Users** [+ Add](#)

«

Select student to change

Action:  0 of 5 selected

<input type="checkbox"/>	NAME	FATHERNAME	CLASSNAME	CONTACT
<input type="checkbox"/>	Suresh	sf	7	3212344444
<input type="checkbox"/>	xyz	abc	6	1234567890
<input type="checkbox"/>	Mokshith	dsr	1	7777799999
<input type="checkbox"/>	Harshith	dsr	3	8888899999
<input type="checkbox"/>	Santosh	dvr	4	9999999999

5 students

[ADD STUDENT +](#)

This is how django automatically provides a very beautiful admin interface.

## DJANGO – ORM

The following are the various ways to implement various SQL queries through Django ORM concept. Following tables shows what's the query we want to write, How to implement it in django ORM way.

Selecting all columns

<code>select * from admissions_student</code>	<pre>qs = Student.objects.all()     print (qs.query)     for s in qs:         print (s.name)</pre>
<code>select * from admissions_student where id=2</code>	<pre>s = Student.objects.get(id=2)     print (s.name)</pre>

Selective columns(values\_list or values or only)

<code>select id,firstname from admissions_student</code>	<pre>students = Student.objects.all().values_list('id', 'firstname')</pre>
--	--

filtering using (gt,gte,lt,lte,contains,icontains,in,startswith,endswith)

<code>select * from admissions_student where id&gt;2</code>	<pre>students = Student.objects.filter(id__gt=2)</pre>
---	--

Logical Operators (And, Or, Not)

<code>select * from admissions_student where id&gt;2 or name LIKE '%sa%'</code>	<pre>students = Student.objects.filter(id__gt=2)   Student.objects.filter(name__contains='sa')</pre>
<code>select * from admissions_student where id&gt;2 and id&lt;5</code>	<pre>students = Student.objects.filter(id__gt=2) &amp; Student.objects.filter(id__lt=5)</pre>
<code>select * from admissions_student where NOT(id&gt;5)</code>	<pre>students = Student.objects.exclude(id__gt=5)</pre>

### Aggregate Functions (Avg, Sum, Max, Min, Count)

```
select id,firstname from admissions_student
```

```
from django.db.models import Avg,Sum,Max,Min,Count
students = Student.objects.all().aggregate(Avg('id'))
```

### inserting data into the database (create)

```
insert into admissions_student
values(6,'abc','xyz',8,'5648562233')
```

```
1) s = Student(name='',  
fathername='',classname='',contact='')

s.save()

2) Student.objects.create(name='',  
fathername='',classname='',contact='')
```

### Bulk creating/inserting (bulk\_create)

```
insert into admissions_student
values(6,'abc','xyz',8,'5648562233')
```

```
Student.objects.bulk_create([
    Student(name='', fathername='', classname='', contact=''),
    Student(name='', fathername='', classname='', contact=''),
    Student(name='', fathername='', classname='', contact=''),
])
```

### Delete from tables

```
Delete from admissions_student where id=2
```

```
s = Student.objects.get(id=1)

s.delete()

qs = Student.objects.filter(id__gt=5)

qs.delete()
```

### Updating tables

```
Update admissions_student set name='' where
id=2
```

```
s = Student.objects.get(id=1)

s.firstname=''

s.save()
```

Sorting data in tables

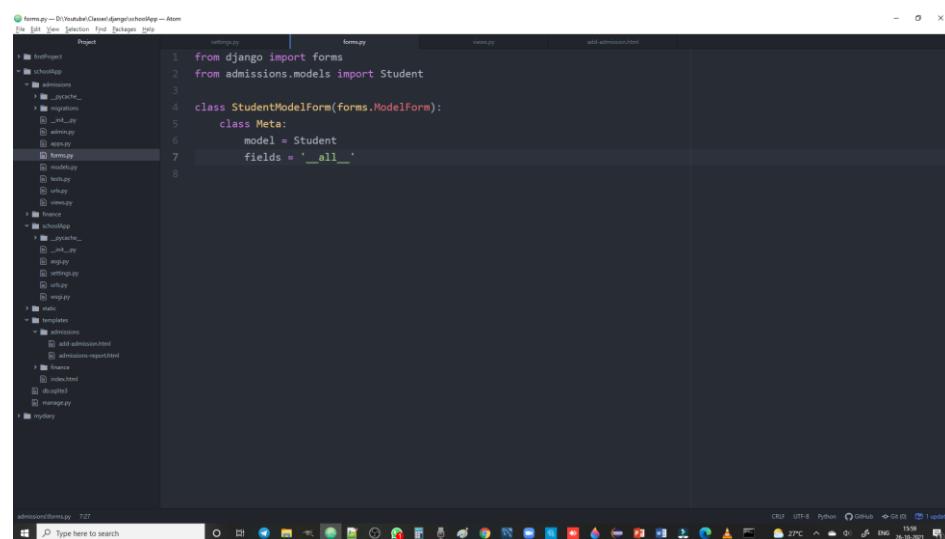
Select * from student order by firstname	s = Student.objects.all().order_by('firstname')
	s = Student.objects.all().order_by('-firstname') => reverse

## DJANGO – MODEL FORMS

**To create model forms**

- 1) create a file forms.py in your application folder
- 2) Create a class that is inherited from forms.ModelForm
- 3) Create a class Meta

```
class classname(forms.ModelForm):
    class Meta:
        model = modelname
        fields = '__all__' (to import selected fields mention them as list)
```



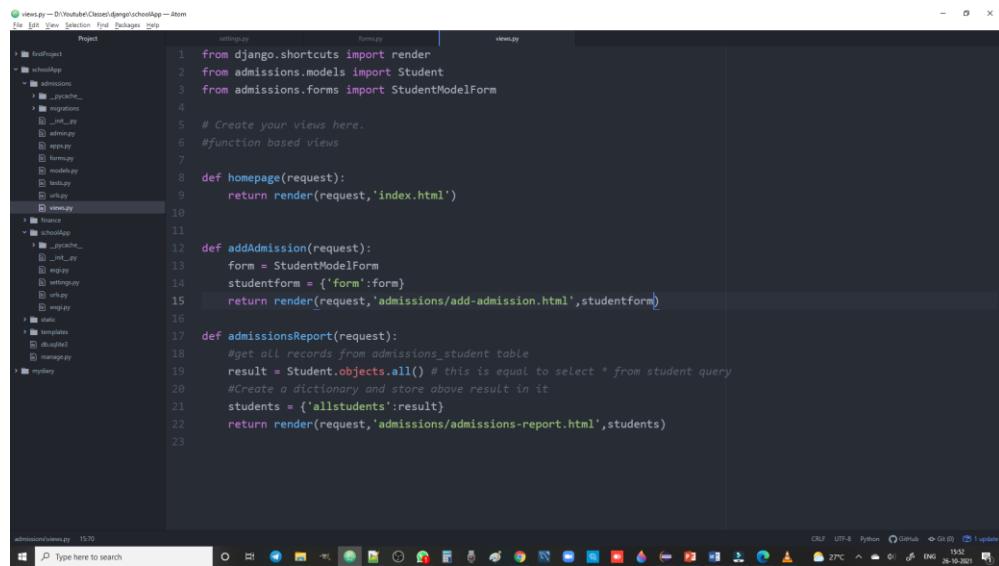
The screenshot shows the Atom code editor interface. The left sidebar displays a project structure for a Django application named 'firstProject'. Inside 'firstProject', there are several apps: 'admissions', 'schoolapp', 'finance', 'static', and 'mystry'. The 'admissions' app contains files like 'admin.py', 'models.py', 'tests.py', and 'views.py'. The 'forms.py' file is open in the main editor area. The code defines a ModelForm named 'StudentModelForm' that inherits from 'forms.ModelForm'. It specifies the 'model' as 'Student' and the 'fields' as '\_\_all\_\_'. The status bar at the bottom shows the file path 'forms.py -- D:\Youtube\Classes\djangop\r\schoolapp\admissions', the current line '7/27', and various system icons.

```
from django import forms
from admissions.models import Student

class StudentModelForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = '__all__'
```

- 4) In your view get the model form and send as dictionary to template (import model, model form first)

```
form = modelformname() ---- creating object of modelform
return render (request, template name , dictionary)
```



views.py — D:\Youtube\Classes\django\schoolApp — Atom

```

Project settings.py forms.py views.py
1 from django.shortcuts import render
2 from admissions.models import Student
3 from admissions.forms import StudentModelForm
4
5 # Create your views here.
6 #function based views
7
8 def homepage(request):
9     return render(request,'index.html')
10
11
12 def addAdmission(request):
13     form = StudentModelForm()
14     studentform = {'form':form}
15     return render(request,'admissions/add-admission.html',studentform)
16
17 def admissionsReport(request):
18     #get all records from admissions_student table
19     result = Student.objects.all() # this is equal to select * from student query
20     #create a dictionary and store above result in it
21     students = {'allstudents':result}
22     return render(request,'admissions/admissions-report.html',students)
23

```

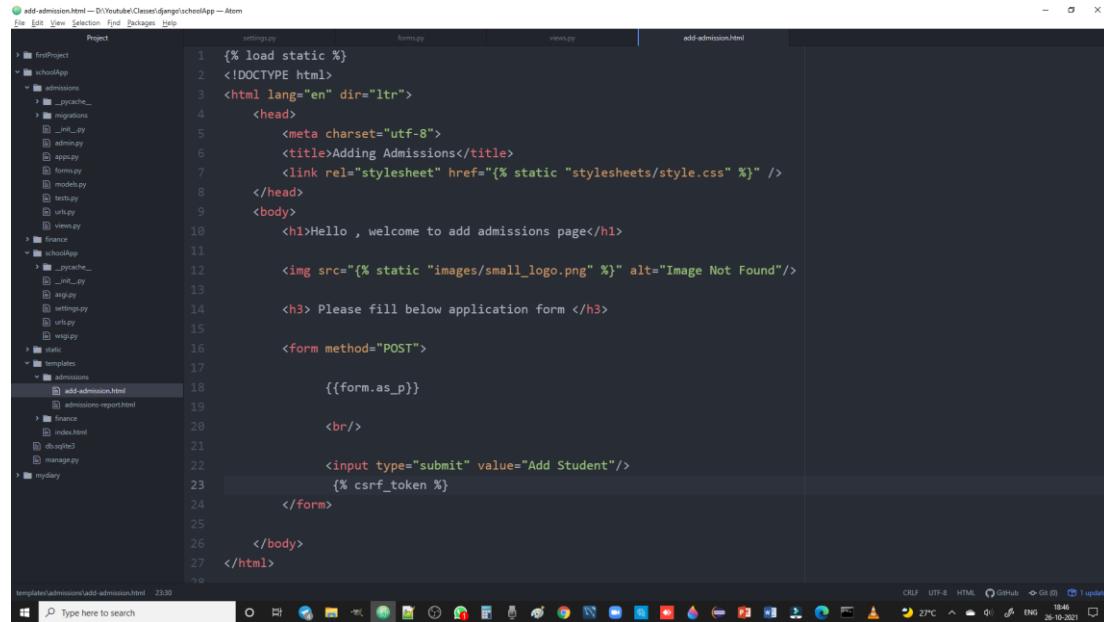
administration.py 15:20

## 5) display the form in template

```

<form method="POST">
    {{form.as_p}}
    {% csrf_token %}
</form>

```



add-admission.html — D:\Youtube\Classes\django\schoolApp — Atom

```

Project settings.py forms.py views.py add-admission.html
1 {% load static %}
2 <!DOCTYPE html>
3 <html lang="en" dir="ltr">
4     <head>
5         <meta charset="utf-8">
6         <title>Adding Admissions</title>
7         <link rel="stylesheet" href="{% static "stylesheets/style.css" %}" />
8     </head>
9     <body>
10        <h1>Hello , welcome to add admissions page</h1>
11
12        
13
14        <h3> Please fill below application form </h3>
15
16        <form method="POST">
17            {{form.as_p}}
18
19            <br/>
20
21            <input type="submit" value="Add Student"/>
22            {% csrf_token %}
23
24        </form>
25
26
27    </body>
28 </html>

```

templates/admissions/add-admission.html 23:38



Hello , welcome to add admissions page

Please fill below application form

Name:

Fathername:

Classname:

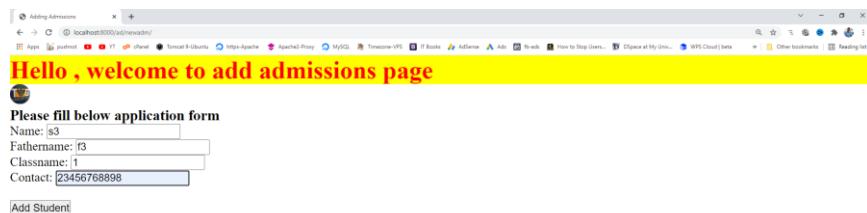
Contact:



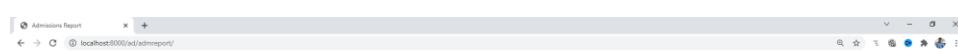
## SAVING FORM DATA IN DJANGO

To get input from model forms and save in database we need to get the form data and check whether it is valid or not if valid we can save by using save method as follows.

```
form = modelformname() ---- creating object of modelform
if form.method=='POST':
    form = modelformname(request.POST)
    if form.is_valid():
        form.save()
    return indexpageviewname(request)
return render (request, template name , dictionary)
```



whenever we clicked on the button data will be saved into the database. if we check the admissions report then we can see this entry in the table.

**Admissions Report**

Student Name	Father Name	Class	Contact
Santosh	dvr	4	9999999999
Harshith	dsr	3	8888899999
Mokshith	dsr	1	7777799999
xyz	abc	6	1234567890
Suresh	sf	7	3212344444
s3	f3	1	23456768898

## DJANGO FORMS

**without using model & model form we can create the forms using django forms.**

**To create forms**

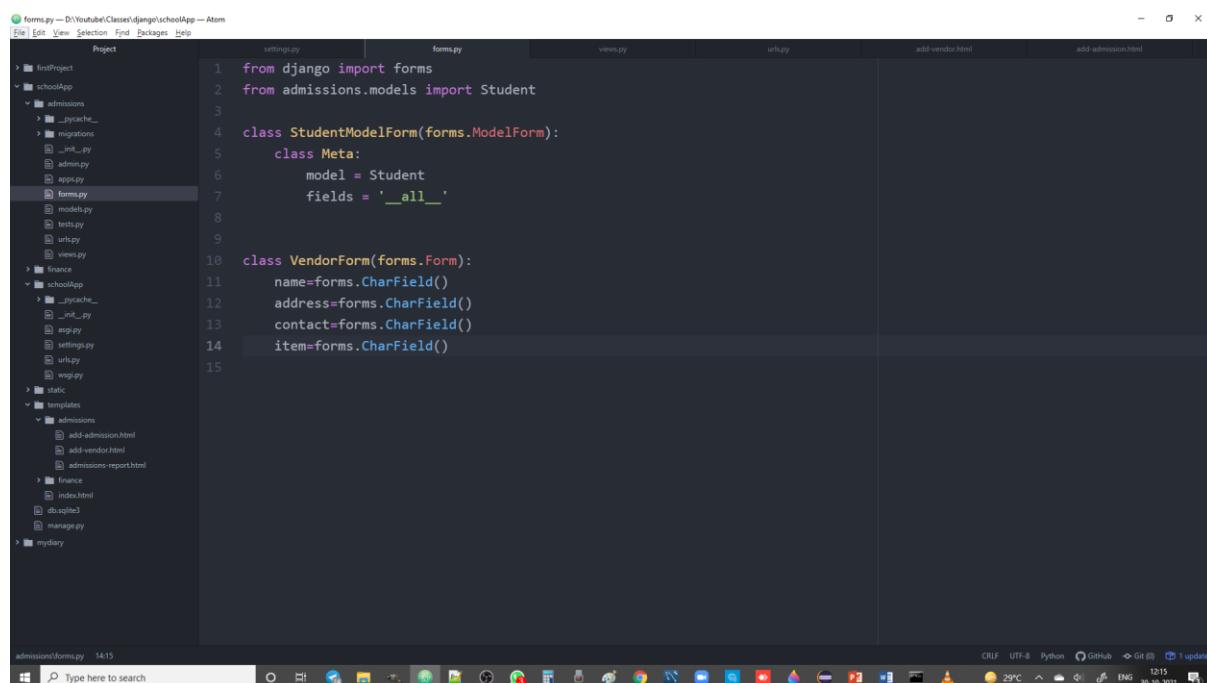
- 1) create a file forms.py in your application folder
- 2) Create a class that is inherited from forms.Form
- 3) define fields to be included in the form

```
class VendorForm(forms.Form):
    name=form.CharField()
    address=form.CharField()
    contact=form.CharField()
    item=form.CharField()
```

- 4) Use Form in view

- 5) display the form in template

```
<form method="POST">
    {{form.as_p}}
    {% csrf_token %}
</form>
```



```
from django import forms
from admissions.models import Student

class StudentModelForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = '__all__'

class VendorForm(forms.Form):
    name=forms.CharField()
    address=forms.CharField()
    contact=forms.CharField()
    item=forms.CharField()
```



The screenshot shows the PyCharm IDE interface with the file `views.py` open. The code implements three views: `addAdmission`, `addVendor`, and `admissionsReport`. The `addAdmission` view handles POST requests to save student form data. The `addVendor` view handles POST requests to save vendor form data, printing the cleaned data for name, address, contact, and item. The `admissionsReport` view retrieves all records from the `admissions_student` table.

```
views.py -- D:\YouTube\Classes\django\schoolApp -- Atom
File Edit View Selection Find Packages Help
Project settings.py forms.py views.py urls.py add-vendor.html add-admission.html
> firstProject
  schoolApp
    admissions
      __pycache__
      migrations
        __init__.py
        admin.py
        apps.py
        forms.py
        models.py
        tests.py
        urls.py
        views.py
    finance
    schoolApp
      __pycache__
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
  static
  templates
    admissions
      add-admission.html
      add-vendor.html
      admissions-report.html
  finance
    index.html
  db.sqlite3
  manage.py
> mydir
admissions/views.py 33:43

12
13     def addAdmission(request):
14         form = StudentModelForm
15         studentform = {'form':form}
16         if request.method=='POST':
17             form = StudentModelForm(request.POST)
18             if form.is_valid():
19                 form.save()
20
21             return render(request,'admissions/add-admission.html',studentform)

22
23     def addVendor(request):
24         form = VendorForm
25         vform = { 'form':form}
26         if request.method=='POST':
27             form = VendorForm(request.POST)
28             if form.is_valid():
29                 print (form.cleaned_data['name'])
30                 print (form.cleaned_data['address'])
31                 print (form.cleaned_data['contact'])
32                 print (form.cleaned_data['item'])
33
34             return render(request,'admissions/add-vendor.html',vform)

35
36
37     def admissionsReport(request):
38         #get all records from admissions_student table
39         result = Student.objects.all() # here we can use filter * From admission records
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
217
218
219
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
247
248
249
249
250
251
252
253
254
255
256
257
257
258
259
259
260
261
262
263
264
265
266
267
267
268
269
269
270
271
272
273
274
275
276
277
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
319
320
321
322
323
324
325
326
327
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
367
368
369
369
370
371
372
373
374
375
376
377
377
378
379
379
380
381
382
383
384
385
386
387
387
388
389
389
390
391
392
393
394
395
396
397
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
414
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
154
```

add-vendor.html — D:\VSCode\Classes\django\schoolApp — Atom

File Edit View Selection Find Packages Help

Project

settings.py forms.py views.py urls.py add-vendor.html add-admission.html

```
1  {% load static %} 
2  <!DOCTYPE html> 
3  <html lang="en" dir="ltr"> 
4      <head> 
5          <meta charset="utf-8"> 
6          <title>Adding vendor</title> 
7          <link rel="stylesheet" href="{% static "stylesheets/style.css" %}" /> 
8      </head> 
9      <body> 
10         <h1>Hello , welcome to add vendor page</h1> 
11          
12         <h3> Please fill below application form </h3> 
13         <form method="POST"> 
14             <table> 
15                 {{form.as_table}} 
16             </table> 
17             <br/> 
18             <input type="submit" value="Add Vendor"/> 
19             {% csrf_token %} 
20         </form> 
21         </body> 
22     </html> 
23 
```



Adding vendor × +

localhost:8000/ad/newvendor/

Apps pushnot YT cPanel Tomcat 9-Ubuntu https-Apache Apache2-Proxy MySQL Timezone-VPS IT Books AdSense Ads fb-ads How To Stop Users... DSpace at My Univ... How To Buy Google... Other bookmarks Reading list

## Hello , welcome to add vendor page

Please fill below application form

Name:

Address:

Contact:

Item:

Add Vendor

Adding vendor × +

localhost:8000/ad/newvendor/

Apps pushnot YT cPanel Tomcat 9-Ubuntu https-Apache Apache2-Proxy MySQL Timezone-VPS IT Books AdSense Ads fb-ads How To Stop Users... DSpace at My Univ... How To Buy Google... Other bookmarks Reading list

## Hello , welcome to add vendor page

Please fill below application form

Name: telugu web guru

Address: visakhapatnam

Contact: teluguwebguru@gmail.com

Item: Online Courses

Add Vendor





```
C:\Windows\System32\cmd.exe - python manage.py runserver
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[30/Oct/2021 12:18:34] "GET /ad/newvendor/ HTTP/1.1" 200 1190
[30/Oct/2021 12:19:37] "GET /ad/newvendor/ HTTP/1.1" 200 1189
D:\Youtube\Classes\django\schoolApp\admissions\views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 30, 2021 - 17:04:51
Django version 3.1, using settings 'schoolApp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
telugu web guru
visakhapatnam
teluguwebguru@gmail.com
Online Courses
[30/Oct/2021 17:05:14] "POST /ad/newvendor/ HTTP/1.1" 200 1189
Not Found: /favicon.ico
[30/Oct/2021 17:05:14] "GET /favicon.ico HTTP/1.1" 404 2282
```

observe that data entered in form is displayed here

## CRUD OPERATIONS USING DJANGO

CRUD operations are Create, Read, Update, delete which we generally performed on data bases.

Now we will learn how to perform these crud operations.

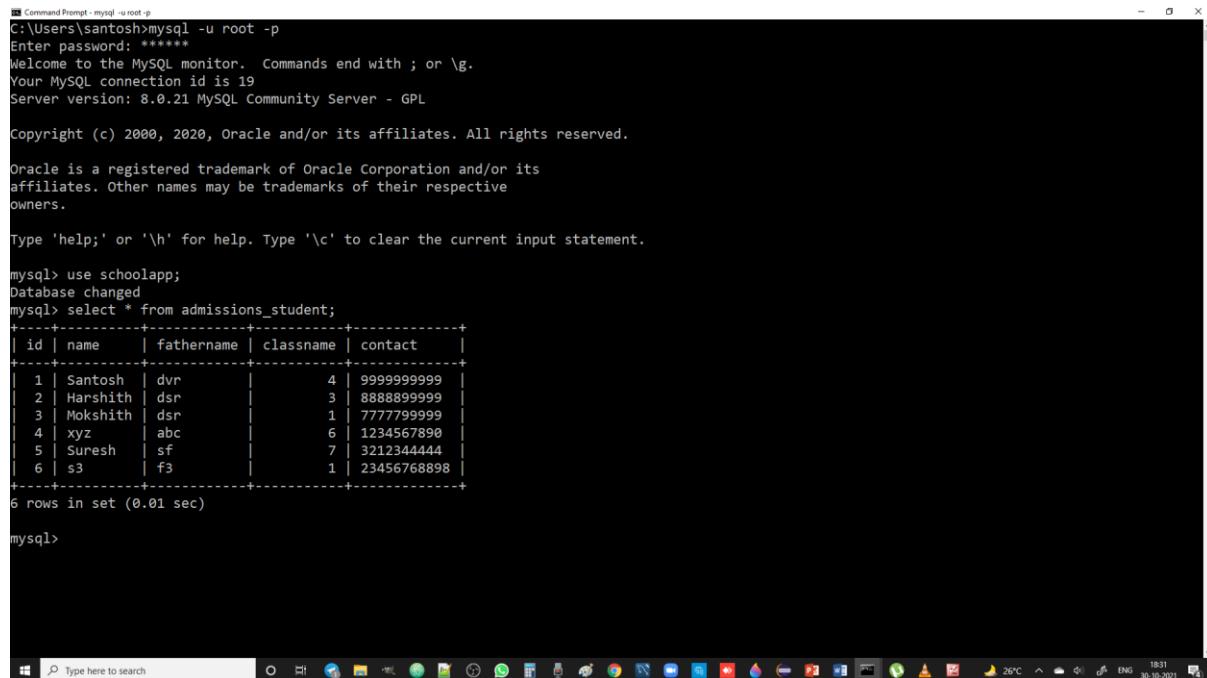
We already covered Create and Read in our previous lectures.

`form.save()` which we performed in above classes is Create operation

`model.objects.all()`, `model.objects.get(..)` these represents read operation.

Now let us learn how to update/delete some records based on id.

for this let us first check the existing table entries



```

C:\Command Prompt - mysql -u root -p
C:\Users\santosh>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.21 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use schoolapp;
Database changed
mysql> select * from admissions_student;
+----+-----+-----+-----+
| id | name | fathername | classname | contact |
+----+-----+-----+-----+
| 1 | Santosh | dvr | 4 | 9999999999 |
| 2 | Harshith | dsr | 3 | 8888889999 |
| 3 | Mokshith | dsr | 1 | 7777799999 |
| 4 | xyz | abc | 6 | 1234567890 |
| 5 | Suresh | sf | 7 | 3212344444 |
| 6 | s3 | f3 | 1 | 23456768898 |
+----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>
```

There will be a slight difference between other URLs and update, delete. To update or delete records from the database we must specify the id of that record to uniquely identify and perform the actions.

To delete a particular record with id 2 : First get the record and then perform delete operation.

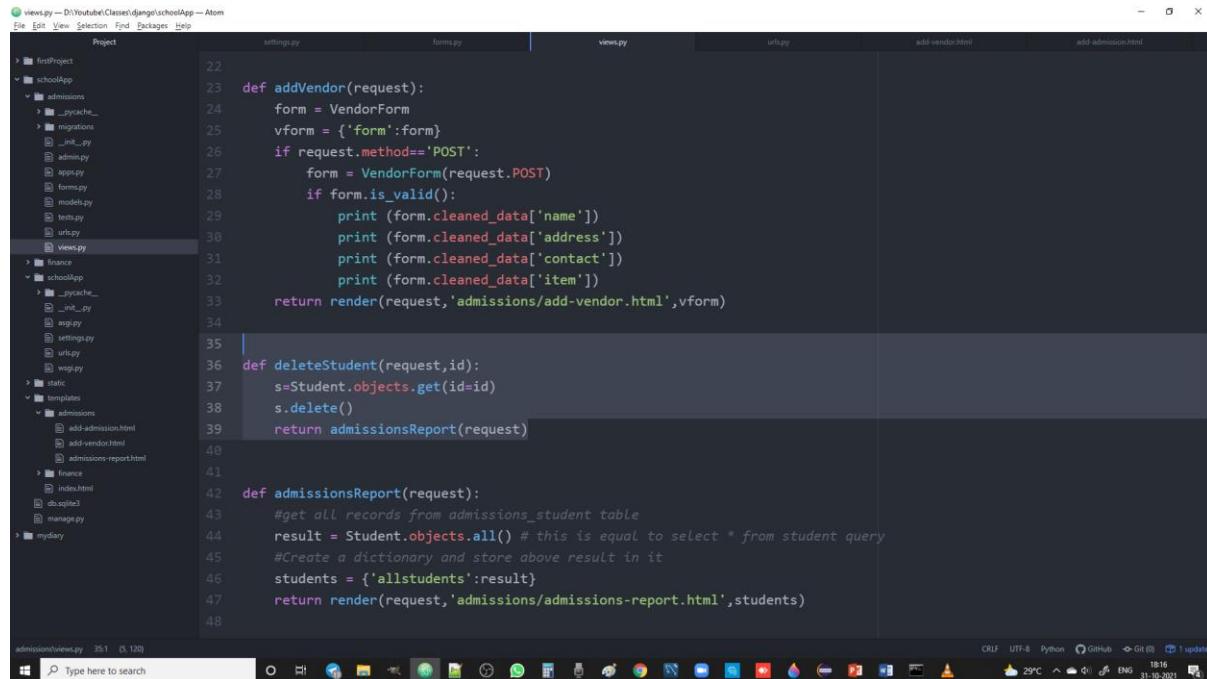
```

def deleteStudent(request,id):
    s = Student.objects.get(id = 2)
    s.delete()
```

in url we must specify that an integer value will be passed.

```
path('delete/<int:id>', deleteStudent)
```

Now let us implement delete operation and try to delete the record with id 2 from the above table



```

views.py -- D:\Youtube\Classes\django\schoolApp -- Atom
File Edit View Selection Find Packages Help
Project settings.py forms.py views.py urls.py add-vendor.html add-admission.html
admissions _pycache_ _migrations _migrations.py admin.py apps.py forms.py models.py tests.py test.py urls.py wsgi.py
views.py
finance schoolapp _pycache_ _migrations _migrations.py asgi.py settings.py urls.py wsgi.py
static admissions add-admission.html add-vendor.html admissions-report.html
finance index.html db.sqlite3 manage.py mydiary
admissions/views.py 35/1 (5, 120)
admissions/views.py 35/1 (5, 120) Type here to search O H E M S C G R P F Y T B D V A L I U P GitHub 10:16 29°C ENG 31-10-2021

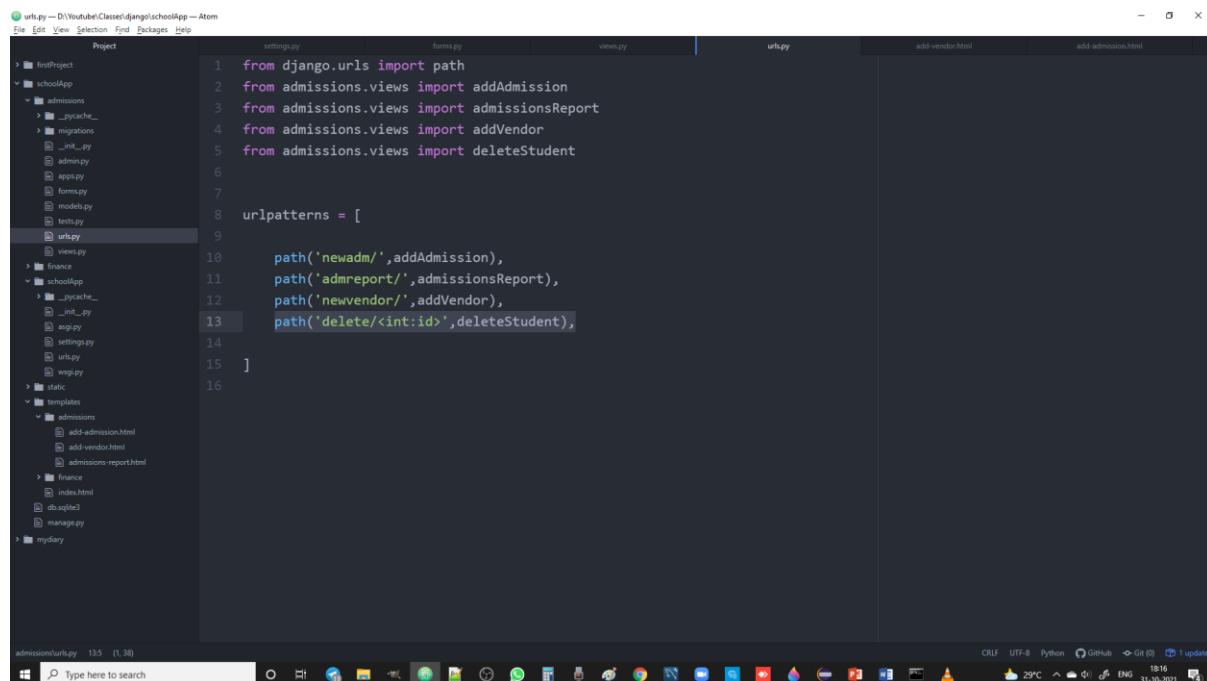
```

views.py

```

22
23 def addVendor(request):
24     form = VendorForm
25     vform = {'form':form}
26     if request.method=='POST':
27         form = VendorForm(request.POST)
28         if form.is_valid():
29             print (form.cleaned_data['name'])
30             print (form.cleaned_data['address'])
31             print (form.cleaned_data['contact'])
32             print (form.cleaned_data['item'])
33     return render(request,'admissions/add-vendor.html',vform)
34
35
36 def deleteStudent(request,id):
37     s=Student.objects.get(id=id)
38     s.delete()
39     return admissionsReport(request)
40
41
42 def admissionsReport(request):
43     #get all records from admissions_student table
44     result = Student.objects.all() # this is equal to select * from student query
45     #Create a dictionary and store above result in it
46     students = {'allstudents':result}
47     return render(request,'admissions/admissions-report.html',students)
48

```



```

urls.py -- D:\Youtube\Classes\django\schoolApp -- Atom
File Edit View Selection Find Packages Help
Project settings.py forms.py views.py urls.py add-vendor.html add-admission.html
admissions _pycache_ _migrations _migrations.py admin.py apps.py forms.py models.py tests.py test.py urls.py wsgi.py
views.py
finance schoolapp _pycache_ _migrations _migrations.py asgi.py settings.py urls.py wsgi.py
static admissions add-admission.html add-vendor.html admissions-report.html
finance index.html db.sqlite3 manage.py mydiary
admissions.urls.py 135 (1, 18)
admissions.urls.py 135 (1, 18) Type here to search O H E M S C G R P F Y T B D V A L I U P GitHub 10:16 29°C ENG 31-10-2021

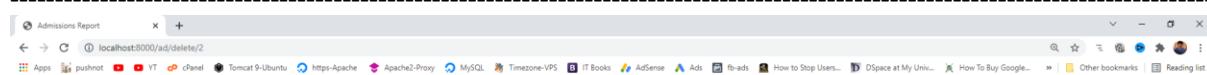
```

urls.py

```

1 from django.urls import path
2 from admissions.views import addAdmission
3 from admissions.views import admissionsReport
4 from admissions.views import addVendor
5 from admissions.views import deleteStudent
6
7
8 urlpatterns = [
9
10     path('newadm/',addAdmission),
11     path('admreport/',admissionsReport),
12     path('newvendor/',addVendor),
13     path('delete/<int:id>',deleteStudent),
14
15 ]
16

```



## Admissions Report

Student Name	Father Name	Class	Contact
Santosh	dvr	4	9999999999
Mokshith	dsr	1	7777799999
xyz	abc	6	1234567890
Suresh	sf	7	3212344444
s3	f3	1	23456768898



so, observe the database that the id=2 entry (Harshith) is deleted from the database.

```
Command Prompt - mysql -u root -p
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use schoolapp;
Database changed
mysql> select * from admissions_student;
+----+-----+-----+-----+-----+
| id | name | fathername | classname | contact |
+----+-----+-----+-----+-----+
| 1 | Santosh | dvr | 4 | 9999999999 |
| 2 | Harshith | dsr | 3 | 8888899999 |
| 3 | Mokshith | dsr | 1 | 7777799999 |
| 4 | xyz | abc | 6 | 1234567890 |
| 5 | Suresh | sf | 7 | 3212344444 |
| 6 | s3 | f3 | 1 | 23456768898 |
+----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> select * from admissions_student;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 22
Current database: schoolapp

+----+-----+-----+-----+-----+
| id | name | fathername | classname | contact |
+----+-----+-----+-----+-----+
| 1 | Santosh | dvr | 4 | 9999999999 |
| 3 | Mokshith | dsr | 1 | 7777799999 |
| 4 | xyz | abc | 6 | 1234567890 |
| 5 | Suresh | sf | 7 | 3212344444 |
| 6 | s3 | f3 | 1 | 23456768898 |
+----+-----+-----+-----+-----+
5 rows in set (0.06 sec)

mysql>
```

Now let us change the admission report template to add delete option so that the page will become user friendly.

admissions-report.html — D:\Youtube\Classes\django\schoolApp — Atom

File Edit View Selection Find Packages Help

Project admissions-report.html add-vendor.html add-admission.html

```

2 <html lang="en" dir="ltr">
3     <head>
4         <meta charset="utf-8">
5         <title>Admissions Report</title>
6     </head>
7     <body>
8         <h1>Admissions Report</h1>
9         <table border="1">
10            <tr>
11                <th>Student Name</th>
12                <th>Father Name</th>
13                <th>Class</th>
14                <th>Contact</th>
15                <th>Action</th>
16            </tr>
17            {% for s in allstudents %}
18            <tr>
19                <td>{{s.name}}</td>
20                <td>{{s.fathername}}</td>
21                <td>{{s.classname}}</td>
22                <td>{{s.contact}}</td>
23                <td><a href="/ad/delete/{{s.id}}">delete</a></td>
24            </tr>
25            {% endfor %}
26        </table>
27    </body>
28 </html>

```

templates\Admission\Admissions-report.html 15:32

CRU UTF-8 HTML GitHub Git (0) 1 update

24°C 2041 31-10-2021



## Admissions Report

Student Name	Father Name	Class	Contact	Action
Santosh	dvr	4	9999999999	<a href="#">delete</a>
Mokshith	dsr	1	7777799999	<a href="#">delete</a>
xyz	abc	6	1234567890	<a href="#">delete</a>
Suresh	sf	7	3212344444	<a href="#">delete</a>
s3	f3	1	23456768898	<a href="#">delete</a>



updating records :

updating is also follows the same style. But to update values we need to show the form to the user with existing values. Then user will update the required values and show the results.

views.py — D:\Youtube\Classes\django\schoolApp — Atom

```

File Edit View Selection Find Packages Help
Project views.py update-student.html urls.py admissions-report.html add-vendor.html add-admission.html
> firstProject
> schoolApp
  > admissions
    > __pycache__
    > migrations
      __init__.py
      admin.py
      apps.py
      forms.py
      models.py
      tests.py
      urls.py
    > views.py
  > finance
  > schoolapp
    > __pycache__
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
  > static
  > templates
    > admissions
      add-admission.html
      add-vendor.html
      admissions-report.html
      update-student.html
    > finance
      index.html
      db.sqlite3
      manage.py
  > mydairy
admissions/views.py 31:65 (11, 80)
Type here to search

```

CR LF UTF-8 Python GitHub ⌂ Git (0) ⌂ Update 25°C ENG 01-11-2021

update-student.html — D:\Youtube\Classes\django\schoolApp — Atom

```

File Edit View Selection Find Packages Help
Project views.py update-student.html urls.py admissions-report.html add-vendor.html add-admission.html
> firstProject
> schoolApp
  > admissions
    > __pycache__
    > migrations
      __init__.py
      admin.py
      apps.py
      forms.py
      models.py
      tests.py
      urls.py
    > views.py
  > finance
  > schoolapp
    > __pycache__
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
  > static
  > templates
    > admissions
      add-admission.html
      add-vendor.html
      admissions-report.html
      update-student.html
    > finance
      index.html
      db.sqlite3
      manage.py
  > mydairy
templates/admissions/update-student.html 22:50
Type here to search

```

CR LF UTF-8 HTML GitHub ⌂ Git (0) ⌂ Update 25°C ENG 01-11-2021



# Python Django Lecture Notes

TeluguWebGuru

The screenshot shows the Atom code editor interface with the following details:

- Project Tree:** On the left, it lists the project structure:
  - firstProject
  - schoolApp
    - migrations
    - \_\_init\_\_.py
    - admin.py
    - apps.py
    - forms.py
    - models.py
    - tests.py
  - urls.py
  - views.py
  - finance
  - schoolApp
    - \_\_init\_\_.py
    - asgi.py
    - settings.py
    - urls.py
    - wsgi.py
  - static
  - templates
    - admissions
  - mydairy
- Open Files:** The main area displays several files:
  - view.py
  - update-student.html
  - urls.py
  - admissions-report.html
  - add-vendor.html
  - add-admission.html
- Code Editor:** The code in urls.py is as follows:

```
from django.urls import path
from admissions.views import addAdmission
from admissions.views import admissionsReport
from admissions.views import addVendor
from admissions.views import deleteStudent
from admissions.views import updateStudent

urlpatterns = [
    path('newadm/',addAdmission),
    path('adreport/',admissionsReport),
    path('newvendor/',addVendor),
    path('delete/<int:id>',deleteStudent),
    path('update/<int:id>',updateStudent),
]
```
- System Status:** At the bottom, there are system status icons for CRF, UTF-8, Python, GitHub, Git (0), 25°C, ENG, and 01-11-2021.



## Admissions Report

Student Name	Father Name	Class	Contact	Action
Santosh	dvr	4	9999999999	<a href="#">update</a> <a href="#">delete</a>
Mokshith	dsr	1	7777799999	<a href="#">update</a> <a href="#">delete</a>
xyz	abc	6	1234567890	<a href="#">update</a> <a href="#">delete</a>
Suresh	sf	7	3212344444	<a href="#">update</a> <a href="#">delete</a>
s3	f3	1	23456768898	<a href="#">update</a> <a href="#">delete</a>



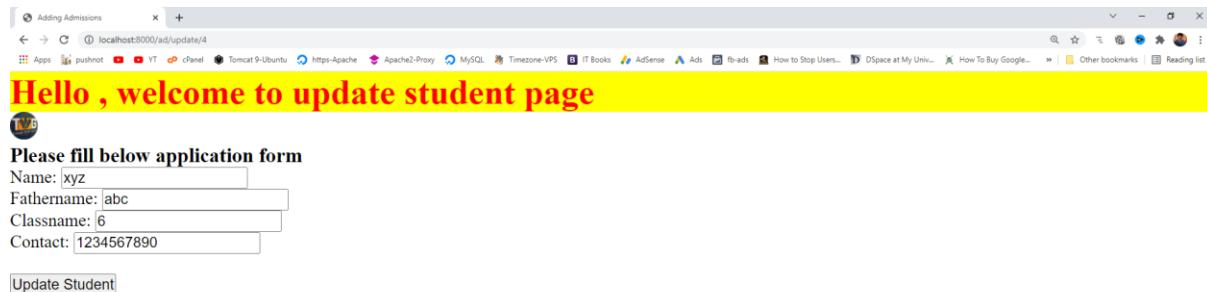
Now let us see the existing table data

```
Command Prompt - mysql -u root -p
mysql> select * from admissions_student;
+----+-----+-----+-----+-----+
| id | name | fathername | classname | contact |
+----+-----+-----+-----+-----+
| 1 | Santosh | dvr | 4 | 9999999999 |
| 3 | Mokshith | dsr | 1 | 7777799999 |
| 4 | xyz | abc | 6 | 1234567890 |
| 5 | Suresh | sf | 7 | 3212344444 |
| 6 | s3 | f3 | 1 | 23456768898 |
+----+-----+-----+-----+
5 rows in set (0.02 sec)

mysql>
```

Now let us change the name of the student with id =4 from xyz to Rajesh

When we click on update button in admissionsreport page it will open a form for us



Please fill below application form

Name: xyz

Fathername: abc

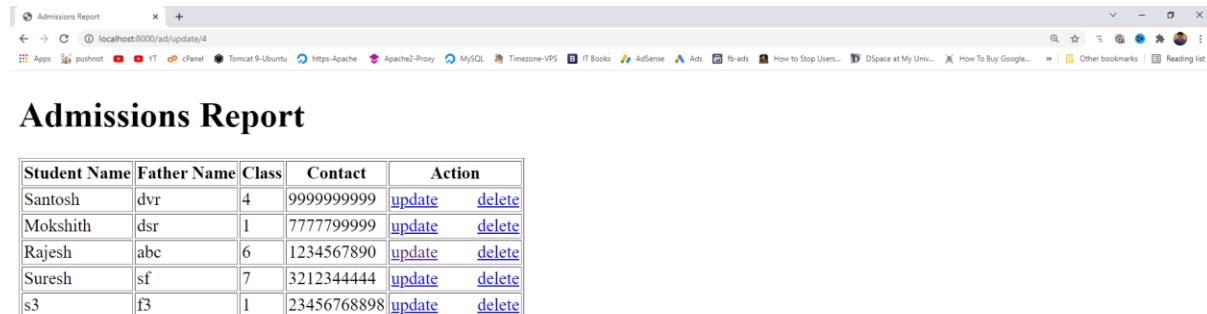
Classname: 6

Contact: 1234567890

**Update Student**



Now let us change the value of name to Rajesh and submit it.



**Admissions Report**

Student Name	Father Name	Class	Contact	Action
Santosh	dvr	4	9999999999	<a href="#">update</a> <a href="#">delete</a>
Mokshith	dsr	1	7777799999	<a href="#">update</a> <a href="#">delete</a>
Rajesh	abc	6	1234567890	<a href="#">update</a> <a href="#">delete</a>
Suresh	sf	7	3212344444	<a href="#">update</a> <a href="#">delete</a>
s3	f3	1	23456768898	<a href="#">update</a> <a href="#">delete</a>



Observe that student name is updated successfully.

## Class Based Views in DJANGO

### Class Based Views

We can create views by using classes also.

To create a class based views

- 1) we need to create a class that is inherited from django.views.generic.View

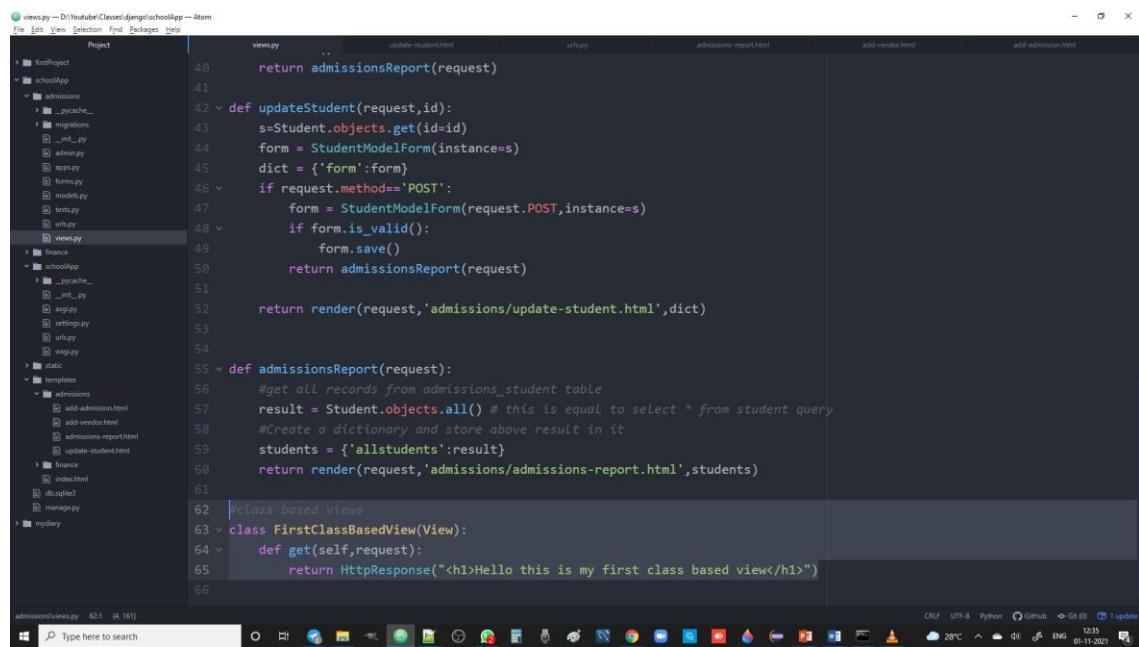
```
import django.views.generic import view
class exview(View):
    #class code
```

- 2) define separate methods for different http methods

```
def get(self,request):
    return HttpResponse("")
```

- 3) configure url in urls.py

```
path("",exview.as_view())
```



The screenshot shows the PyCharm IDE interface with the code editor open. The code is as follows:

```

views.py -- D:\Youtube\Classes\django\schoolapp -- Atom
Project views.py update-student.html urls.py admissions-report.html add-vendor.html add-admission.html
firstProject
hostApp
admissions
migrations
admin.py
tests.py
urls.py
forms.py
models.py
views.py
views.py
finance
schoolapp
__init__.py
settings.py
urls.py
wsgi.py
static
templates
admissions
add-admission.html
add-vendor.html
admissions-report.html
update-student.html
index.html
db.sqlite3
manage.py
mydry

40     return admissionsReport(request)
41
42     def updateStudent(request,id):
43         s=Student.objects.get(id=id)
44         form = StudentModelForm(instance=s)
45         dict = {'form':form}
46         if request.method=='POST':
47             form = StudentModelForm(request.POST,instance=s)
48             if form.is_valid():
49                 form.save()
50             return admissionsReport(request)
51
52         return render(request,'admissions/update-student.html',dict)
53
54
55     def admissionsReport(request):
56         #get all records from admissions_student table
57         result = Student.objects.all() # this is equal to select * from student query
58         #Create a dictionary and store above result in it
59         students = {'allstudents':result}
60         return render(request,'admissions/admissions-report.html',students)
61
62     class FirstClassBasedView(View):
63         def get(self,request):
64             return HttpResponse("<h1>Hello this is my first class based view</h1>")
65
66
admissions/views.py 62:1 (4/161)  ○ Type here to search  ○ 28°C  ENG 12:35  01-11-2021
```



The screenshot shows the Atom code editor with a Django project structure. The left sidebar displays the project tree:

- Project
- frisProject
- schoolApp
- admissions
  - \_\_init\_\_.py
  - migrations
    - \_\_init\_\_.py
  - admin.py
  - apps.py
  - forms.py
  - models.py
  - tests.py
- urls.py
- views.py

Below the tree, there are additional files: static, templates (with admissions, finance, index.html, add-admission.html, add-vendor.html, admissions-report.html, update-student.html), and mydir.

The main editor area shows the `urls.py` file content:

```
from django.urls import path
from admissions.views import addAdmission
from admissions.views import admissionsReport
from admissions.views import addVendor
from admissions.views import deleteStudent
from admissions.views import updateStudent
from admissions.views import FirstClassBasedView

urlpatterns = [
    path('newadm/',addAdmission),
    path('admreport/',admissionsReport),
    path('newvendor/',addVendor),
    path('delete/<int:id>',deleteStudent),
    path('update/<int:id>',updateStudent),
    path('firstcbv/',FirstClassBasedView.as_view()),
]
```

The status bar at the bottom indicates the file is "admissions/urls.py" with 18 lines, and the system status shows 28°C, 10:51 AM, and various system icons.



## Hello this is my first class based view

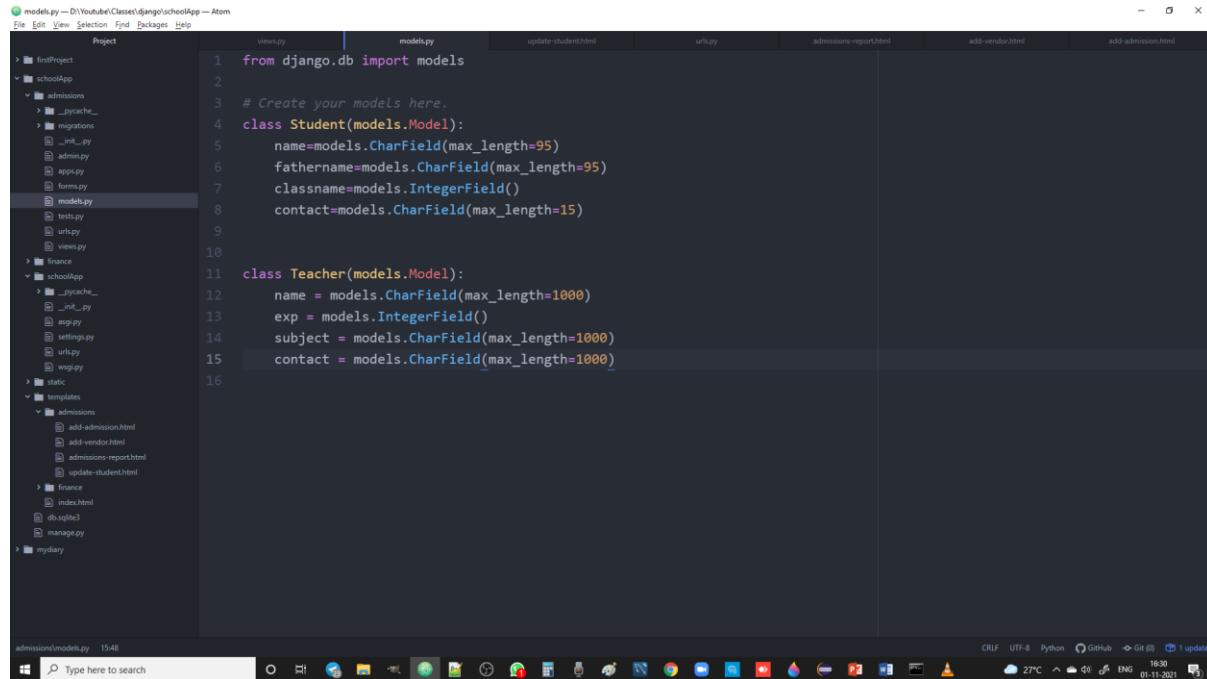


## CRUD Operations using Class Based Views in DJANGO

### CRUD Operations using class Based Views

We can do CRUD operations by using various views available in django.views.generic : ListView, DetailView, UpdateView, DeleteView

First let us create a model Teacher and then perform crud operations using class based views.



```

models.py -- D:\Youtube\Classes\django\schoolApp -- Atom
File Edit View Selection Find Packages Help
Project
> testProject
  > schoolApp
    > admissions
      > __pycache__
      > migrations
        __init__.py
        admin.py
        apps.py
        forms.py
        models.py
        tests.py
        urls.py
        views.py
    > finance
    > schoolApp
      > __pycache__
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
    > static
    > templates
      > admissions
        add-admission.html
        add-vendor.html
        admissions-report.html
        update-student.html
      > finance
        index.html
    db.sqlite3
    manage.py

admissions\models.py 15:48
Windows Type here to search C:\Windows\system32 GitHub 27°C ENG 01-11-2021 16:30
CRLF UTF-8 Python GitHub GitHub 1 update

```

models.py

```

from django.db import models

# Create your models here.
class Student(models.Model):
    name=models.CharField(max_length=95)
    fathername=models.CharField(max_length=95)
    classname=models.IntegerField()
    contact=models.CharField(max_length=15)

class Teacher(models.Model):
    name = models.CharField(max_length=1000)
    exp = models.IntegerField()
    subject = models.CharField(max_length=1000)
    contact = models.CharField(max_length=1000)

```

Now let us run makemigrations and migrate inorder to create teacher table automatically.



```
D:\Youtube\Classes\django\schoolApp>python manage.py makemigrations
Migrations for 'admissions':
  admissions\migrations\0002_teacher.py
    - Create model Teacher

D:\Youtube\Classes\django\schoolApp>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, admissions, auth, contenttypes, sessions
Running migrations:
  No migrations to apply.

D:\Youtube\Classes\django\schoolApp>
```

```
| django_session      |
+-----+
12 rows in set (0.03 sec)

mysql> desc admissions_teacher
-> ;
+----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra       |
+----+-----+-----+-----+-----+
| id   | int       | NO  | PRI | NULL    | auto_increment |
| name | varchar(1000) | NO  |     | NULL    |               |
| exp  | int       | NO  |     | NULL    |               |
| subject | varchar(1000) | NO  |     | NULL    |               |
| contact | varchar(1000) | NO  |     | NULL    |               |
+----+-----+-----+-----+-----+
5 rows in set (0.08 sec)

mysql>
```

Teacher table is created successfully.

Now let us insert data into the table and then perform crud operations on it.

```
Command Prompt - mysql -u root -p
mysql> select * from admissions_teacher;
+---+-----+-----+-----+
| id | name | exp | subject | contact |
+---+-----+-----+-----+
| 1 | santosh Raju | 15 | Computer Science | 999 |
| 2 | Ramya | 10 | Maths | 111 |
| 4 | Roshan Kumar | 12 | Physics | 943545 |
| 5 | Vijay | 11 | English | 234232 |
+---+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

## Read Operations:

To perform Read operation, we need to use ListView. Steps are as follows

- 1) create a class based view that inherits from ListView
- 2) provide value to the model attribute (mandatory)
- 3) default template name is modelName\_list.html you may pass your own template name by passing value to the template\_name attribute.
- 4) default context object name ( object that receives all the objects from ORM) is modelName\_list. You can set your own context\_object\_name by passing the value to this attribute.
- 5) Create a template with name modelName\_list.html and print the object's attributes from the list received (modelName\_list).
- 6) Configure the url (classname.as\_view())

## Retrieve a single row using DetailView

To perform this, we need to use DetailView as follows.

- 1) create a class-based view that inherits from DetailView
- 2) provide value to the model attribute (mandatory)
- 3) default template name is modelName\_detail.html you may pass your own template name by passing value to the template\_name attribute.

- 4) default context object name ( object that receives all the objects from ORM) is modelName. You can set your own context\_object\_name by passing the value to this attribute.
- 5) Create a template with name modelName\_detail.html and print the object's attributes from the list received (modelName\_list).
- 6) Configure the url - path('getdetails/<int:pk>',classname.as\_view())

### Create Operations:

To perform this operation, we need to use CreateView. Steps are as follows

- 1) create a class based view that inherits from CreateView
- 2) provide value to the model attribute (mandatory)
- 3) provide value to fields ( fields = (fields list separated by comma) )
- 4) default template name is modelName\_form.html you may pass your own template name by passing value to the template\_name attribute.
- 5) create a method get\_absolute\_url in model class

```
get_absolute_url(self):
    return reverse('urlname',kwargs={'pk':self.pk})
```

- 6) Create a template with name modelName\_form.html and print the object's attributes from the list received (modelName\_list).
- 7) Configure the url (classname.as\_view())

### Update Operations:

To perform Read operation, we need to use UpdateView. Steps are as follows

- 1) create a class based view that inherits from UpdateView
- 2) provide value to the model attribute (mandatory)
- 3) provide value to fields ( fields = (fields list separated by comma) )
- 4) default template name is modelName\_form.html you may pass your own template name by passing value to the template\_name attribute.
- 5) create a method get\_absolute\_url in model class

```
get_absolute_url(self):
    return reverse('urlname',kwargs={'pk':self.pk})
```

- 6) Create a template with name modelName\_form.html and print the object's attributes from the list received (modelName\_list).
- 7) Configure the url (classname.as\_view())

### Delete Operations:

To perform Read operation, we need to use DeleteView. Steps are as follows

- 1) create a class based view that inherits from DeleteView

- 2) provide value to the model attribute (mandatory)
  - 3) provide value to success\_url = reverse\_lazy('urlname')
  - 4) create model\_confirm\_delete.html to which django forward us and waits for our confirmation.
  - 5) Configure the url (classname.as\_view())
-

## **Security in Django:**

Django supports authentication and authorization automatically.

### **Authentication Steps:**

- 1) Add the auth urls
- 2) Create the login form
- 3) Secure the views
- 4) Create the users
- 5) Test it

#### **Add the auth urls:**

We will use include from django.urls to implement this.

- 1) Add a path in urls.py that points to accounts

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

This step will link all the urls related to auth to accounts. now we can use signin , signout by using /accounts/signin, /accounts/signout so on.,

#### **Create the login form:**

create a folder registration under templates

under this create a file login.html

```
<form method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit" value="Login">
</form>
```

#### **Secure the views:**

In the views.py apply the @login\_required decorator to each view

```
from django.contrib.auth.decorators import login_required
```

now add @login\_required above each view definition

#### **Create the users:**

Create user through admin form with the admin credentials that we created in earlier classes

**Now we can login with the newly created user credentials and access the application.**

To Logout from the application use the url /accounts/logout

To forward the users to a particular page after logout just include the following line at the end of the settings.py

```
LOGOUT_REDIRECT_URL = '/'
```

To implement a custom logout page then create a view with template and give that url in settings.py

## **Authorization:**

Enter into admin login and click the user that we just created and there in the user page we can select and give selected authorizations to the particular user.

```
import permission_required as well along with login_required
```

Now add @permission\_required('appname.actionname\_modelname') with each view

-----

You can add a group from admin login and assign the required authorizations to that group also so that all the users added into this group will automatically get those authorizations.

## **Template Inheritance:**

create a template with common code by using block template tag

```
<html>
<head></head>
<body>
    { %block body_block% }
```

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

In other Html Files just include the following statement

```
{ %extends 'appname/commontemplatename.html'% }
{ %block body_block% }
```

## SESSION MANAGEMENT IN DJANGO

HTTP is a stateless protocol. That means server does not store the state of the application anywhere. whenever a new request comes then it loses all the temporary information (state) that is stored as part of processing previous request and it just starts storing current request's state.

We can solve this problem by using one of the following techniques

- 1) Hidden Variables
- 2) Cookies
- 3) Sessions

Hidden Variables:

Example Program: views.py

```
from django.shortcuts import render

from admissions.models import Student

from admissions.forms import StudentModelForm

from admissions.forms import VendorForm

from django.views.generic import View,ListView,DetailView,CreateView,UpdateView,DeleteView

from django.http import HttpResponse

from admissions.models import Teacher

from django.urls import reverse_lazy

from django.contrib.auth.decorators import login_required,permission_required

# Create your views here.

#function based views
```

```
@login_required

def homepage(request):
    return render(request,'index.html',{'user':request.user})

def logoutUser(request):
    return render(request,'logout.html')

@login_required

def addAdmission(request):

    #retrieve values from hidden variables

    n = request.GET.get('nm')

    a = request.GET.get('addr')

    c = request.GET.get('cont')

    i = request.GET.get('it')

    dict = {"name":n,"address":a,"contact":c,"item":i}

    form = StudentModelForm

    studentform = {'form':form,"name":n,"address":a,"contact":c,"item":i}

if request.method=='POST':

    form = StudentModelForm(request.POST)

    if form.is_valid():

        form.save()
```

```
return render(request,'index.html',dict)

return render(request,'admissions/add-admission.html',studentform);

@login_required
@permission_required('admissions.view_student')

def admissionsReport(request):

    #get all the records from the table

    result = Student.objects.all(); #SELECT * FROM students

    #store it in dictionary students

    students = {'allstudents':result}

    return render(request,'admissions/admissions-report.html',students);

@login_required
@permission_required('admissions.delete_student')

def deleteStudent(request,id):

    s = Student.objects.get(id=id) # select * from admissions_student where id=idvalue

    s.delete()

    return admissionsReport(request)

@login_required
```

```
@permission_required('admissions.change_student') #add delete change view

def updateStudent(request,id):
    s = Student.objects.get(id=id)
    form = StudentModelForm(instance=s)
    dict = {'form':form}

    if request.method=='POST':
        form = StudentModelForm(request.POST,instance=s)
        if form.is_valid():
            form.save()
        return admissionsReport(request)

    return render(request,'admissions/update-admission.html',dict)

@login_required

def addVendor(request):
    form = VendorForm
    vform = {'form':form}

    if request.method=='POST':
        pass
```

```
form = VendorForm(request.POST)

if form.is_valid():

    n = form.cleaned_data['name']

    a = form.cleaned_data['address']

    c = form.cleaned_data['contact']

    i = form.cleaned_data['item']

    dict = {"name":n,"address":a,"contact":c,"item":i}

    return render(request,'index.html',dict);

return render(request,'admissions/add-vendor.html',vform);

#class based view

class FirstClassBasedView(View):

    def get(self,request):

        return HttpResponse("<h1>Hello ... this is my first class based view</h1>")

class TeacherRead(ListView):

    model = Teacher
```



```
class GetTeacher(DetailView):
    model = Teacher

class AddTeacher(CreateView):
    model = Teacher
    fields = ('name', 'subject', 'exp', 'contact')

class UpdateTeacher(UpdateView):
    model = Teacher
    fields = ('name', 'contact')

class DeleteTeacher(DeleteView):
    model = Teacher
    success_url = reverse_lazy('listteachers')
```

## index.html

```
{% load static %}

<html>
  <head>
    <title>add admissions page</title>
    <link rel="stylesheet" href="{% static "stylesheets/style.css" %}" />
  </head>
```

```
<body>

<h2>



telugu web guru academy

</h2>

<br/>

<hr/>

<br/>

<br/><br/><br/>

<div class="module">

<h1>ADMISSIONS</h1><br/>

<form action="/ad/newadm">

    {% csrf_token %}

    <button type="submit" name="button">Add Admission</button></a><br/><br/>

    <input type="hidden" name="nm" value="{{name}}>

    <input type="hidden" name="addr" value="{{address}}>

    <input type="hidden" name="cont" value="{{contact}}>

    <input type="hidden" name="it" value="{{item}}>

</form>

<a href="/ad/admreport"><button type="button" name="button">Admission Report</button></a><br/><br/>
```

```
</div>

<div class="module">

<h1>FINANCE</h1><br/>

<a href="/fin/feecoll"><button type="button" name="button">Fee Collection</button></a><br/><br/>

<a href="/fin/collectionreport"><button type="button" name="button">Fee Collection Report</button></a><br/><br/>

<a href="/fin/duesreport"><button type="button" name="button">Fee Dues Report</button></a>

</div>

<br/><br/>

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

<br/><br/>

{{name}}<br/>

{{address}}<br/>

{{contact}}<br/>

{{item}}<br/>

</body>
```

```
</html>
```

add-admission.html

```
{% load static %}

<html>

<head>

<title>add admissions page</title>

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

</head>

<body>

<h1>Hello , Welcome to add admissions page.</h1>



<h3>please fill below application form</h3><br/><br/>

<form method="POST">

<table>

{{form.as_table}}


</table>
```

```
<br/>

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

<% csrf_token %>

</form>

<br/><br/>

{{name}}<br/>

{{address}}<br/>

{{contact}}<br/>

{{item}}<br/>

</body>

</html>
```



Hello , Welcome to add vendor page.

please fill below application form

Name: kishore  
Address: vizag  
Contact: 123232  
Item: Online Courses

[Add Vendor](#)



**ADMISSIONS**

[Add Admission](#)  
[Admission Report](#)

**FINANCE**

[Fee Collection](#)  
[Fee Collection Report](#)  
[Fee Dues Report](#)

[Logout](#)

kishore  
vizag  
123232  
Online Courses



**Hello , Welcome to add admissions page.**

please fill below application form

Name:	<input type="text"/>
Fathername:	<input type="text"/>
Classname:	<input type="text"/>
Contact:	<input type="text"/>

**Add Student**

kishore  
vizag  
123232  
Online Courses

cookie :

The following are used to set and retrieve the cookies in django

```
response.set_cookie(name,value,expirytime)
requests.COOKIES
```

The following methods are used to check whether cookies are enabled in client browser or not

```
request.session.set_test_cookie()
request.session.test_cookie_worked()
request.session.delete_test_cookie()
```

```
from django.shortcuts import render

from admissions.models import Student

from admissions.forms import StudentModelForm

from admissions.forms import VendorForm

from django.views.generic import View,ListView,DetailView,CreateView,UpdateView,DeleteView

from django.http import HttpResponseRedirect

from admissions.models import Teacher
```



```
from django.urls import reverse_lazy

from django.contrib.auth.decorators import login_required,permission_required

# Create your views here.

#function based views

@login_required

def homepage(request):

    return render(request,'index.html',{'user':request.user})

def logoutUser(request):

    return render(request,'logout.html')

@login_required

def addAdmission(request):

    form = StudentModelForm

    studentform = {'form':form}

    if request.method=='POST':

        form = StudentModelForm(request.POST)

        if form.is_valid():

            form.save()
```

```
return homepage(request)

return render(request,'admissions/add-admission.html',studentform);

@login_required
@permission_required('admissions.view_student')

def admissionsReport(request):

    #get all the records from the table

    result = Student.objects.all(); #SELECT * FROM students

    #store it in dictionary students

    students = {'allstudents':result}

    return render(request,'admissions/admissions-report.html',students);

@login_required
@permission_required('admissions.delete_student')

def deleteStudent(request,id):

    s = Student.objects.get(id=id) # select * from admissions_student where id=idvalue

    s.delete()

    return admissionsReport(request)

@login_required
```

```
@permission_required('admissions.change_student') #add delete change view

def updateStudent(request,id):

    s = Student.objects.get(id=id)

    form = StudentModelForm(instance=s)

    dict = {'form':form}

    if request.method=='POST':

        form = StudentModelForm(request.POST,instance=s)

        if form.is_valid():

            form.save()

        return admissionsReport(request)

    return render(request,'admissions/update-admission.html',dict)

@login_required

def addVendor(request):

    form = VendorForm

    vform = {'form':form}

    if request.method=='POST':
```

```
form = VendorForm(request.POST)

if form.is_valid():

    n = form.cleaned_data['name']

    a = form.cleaned_data['address']

    c = form.cleaned_data['contact']

    i = form.cleaned_data['item']

    response = render(request,'index.html');

    response.set_cookie("name",n)

    response.set_cookie("address",a)

    response.set_cookie("contact",c)

    response.set_cookie("item",i)

return response

return render(request,'admissions/add-vendor.html',vform);

#class based view

class FirstClassBasedView(View):

    def get(self,request):

        return HttpResponse("<h1>Hello ... this is my first class based view</h1>")
```

```
class TeacherRead(ListView):
    model = Teacher

class GetTeacher(DetailView):
    model = Teacher

class AddTeacher(CreateView):
    model = Teacher
    fields = ('name','subject','exp','contact')

class UpdateTeacher(UpdateView):
    model = Teacher
    fields = ('name','contact')

class DeleteTeacher(DeleteView):
    model = Teacher
    success_url = reverse_lazy('listteachers')
```

add-admission.html

```
{% load static %}

<html>

<head>

<title>add admissions page</title>

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

</head>

<body>

<h1>Hello , Welcome to add admissions page.</h1>



<h3>please fill below application form</h3><br/><br/>

<form method="POST">

<table>

{{form.as_table}}


</table>

<br/>

<input type="submit" value="Add Student">
```

```
{% csrf_token %}

</form>

<br/><br/>

{% for key,value in request.COOKIES.items %}

{{key}} - {{value}}<br/>

{% endfor %}

</body>

</html>
```

Hello , Welcome to add admissions page.

please fill below application form

Name:

Fathername:

Classname:

Contact:

**Add Student**

csrfToken - QtRDxpxXcm6oFfUhk3DmtRGyGMjXypCPVj8rPhuPRNBuAJ3o23a0yagKCukRzkwL  
 sessionid - 5zop0owvzeagmqn40issefrblal3g7uh  
 name - TMH  
 address - Bangalore  
 contact - 234  
 item - Books

### Session :

Session is the time between a user login and user logout

or

Sequence of actions performed by a user

```
request.session.get(session-name,default-value)
```

```
request.session[name]=value
```

```
from django.shortcuts import render

from admissions.models import Student

from admissions.forms import StudentModelForm

from admissions.forms import VendorForm

from django.views.generic import View,ListView,DetailView,CreateView,UpdateView,DeleteView

from django.http import HttpResponse

from admissions.models import Teacher

from django.urls import reverse_lazy

from django.contrib.auth.decorators import login_required,permission_required

# Create your views here.

#function based views

@login_required

def homepage(request):

    return render(request,'index.html',{'user':request.user})



def logoutUser(request):

    return render(request,'logout.html')




@login_required
```

```
def addAdmission(request):  
  
    form = StudentModelForm  
  
    studentform = {'form':form}  
  
  
    if request.method=='POST':  
  
        form = StudentModelForm(request.POST)  
  
        if form.is_valid():  
  
            form.save()  
  
        return homepage(request)  
  
  
    return render(request,'admissions/add-admission.html',studentform);
```

```
@login_required  
  
@permission_required('admissions.view_student')  
  
def admissionsReport(request):  
  
    #get all the records from the table  
  
    result = Student.objects.all(); #SELECT * FROM students  
  
    #store it in dictionary students  
  
    students = {'allstudents':result}  
  
    return render(request,'admissions/admissions-report.html',students);
```

```
@login_required
```

```
@permission_required('admissions.delete_student')

def deleteStudent(request,id):
    s = Student.objects.get(id=id) # select * from admissions_student where id=idvalue
    s.delete()
    return admissionsReport(request)

@login_required

@permission_required('admissions.change_student') #add delete change view

def updateStudent(request,id):
    s = Student.objects.get(id=id)
    form = StudentModelForm(instance=s)
    dict = {'form':form}

    if request.method=='POST':
        form = StudentModelForm(request.POST,instance=s)
        if form.is_valid():
            form.save()
    return admissionsReport(request)

return render(request,'admissions/update-admission.html',dict)
```

```
@login_required

def addVendor(request):

    form = VendorForm

    vform = {'form':form}

    if request.method=='POST':

        form = VendorForm(request.POST)

        if form.is_valid():

            n = form.cleaned_data['name']

            a = form.cleaned_data['address']

            c = form.cleaned_data['contact']

            i = form.cleaned_data['item']

            response = render(request,'index.html');

            request.session['name']=n;

            request.session['address']=a;

            request.session['contact']=c;

            request.session['item']=i;

    return response
```

```
return render(request,'admissions/add-vendor.html',vform);

#class based view

class FirstClassBasedView(View):
    def get(self,request):
        return HttpResponse("<h1>Hello ... this is my first class based view</h1>")

class TeacherRead(ListView):
    model = Teacher

class GetTeacher(DetailView):
    model = Teacher

class AddTeacher(CreateView):
    model = Teacher
    fields = ('name','subject','exp','contact')
```

```
class UpdateTeacher(UpdateView):
    model = Teacher
    fields = ('name', 'contact')

class DeleteTeacher(DeleteView):
    model = Teacher
    success_url = reverse_lazy('listteachers')
```

```
{% load static %}

<html>

<head>

<title>add admissions page</title>

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

</head>

<body>

<h1>Hello , Welcome to add admissions page.</h1>



<h3>please fill below application form</h3><br/><br/>
```

```
<form method="POST">

<table>

{{form.as_table}}


</table>

<br/>

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




{% csrf_token %}

</form>

<br/><br/>

{% for key,value in request.session.items %}

{{key}} - {{value}}<br/>

{% endfor %}

</body>

</html>
```



Hello , Welcome to add admissions page.

Please fill below application form

Name:   
Fathername:   
Classname:   
Contact:

```
_auth_user_id - 2
_auth_user_backend - django.contrib.auth.backends.ModelBackend
_auth_user_hash - 4e723701b323ef81ff62708c8fc79188f7c966df9537f32da2205dca3f3e4d490
name - abcd
address - hyd
contact - 4321
item - pencils
```

