Run the XAMP server each and every time when you start to work on Django Software.

Django **Framework**:

Framework: It is collection of Python Libraries to develop Dynamic Websites

After type anycode in Django just save by pressing ctrl+s button.

Step-1 to work on Django framework

(How to install Python,Create virtual environment, install Django, create project, run Django Server, create app, link app under project

1) Installation: Python must be installed.
2) After installing Python3, we have to set environment variable path.
3) After setting the path, we have to type “pip” command on cmd prompt to check python is installed and path is set successfully or not.
4) After this we will create folder according to our need in any drive to work on project in Django Framework.
5) To create folder we will type on cmd in any drive location “mkdir <folder_name>” and “press enter key”.
   e.g. D:\>mkdir django-project1 “press enter key”
6) To use this created folder or any pre created folder again we have to type on cmd “cd <folder_name>” and “press enter key”.
   e.g. D:\>cd django-project “press enter key”
7) After this we will create environment and to create environment we will give command.
   a. D:\django-project>pip install virtualenv “press enter”

Run the XAMP server each and every time when you start to work on Django Software.

b. D:\django-project> virtualenv <VirtualEnvirnomentName>
   e.g. D:\django-project> virtualenv myenv
c. After this "myenv" folder will be created under folder name.
d. After this We will activate "myenv".
e. To activate our virtual environment, follow given command below.
f. D:\django-project> myenv\Scripts\activate "press enter key"
g. After this our command screen will look like this.
   (myenv) D:\django-project>

8) After this we will install "Django framework" through command on cmd .
   (myenv) D:\django-project> pip install Django "press enter key".
9) To check "Django" is properly installed ->
   (myenv) D:\django-project>dir "press enter key".
10) We will get message on cmd screen "Directory of D:\django-project"
11) To confirm that "Django" is properly available to any one we will run command.
   (myenv) D:\django-project> Django-admin "press enter key".
   we will get message on cmd screen "Available Subcommands……" It means "Django" framework is ready to use'

12) To create Project in django:-

i) Type given command on cmd
(myenv) D:\django-project>Django-admin startproject <project_name> . (type this dot)
"press enter".
e.g. (myenv) D:\django-project>Django-admin startproject Organization . (type this dot also)

ii) . dot will stop by creating nested folder of our project

Run the XAMP server each and every time when you start to work on Django Software.

iii) To check project is created or not give command:
(myenv) D:\django-project>dir "press enter key" if project is created then we can see name of our project clearly on screen.

iv) After the previous command of cmd, we can see the name of project in form of folder in "D drive location".

v) To see the details of created project we will give command on cmd.
(myenv) D:\django-project>dir "press enter".

vi) We will see "manage.py" file under project name.

vii) "manage.py" file is very important file for our project to run, to manage, to execute, to debug, to start, to stop whatever we want to do with our project.

viii) To run our project on server, type the command on cmd after ">":
(myenv) D:\django-project>python manage.py runserver "press enter key"

ix) After this previous command we will see in last line on cmd
"Starting development server at http://127.0.0.1:8000/" message.

x) After that we will copy this link "http://127.0.0.1:8000/".

xi) After copy this link, we will go to any browser and paste this and "press enter key"

Run the XAMP server each and every time when you start to work on Django Software.

xii) After executing this command on browser we will get this message on browser.
"The install worked successfully! Congratulations!"

xiii) To close the server we will give command to server by using "ctrl+c". After this we will return to our cmd prompt on default folder.

xiv) In above link "http://127.0.0.1" is our server which is responsible to execute our command. Which is responsible to run Django project. And this server is inbuilt local server for Django.

xv) "127.0.0.1" is an IP address of our computer. In other language we can say "Access the server on my own machine".

xvi) "8000" is port number.

xvii) On single server, We can serve multiple services like Django is running on default port "8000" on local server http://127.0.0.1. Or "localhost"

xviii) It means "http://127.0.0.1" this server can serve file, can server time, can serve printer.

(14) To start app inside our project:

(myenv) D:\django-project>Django-admin startapp <app_name> "press enter key"

Run the XAMP server each and every time when you start to work on Django Software.

e.g. (myenv) D:\django-project> Django-admin startapp employees "press enter".

(15) After this another folder of App will be created with name "employees".

(16) We will install our created app "employees" under project name "Organization"

(17) To install : we will go to setting.py file of our project folder "Organization".

(18) After this we will search INSTALLED_APPS=[]

(19) We will type in INSTALLED_APPS like given below

```
INSTALLED_APPS =[……,
……,
'employees',      "type this app name here"
]
```

(20) After installing our application "employees" under project "Organization", we will include application's "urls" in Django project's "urls.py" file.

(21) To include our app's urls.py in project's urls.py file we will follow below steps carefully.

(22) After clicking on urls.py file of project we will get the screen as it is given below.

```
from django.contrib import admin
urlpatterns = [
path('admin/', admin.site.urls),
```

Run the XAMP server each and every time when you start to work on Django Software.

```
]
```

(23) Now going to include application's urls.py file in Django project's urls.py file. For this, few things will added in upper code. So see this clearly

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

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

(24) First argument of 1st path method of above urlpatterns is blank under single quotation because our request comes first here inside our project's urlpatterns then it go to our app. Means request from urls comes first in project and then go to app.

Run the XAMP server each and every time when you start to work on Django Software.

(25) Now we have to create urls.py file in our app by selecting first and right click then click on new file and give the file name urls.py. and press enter.

(26) Now urls.py file under app is created.

Now write the given code in this file given below.

```
from django.urls import path,include

urlpatterns = [
]
```

Run the XAMP server each and every time when you start to work on Django Software.

Step-2 to work on Django framework

(How to work with Django Admin Panel?)

Q- What is Django-Admin?

Ans. Django provides a ready-to-use user interface for administrative activities. We all know how an admin interface is important for a web project. Django automatically generates admin UI based on your project models.

Q. What does Migrate mean?

Ans. Database is provided by Django itself. Our database means the database of Django, the tables of Django are scanned into the database, it commits under the database.

(27) Now we will Migrate Django Tables. Essentially, Django has 10 tables of its own. It belongs by default so it is compulsory to migrate them.

(28) To migrate, fire the below given command on cmd prompt.

(myenv) D:\django-project1> python manage.py migrate "press enter"

(29) So what this command does is, it migrates all your Django tables in our particular database. After this "db.sqllte3" SQLLTE3 file will be created

Run the XAMP server each and every time when you start to work on Django Software.

inside our folder name. It means Django has created a database file.

(30) We have to execute above command compulsorily because then our Admin panel will start working.

(31) Without this "db.sqllite3" file, We can not access our Admin Panel that's why We have migrated.

Now to work on Admin Panel We have to create superuser.

SUPERUSER:- Super User is one type of main Admin who has all the authorities, all the permissions is called Super User.

(32) Now to open the Admin Panel, We will run our server.

(myenv) D:\django-project1> python manage.py runserver "press enter"

After this previous command we will see in last line on cmd

"Starting development server at http://127.0.0.1:8000/" message.

(33) After that we will copy this link "http://127.0.0.1:8000/".

Run the XAMP server each and every time when you start to work on Django Software.

(34) After copy this link, we will go to any browser and paste this and “press enter key”

(35) After executing this command on browser we will get this message on browser.
“The install worked successfully! Congratulations!”

(36) In URL of browser, single click and then type slash admin “/admin”

127.0.0.1:8000/admin “hit enter”

(37) We will switch to Admin Panel where login page will be available and if we enter user name and password then we get error. It means We have to create super user.

(38) Before creating super user, We have to stop server on cmd prompt by pressing “ctrl+c” .

(39) To create superuser, we will type below command:
(myenv) D:\django-project1> python manage.py createsuperuser.

(40) After hitting enter key, Django will ask for few credentials like Username, email-address, Password. You can leave blank username for simplicity

(41) “Username (leave blank to use ‘admin’): “
“press enter”

Run the XAMP server each and every time when you start to work on Django Software.

(42) "Email address:" give any fake email "enter key"

(43) "Password:" "press enter key" I will give admin as password for simplicity

(44) "Password (again):" "press enter" same password like "admin".

(45) Now we will get message after creating superuser successfully.
"Superuser created successfully"

(46) We can login to our admin panel successfully.

1. To login to our Admin follow the above steps from 32 to 37.

Run the XAMP server each and every time when you start to work on Django Software.

Step-3 to work on Django framework (How to Add Models in Admin panel in Django)

1. **What is Django Models?**
   Ans. A model is a class that represents table or collection in our Database, and where every attribute of the class is a field of the table or collection.
2. **Models are defined in the app/models.py (in my example: teacher/models.py)**
3. **Creating a Model:**
   Structure of models:

```
class ModelName(models.Model):
        FieldName=models.datatype(size)
```

4. **First models in above code is models of Django.**
5. **Second Model shows that particular name of our table.**

Run the XAMP server each and every time when you start to work on Django Software.

e.g.

```
class Students(models.Model):  <press enter>
    Firstname=models.CharField(max_length=50)
    Lastname=models.CharField(max_length=50)
    Email=models.EmailField(max_length=50)
    Contact=models.CharField(max_length=50)
```

6. Models defined by user is created, after this we will migrate(convert) this Model class into table form
7. using "python manage.py makemigrations" press enter after that
8. again type "python manage.py migrate" press enter
9. After executing above command we get message on our cmd prompt

```
Migrations for 'students':
    students\migrations\0001_initial.py
      - Create model students
```

10. Now we will run our server by command (myenv) D:\django-project> python manage.py runserver "press enter"
11. After migrations process, we will register our Models in admin.py file of app.

Run the XAMP server each and every time when you start to work on Django Software.

12. Type the below command to register Models classes:

```
from .models import *

admin.site.register(Students)
```

13.Again run the server, go to admin panel and we will see the name of Models class Students.
14.Now we can add declared data in Students Table.

Step-4 to work on Django framework
(How to Load template on browser in Django)

1.Q. How to create template in Django?
Ans. Go to your app folder-> create new folder(Templates)->Again create new folder(Name of your App for easy purpose->Create new file with name insert.html "press enter"

2. Now write your html code inside this file to load on browser.

Run the XAMP server each and every time when you start to work on Django Software.

```
from django.shortcuts import render

# Create your views here. It is insert.html view

def InsertPageView(request):
    return render(request,"students /insert.html")
```

3. To call this views.py file, we will go to urls.py file of app to set path. So follow the instruction as given below:

```
from django.urls import path,include
  from . import views
  urlpatterns = [
  path("",views.InsertPageView,name="insertpage"),
    ]
```

Run the XAMP server each and every time when you start to work on Django Software.

4. Now run the server and templates will be called where content of insert.html file will be displayed.

Step-5 to work on Django framework (Display Forms in HTML)

Q. What do you mean by Forms?
Ans. An HTML form is used to collect user input. The user input is most often sent to a server for processing.
An HTML <form> element is used to create an HTML form for user input.

1. To display form through templates, we have to again create file under app folder of templates folder insert.html file.
employees->templates->employees->insert.html
2. Now start write the html code under this insert.html file.

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

Run the XAMP server each and every time when you start to work on Django Software.

```
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>this is form through templates</title>
</head>
<body>
    <centre>
        <h1> HTML FORM</h1>
        <form action="" method="post">
            <table>
                <tr>
                    <td>Firstname</td>
                    <td><input type="text" name="fname"
id=""></td>
                </tr>
                <tr>
                    <td>Lastname</td>
                    <td><input type="text" name="lname"
id=""></td>
                </tr>
                <tr>
                    <td>Email</td>
                    <td><input type="text" name="email"
id=""></td>
                </tr>
                <tr>
```

Run the XAMP server each and every time when you start to work on Django Software.

```
                <td>Contact</td>
                <td><input type="tel" name="contact"
id=""></td>
            </tr>
            <tr>
                <td><input type="submit"
value="Insert"></td>
            </tr>
        </table>
    </form>
  </centre>
</body>
</html>
```

3.Set the urlpattern path under urls.py file under app folder.

```
from django.urls import path,include
from . import views

urlpatterns = [

path("",views.InsertPageViews,name="insertpage"),
    ]
```

Run the XAMP server each and every time when you start to work on Django Software.

4. Now run the server and Login page will be displayed on browser.

Step-1 Again to work on Django framework
(Working on CRUD Application)

(A-How to Connect with Mysql in Django)

1- **Follow Step-1 instruction completely again to work on database for this unit.**

2- First of all we have to open xampp Control Panel and click on Admin against MySQL, It will switch to phpMyAdmin page.

3- Click on New Option and create database name CRUD (IT is just name; means C-create, R-read, U-update, and D-delete the data from data base)

4- And click on create button.

5- After creating database, go to settings.py file of project and search

```
DATABASES= {
  'default' :
    {
    }
}
```

Run the XAMP server each and every time when you start to work on Django Software.

Replace the default from below code:

```
DATABASES = {
 'default':
    {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'crud',   #Your database name
        'HOST': 'localhost', #Your localhost name
        'PORT': '3306',
        'USER': 'root',   # Your Database Username
        'PASSWORD': '', # YourDatabaseUser Paswrd
    }
}
```

6- Go to cmd from where you run your project. (Do not run)

7- Now type this command “python manage.py migrate” “press enter”

8- It will ask you the question “Did you install myclient?”

9- It means you have to install mysqlclient inside your environment.(means project)

10- To install MysqlClient type this command

11- “pip install --only-binary :all: mysqlclient “press enter”

Run the XAMP server each and every time when you start to work on Django Software.

12- Now myclient is successfully installed

13- After this migrate your table by giving this command

14- "python manage.py migrate" "press enter"

15- After above command our table will show in our database(click on refresh button in url of browser)

16- Click on crud database and we will see our default tables name of Django application.

17- It means my Django is connected with mysql.

18- After you have a project and an app, let's create a model of which we will be creating instances through our view. In app/models.py which is already shown in "step-3".

19- Which will help in CRUD operations.

```
class Student(models.Model):  <press enter>
        Firstname=models.CharField(max_length=50)
        Lastname=models.CharField(max_length=50)
        Email=models.EmailField(max_length=50)
        Contact=models.BigIntegerField()
```

20- After this go to cmd and type this command

```
>python manage.py makemigrations
```

21- It will create your model with this message on screen "Create model Student"

Run the XAMP server each and every time when you start to work on Django Software.

22- After executing above command, we will give next command > python manage.py migrate

23- Now it will get migrated.

24- Now we will go to browser and then refresh the browser, and we see the name of model(means Table name) -> “app_student” with its attribute name.

25- Go to your app folder-> create new folder(Templates)->Again create new folder(Name of your App for easy purpose

->create 3 files under app folder with names

“insert.html “ <press enter>

“edit.html” <press enter>

“show.html” <press enter>

26- First of all, we will go to “insert.html “ file and create a form (Click and follow Step-5 to work on Django framework->>Display Forms in HTML)

27- Now we will insert data to our database through this created Form.

28- First of all, we will create new views means new function(def InsertData()) in view.py file.

Run the XAMP server each and every time when you start to work on Django Software.

29- To create new function(def InsertData()), we will type below code in views.py file again(do not delete previous code of InsertPageView definition)

```
def InsertData(request):
    #Data come from HTML to View
    fname=request.POST['fname']
    lname=request.POST['lname']
    email=request.POST['email']
    contact=request.POST['contact']
```

30- Now, we will create the object of our Model class to store our data into particular database. So write below code under views.py file again but below contact=request.POST['contact']. After pressing 2 times enter key write the below Yellow color code.(Don't write cut code in views.py file again->only under #creating Object of Model class)

```
from . models import *

def InsertData(request):
    #Data come from HTML to View
    fname=request.POST['fname']
    lname=request.POST['lname']
```

Run the XAMP server each and every time when you start to work on Django Software.

~~email=request.POST['email']~~

~~contact=request.POST['contact']~~

#Creating Object of Model Class

#Inserting Data into Table

newuser = Student.objects.create(

Firstname=fname,Lastname=lname,

Email=email,Contact=contact)

31- After type each and every code, just save by pressing ctrl+s button.

32- After inserting data just show.html page should call to show inserted data on that page.(To call show.html file we have to render this page. Ignore other coding if you have written. Only type new code in Yellow highlighted color from below code)

~~from . models import *~~

~~def InsertData(request):~~

~~#Data come from HTML to View~~

~~fname=request.POST['fname']~~

~~lname=request.POST['lname']~~

~~email=request.POST['email']~~

Run the XAMP server each and every time when you start to work on Django Software.

```
    contact=request.POST['contact']

    #Creating Object of Model Class
    #Inserting Data into Table
    newuser = Student.objects.create(
                Firstname=fname,Lastname=lname,
                Email=email,Contact=contact)

    # After Insert render on Show.html
    return render(request,"app/show.html")
```

33- After writing above code in views.py file, we will go to create new path for new def of views.py file.

34- path("insert/",views.InsertData,name="insert"),

35- We will copy the name="insert" and go to insert.html file and type in

```
<form action="{% url 'insert' %}" method="post">
{% csrf_token %}
```

36- csrf_token take responsibility to secure your data. Now press ctrl+s button to save changes.

37- Now write the below code in show.html file

```
<h1>Show Page</h1>
```

38- And save this file.

Run the XAMP server each and every time when you start to work on Django Software.

39- Now our server is on run mode, so refresh the server and insert data into form and click on insert button.

40- After clicking on insert button show.html file page will be called on browser "Show Page".

41- When we refresh our database page. Our inserted data through insert.html file will be shown in database(mysql).

B-How To Show data from database on browser in Table Form?

42- To show data from database on browser in tablular form we will use Bootstrap.

43- Bootstrap: Bootstrap is the most popular framework of javascript to make responsive page. To make responsive table we have to use Bootstrap Tables.

44- To implement this concept, we will go to show.html file in (templates->app>show.html)

45- Go to google and type "bootstrap table".

46- Go to w3school site and copy the code of bootstrap table including border and paste in show.html file.

Run the XAMP server each and every time when you start to work on Django Software.

47- And remove the code “ The .table-bordered class adds borders to a table: ” .

48- To display the output of this code on browser, we have to create new definition in views.py file.(Don’t delete previous code of this file)

```
#show Page View
def ShowPage(request):
    return render(request,”app/show.html”)
```

49- To call this definition, we have to set new path for this file in urls.py file.(don’t delete previous path).

50- path("showpage/",views.ShowPage,name="showpage"),

51- We have to save it and runserver and insert the data and this tablular form will be shown on browser.

52- Now we will edit the show.html file with this code in first <tr>.

```
<tr>
<th>Id</th>
<th>Firstname</th>
```

Run the XAMP server each and every time when you start to work on Django Software.

```
<th>Lastname</th>
<th>Email</th>
<th>Contact</th>
</tr>
```

53- We will save this file again and refresh this page and new column name 'id' will be shown.

54- The data shown in "Show Data Table" is dummy data but we have to show actual data. So, to show actual data (dynamic data) we have to edit some code in definition of ShowPage of views.py file.

55- We have to fire select query of database(select *from tablename) in Django in ShowPage(request).

56- Replace the existing code ShowPage(request) with this below code.


```
#Show Page View
def ShowPage(request):
    # select * from tablename
```

Run the XAMP server each and every time when you start to work on Django Software.

```
# For fetching all the data of the table
all_data = Student.object.all()
return render(request,"app/show.html",{'key1':all_data})
```

57- Now we will go to show.html file and edit the existing code of form below </thead>.

58- Write this line code just below like

</thead>

{% if key1 %}

59- And below </tbody> write this code.

{% endif %}

60- Now we have to run for loop because our table has multiple data keys. For multiple data, we can write multiple queries again and again for each particular data.

61- Now run(means write) 'for' loop under <tbody>

{% for i in key1 %}

62- And end this query after first </tr> just by writing

{% endfor %}

63- Now first <td> of <tr> will represent id then how will call the value. Whenever we call value we have to use double curly bracket for 'for' loop.

{{ }} (don't write it this time)

64- So replace previous all code of show.html after </thead> and before </table> with below code or

Run the XAMP server each and every time when you start to work on Django Software.

just copy and paste this code in show.html file.

```
{% if key1 %}
<tbody>
{% for i in key1 %}
<tr>
<td> {{i.id}}</td>
<td>{{i.Firstname}}</td>
<td>{{i.Lastname}}</td>
<td>{{i.Email}}</td>
<td>{{i.Contact}}</td>
</tr>
{% endfor %}
</tbody>
{% endif %}
```

65- Now save this file and run the server again and insert the data at least 1 data and this data will be inserted in database and to show this data we have to refresh the database.

66- But we are not able to see any data in table on browser because when we had created our insert views, we had only rendered our show.html page.

67- But the ShowPage in views.py which we have just created is displaying data, so as soon as our work is completed with insert view, we should call

view of show for that we have to import redirect. Replace first line of views.py file with below code

from django.shortcuts import render,redirect

68- After replacing first line of views.py file. Again, replace this below cut line from just next line code. So, see it carefully and then replace.

~~# After Insert render on Show.html~~
~~return render(request,"app/show.html")~~

After Insert render on ShowPage View
return redirect('showpage')

69- Now save this file and run the server and insert the data through form. Now show page will show inserted data in table form.

C-How to updata Data in Django

70- First of all, we will fetch(means access) the data means we will fetch the one particular user from multiple data that we have stored by selecting and then we will update the data.

71- We will fetch the data by using get() method in django.

Run the XAMP server each and every time when you start to work on Django Software.

72- get() method is used to fetch particular single data from database.

73- Now, after fetching the particular data of user, we will update the data of user.

74- We will update the data by using save() method in Django.

75- We will add 2-buttons Edit and Delete in show.html file to show these buttons on browser. Add two extra line after this ~~<th>Contact</th>~~ .

```
<th>Edit</th>
<th>Delete</th>
```

76- Save this file and refresh page again. After this Edit and Delete Column will be shown.

77- We will create new form with name 'EditForm' in previous show.html file because When we will click on Edit button then form will be called.

78- To show form write this code which is shown below. (Ignore cut code it is just given from where you have to start new form code).

Run the XAMP server each and every time when you start to work on Django Software.

79. So, replace previous all code of show.html after </thead> and before </table> with below code.

```
{% if key1 %}
<tbody>
{% for i in key1 %}
<tr>
<td> {{i.id}}</td>
<td>{{i.Firstname}}</td>
<td>{{i.Lastname}}</td>
<td>{{i.Email}}</td>
<td>{{i.Contact}}</td>
<td>
 <form name="EditForm" action=""
method="post">
{% csrf_token %}
<input type="submit" value="Edit">
</form>
</td>
<td>
 <form name="DeleteForm" action=""
method="post">
```

Run the XAMP server each and every time when you start to work on Django Software.

```
{% csrf_token %}
<input type=”submit” value=”Delete”>
</form>
</td>

</tr>
{% endfor %}
</tbody>
{% endif %}
```

80- After edit show.html page, save it and run or refresh the server if it is opened. Now Edit and Delete button will show on Show Page.

81- When we click on Edit button then particular data should get fetched and displayed.

82- For this, we will create edit.html page as already created under app folder(means app->templates->app->edit.html)

83- Now, we will copy the insert.html file code and paste in edit.html file. (It is shown in below line)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>this is form through templates</title>
</head>
<body>
    <center>
```

Run the XAMP server each and every time when you start to work on Django Software.

```
        <h1> INSERT FORM</h1>
        <form action="{% url 'insert'  %}" method="post">
            {% csrf_token %}
            <table>
                <tr>
                    <td>Firstname</td>
                    <td><input type="text" name="fname" id=""></td>
                </tr>
                <tr>
                    <td>Lastname</td>
                    <td><input type="text" name="lname" id=""></td>
                </tr>
                <tr>
                    <td>Email</td>
                    <td><input type="text" name="email" id=""></td>
                </tr>
                <tr>
                    <td>Contact</td>
                    <td><input type="tel" name="contact" id=""></td>
                </tr>
                <tr>
                    <td><input type="submit" value="Insert"></td>
                </tr>
            </table>
        </form>
    </center>
</body>
</html>
```

84- **After this we will change the heading means**

~~**<h1> INSERT FORM </h1>**~~

<h1>UPDATE FORM</h1>

85- **And also we will remove url part from the below code. Delete the cut part from the code and save it.**

<form action="~~{% url 'insert' %}~~" method="post">

86- **And also change the name of submit button from ~~Insert~~ to Update.**

It means for this copy the below code and replace

the existing code.(Do not copy the code which is cut)

~~<td><input type="submit" value="Insert"></td>~~

<td><input type="submit" value="Update"></td>

87- Now save it again. As I have already told after any change in code save it every time.

88- Now, when we click on Edit button, data should display and for that we have to render this file means edit.html file should call.

89- So, to call this file we have to create new View in views.py file. So, go to views.py file but do not delete previous code and add this below code after pressing two times enter key.

```
# Edit Page View
def EditPage(request,pk):
    #Fetching the data of Particular ID
    get_data = Student.objects.get(id=pk)
    return render(request,"app/edit.html",{'key2':get_data})
```

90- To call this View, we have go to urls.py for path. add the below path in previous code of urls.py file.

91- path("editpage/<int:pk>",views.EditPage,name="editpage")

Note: <int:pk> means we will pass id to EditPage

view though this url.

92- Now we will copy the name of url i.e. editpage and update the form code of the show.html file with below code. (don't write cut section)

~~<form name="EditForm" action="~~{% url 'editpage' pk=i.id %}" ~~method="post">~~

Note: pk={{i.id}} by this code we will get the particular id dynamically. It means when we click on Edit button of a particular id it will reach here means in show.html file

93- Now save this file again and run the server. And when we click on Edit button of any particular data from shown table then we jump to new page with new url "127.0.0.1:8000/editpage/2" with update form but the data is not displayed yet. We reached

here with data but data is not displayed we even got ID here i.e. "/2"

94- To show data on page of editpage, we will go to edit.html file with few update in existing code.(Don't write the code of cut sign)

~~ ~~

~~<h1> Update Form</h1>~~

{% if key2 %}

~~<form action="" method="post">~~

~~{% csrf_token %}~~

~~.....~~

~~.....~~

~~</table>~~

~~</form>~~

{% endif %}

95- In this edit.html file we don't need to use 'for' loop because we are fetching data of single user.

96- So Django gives you the facility to fetch data though key but how? We will see through some update again in edit.html file. So, add this code in existing code of edit.html file. (Don't write the code of cut sign)

~~<td><input type="text" name="fname" id=""~~

value="{{key2.Firstname}}"~~></td>~~

Run the XAMP server each and every time when you start to work on Django Software.

~~<td><input type="text" name="fname" id=""~~
value="{{key2.Lastname}}"~~></td>~~

~~<td><input type="text" name="fname" id=""~~
value="{{key2.Email}}"~~></td>~~

~~<td><input type="text" name="fname" id=""~~
value="{{key2.Contact}}"~~></td>~~

97- Now, save this file again and run the server if closed or refresh if already open. And if click on Edit button then Update Form will show the data.

Note: After every update in any code of any file just save that file and also views.py and urls.py file also.

98- Now data is fetched and we want to update this data just by clicking on 'Update' button. So to update the existing data we will create new definition by updating views.py file. (don't delete the previous code of views.py file).
Write the below code carefully in views.py file.

Run the XAMP server each and every time when you start to work on Django Software.

```
# Update Data value
def UpdateData(request,pk):
    udata = Student.objects.get(id=pk)
    udata.Firstname = request.POST['fname']
    udata.Lastname = request.POST['lname']
    udata.Email = request.POST['email']
    udata.Contact = request.POST['contact']
```

Note: In above code Firstname, Lastname, Email, Contact are the columns name which means names of the fields of the database.
And fname, lname, email, contact are coming from edit.html file.

99- Now we will run query for update in same code. And we know to update the data we use the save() method in Django. (Ignore the code which is cut)

~~# Update Data value~~
~~def UpdateData(request,pk):~~
~~udata = Student.objects.get(id=pk)~~
~~udata.Firstname = request.POST['fname']~~
~~udata.Lastname = request.POST['lname']~~
~~udata.Email = request.POST['email']~~
~~udata.Contact = request.POST['contact']~~

Run the XAMP server each and every time when you start to work on Django Software.

```
# Query for update
udata.save()
```

100- And after updating above code which page will open, our show.html page will open which means update is update but needs to get displayed.

So, to show the showpage we will update the code in views.py file. So see it carefully and ignore the code which is cut.

~~# Update Data value~~
~~def UpdateData(request,pk):~~
~~udata = Student.objects.get(id=pk)~~
~~udata.Firstname = request.POST['fname']~~

Run the XAMP server each and every time when you start to work on Django Software.

~~udata.Lastname = request.POST['lname']~~

~~udata.Email = request.POST['email']~~

~~udata.Contact = request.POST['contact']~~

~~# Query for update~~

~~udata.save()~~

Render to show page

return redirect('showpage')

Note: Save the code every time after updating the code. And also save the views.py and the urls.py files.

101- Now to call the UpdateData view we will set the path by just clicking on urls.py file and add the new path. See the below code (don't delete the previous path)

path("update/<int:pk>",views.UpdateData,name="update"),

102- Now copy the name of url means 'update' and paste this name of url in edit.html file in action section. (We will update the edit.html file by below code and don't write the which is cut)

Run the XAMP server each and every time when you start to work on Django Software.

<form action="{% url 'update' pk=key2.id %}"
method ="post">

103- Now if we refresh the server or run the Django server and after filling the form and clicking on insert button 'Show Data Table' will show and if we click on Edit button to update the existing data then updated data of particular id will show with updated value of any Column.

104- Q. How to Delete the Data in Django after clicking on Delete Button?

Ans: To delete the data, we will use delete() method of ORM(Object Relational Management)

So to implement this concept we will write new View in views.py file. (Don't delete the previous data)

```
# Delete Data View
def DeleteData(request,pk):
    ddata = Student.objects.get(id=pk)
    # Query for Delete
```

Run the XAMP server each and every time when you start to work on Django Software.

ddata.delete()

render redirect('showpage')

105- Now save the above code and now set the path in urls.py file of app. (Don't delete the previous code)

path("delete/<int:pk>",views.DeleteData,name="delete")

106- Now copy the name of url means "delete" and paste it in show.html file through below code. (don't write the code which is cut)

~~<form action="DeleteForm" action=~~"{% url 'delete' pk=i.id %}" ~~method = "post"~~>

107- Now save this code and the views.py and the urls.py code also.

108- Now run the server through refresh or run again the server of Django.

Run the XAMP server each and every time when you start to work on Django Software.

109- **In above section all sections we have worked on DJANGO ADMIN PANEL, DJANGO HTML AND DATABASE CONNECTIVITY AND DJANGO CRUD APPLICATONS.**

Run the XAMP server each and every time when you start to work on Django Software.

BUILDING A REGISTRATION TEMPLATE IN DJANGO TO WORK ON PROJECT

Q. What are the benefits of using a Django registration template?

Ans. Django registration templates are a new way of thinking about how to manage user registration in Django projects. They are simple, reusable, and easy to implement. There are many benefits of using Django registration templates- Easy implementation, These templates can be used in multiple projects, We can protect our users from malicious attacks.

Note: If we register to any website then we need to take SIGN UP Template, LOGIN Template etc…

Q- How does the Registration Template work?

Ans: We will use CSS first to create Form in better look. We don't need to write CSS code because we are going to integrate means will use pre-define code from website.

So first of all we will download the template First and then integrate with Django Project. We will use the concept External CSS in our project.

Run the XAMP server each and every time when you start to work on Django Software.

1- To work on Registration template(Take the Help form Step-1 To Step-5 from above pages including CRUD Applications Concepts also)->
   i. create a folder with suitable name(i.e. 'Registration Template Integration') and install virtual environment inside that folder. And create project with 'project' name and and create app with name 'app' and install 'app' inside settings.py of the 'project' . Then we have to go to 'urls.py' file of 'project' and import 'include' here because we have to send our request directly in our application i.e. in 'app'.
2- Copy the code of urls.py file of 'project'.
3- Now we will go to our 'app' and create a new file urls.py, paste the code of urls.py file of 'project'.
4- Delete the code inside of path. Our project is complete and integrated.
5- Now, we have made our 'project', we have installed our 'app' , we have linked our urls.py. We have done everything.
6- Now, we are going to integrate our template, for that we have create a 'template' folder, after that we have to make an 'app' folder and then create the under the 'app' folder(register.html file)

Run the XAMP server each and every time when you start to work on Django Software.

7- Before doing anything, we will go to browser and type "registration template bootstrap" and go to colorib.com and find appropriate design (to make the responsive page) then click on download button.

8- After clicking on download button, we get HTML and CSS code both.

9- So, to integrate the bootstrap form from website, we will copy the 'html code' from above step-8 and paste in already created file 'register.html' file.

10- To see the look of register page without CSS. We will create a view under 'views.py' file of our 'app'.
(From here save every code after either new typing or editing the previous code -> To save press ctrl + s button)

```
from django.shortcuts import render

# Create your views here.

# View for Register Page
def RegisterPage(request):
    return render(request,"app/register.html")
```

Run the XAMP server each and every time when you start to work on Django Software.

11- After it we will make new url. (Edit the below code in urls.py file of our 'app')

```
from django.urls import path,include
from . import views

urlpatterns = [
path("",views.RegisterPage,name="registerpage"),
]
```

12- Now, We will run the server but before run the server first time, we have to migrate the default table of Django by giving below command.

```
(myenv) D:\Registration Template Integration> python manage.py migrate  "press enter"
```

13- Now run the server after type below code

```
(myenv) D:\Registration Template Integration> python manage.py runserver "press enter"
```

14- Ans we see the Registration page without CSS.

15- Now to page responsive, we have to integrate CSS code by copy the Bootstrap CSS code from above link

Run the XAMP server each and every time when you start to work on Django Software.

from where we had copied the html code for register.html file.

16- Now, create a new folder ‘static’ (It is compulsory to give name ‘static’ to integrate Bootstrap file) inside our ‘app’ means ‘app’ -> ‘static’ and then create a file with name ‘style.css’ and inside this file paste the code of CSS Bootstrap.
‘app’ -> ‘static’ -> ‘style.css’ -> Paste the code of CSS from above website ‘colorib.com’.

~~17-~~ Go to register.html file to change the link of External CSS File which is existing inside ‘static’ -> ‘style.css’ file for this copy this code in existing code. (Ignore the code which is cut)

~~<!--~~

~~Author: Colorlib~~

~~Author URL: https://colorlib.com~~

~~License: Creative Commons Attribution 3.0 Unported~~

~~License URL:~~
~~http://creativecommons.org/licenses/by/3.0/~~

~~-->~~

Run the XAMP server each and every time when you start to work on Django Software.

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
<title>SignUp Form</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="application/x-javascript">
addEventListener("load", function() {
setTimeout(hideURLbar, 0); }, false); function
hideURLbar(){ window.scrollTo(0,1); } </script>
<!-- Custom Theme files -->
<link href="{% static '/css/style.css' %}"
rel="stylesheet" type="text/css" media="all" />
<!-- //Custom Theme files -->
```

18- This is how our CSS file will be linked.

19- Now refresh the page or run the django server again and we will get Responsive Web Page.

Run the XAMP server each and every time when you start to work on Django Software.

20- Register user with server side validation (Follow below steps with above code)

21- Now I am going to do some modification in above Registration template(means in register.html file)

22- So, Update the code of register.html file. Delete rest code of the register.html file if not matching with below code.

```
(47)  {% load static %}
(48)  <!DOCTYPE html>
(49)  <html>
(50)  <head>
(51)  <title>SignUp Form</title>
(52)  <meta name="viewport" content="width=device-width, initial-scale=1">
(53)  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
(54)  <script type="application/x-javascript"> addEventListener("load", function() {
      setTimeout(hideURLbar, 0); }, false); function hideURLbar(){
      window.scrollTo(0,1); } </script>
(55)  <!-- Custom Theme files -->
(56)  <link href="{% static '/css/style.css' %}" rel="stylesheet" type="text/css"
      media="all" />
(57)  <!-- //Custom Theme files -->
(58)  <!-- web font -->
(59)  <link href="//fonts.googleapis.com/css?family=Roboto:300,300i,400,400i,700,700i"
      rel="stylesheet">
(60)  <!-- //web font -->
(61)  </head>
(62)  <body>
(63)      <!-- main -->
(64)      <div class="main-w3layouts wrapper">
(65)          <h1>SignUp Form</h1>
(66)  <div class="main-agileinfo">
(67)              <div class="agileits-top">
(68)                  <form action="#" method="post">
(69)
(70)                      <input class="text email" type="text" name="fname"
      placeholder="Firstname" required="">
(71)                      <input class="text" type="text" name="lname"
      placeholder="Lastname" required="">
```

Run the XAMP server each and every time when you start to work on Django Software.

```
(72)                    <input class="text email" type="email" name="email"
      placeholder="Email" required="">
(73)                    <input class="text email" type="text" name="contact"
      placeholder="Contact" required="">
(74)                    <input class="text" type="password" name="password"
      placeholder="Password" required="">
(75)                    <input class="text w3lpass" type="password" name="cpassword"
      placeholder="Confirm Password" required="">
(76)                    <input type="submit" value="SIGNUP">
(77)                </form>
(78)                <p>Don't have an Account? <a href="#"> Login Now!</a></p>
(79)            </div>
(80)        </div>
```

23- Now save this register.html file and refresh the page or run the django server. And we will see beautiful Registration page.

24- Now, I have to make the user registered through this particular form and to check that user is valid or not. So how we do it just. So first of all we will create new view in views.py file that will register our user. But before that we will create Models (means class) in models.py file. So models will be created according to template means according to input fields of register.py form means in models I will include Firstname, Lastname, Email, Password.

Run the XAMP server each and every time when you start to work on Django Software.

25- Open models.py file under our 'app' and write the below code. (After writing every code just save it and then save urls.py file and then views.py file also)

```
from django.db import models


# Create your models here.
class User(models.Model):
    Firstname = models.CharField(max_length=50)
    Lastname = models.CharField(max_length=50)
    Email = models.EmailField(max_length=50)
    Contact = models.CharField(max_length=50)
    Password = models.CharField(max_length=50)
```

26- Now we have to integrate this model with mysql so we have to install mysqlclient through given below command.

```
(myenv) D:\Django-Complete\D Building a Registration Template Integration in Django to work on Project>pip install –only-binary :all: mysqlclient  "press enter key"
```

27- After installation of mysqlclient give command for makemigrations.

Run the XAMP server each and every time when you start to work on Django Software.

(myenv) D:\Django-Complete\D Building a Registration Template Integration in Django to work on Project>python manage.py makemigrations “press enter key”

28- After executing above code, User model will be created.

29- Before migrating our model, we will create DATABASE in xamp Server with the help of clicking on admin button.

I will give database name ‘register’.

30- And click on create button.

31- After creating database, go to settings.py file of project and search

```
DATABASES= {
  ‘default’ :
    {
    }
}
```

Replace the default from below code:

```
DATABASES = {
 'default':
```

Run the XAMP server each and every time when you start to work on Django Software.

```
    {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'crud',   #Your database name
        'HOST': 'localhost', #Your localhost name
        'PORT': '3306',
        'USER': 'root',   # Your Database Username
        'PASSWORD': '', # YourDatabaseUser Paswrd
    }
}
```

3- Go to cmd from where you run your project. (Do not run)

4- Now type this command “python manage.py migrate” “press enter”

5- It will ask you the question “Did you install myclient?”

6- It means you have to install mysqlclient inside your environment.(means project)

7- To install MysqlClient type this command

30- Now give command to migrate the above model.

Run the XAMP server each and every time when you start to work on Django Software.

(myenv) D:\Django-Complete\D Building a Registration Template Integration in Django to work on Project>python manage.py migrate “press enter”

31. After migrations process, we will register our Models in admin.py file of app.

32.Type the below command to register Models classes:

from .models import *

admin.site.register(User)

33.Again run the server, go to admin panel and we will see the name of Models class ‘User’.

34.Now we can add declared data in ‘User’ Table.

35- Now we go to views.py file again to create new view for User registration. (Ignore the code which is already cut)

```
from django.shortcuts import render
from . models import *
# Create your views here.

# View for Register Page
```

Run the XAMP server each and every time when you start to work on Django Software.

```
def RegisterPage(request):
    return render(request,"app/register.html")

# View for user registration
def UserRegister(request):
    if request.method == "POST":
        fname = request.POST['fname']
        lname = request.POST['lname']
        email = request.POST['email']
        contact = request.POST['contact']
        password = request.POST['password']
        cpassword = request.POST['cpassword']

        # First we will validate that user already exist
        user =User.objects.filter(Email=email)
        # first 'Email' is from Table Field and Second 'email' is from above view
        # if user already exist then we will pass a message through below code
        if user:
            message = "User already exist"
            return render(request, "app/register.html",{'msg':message})
        else:
            # This section is the server side validation of password
            if password == cpassword:
                newuser = User.objects.create(Firstname=fname,
                          Lastname=lname,Email=email,Contact=contact,Password=password)
                message = "User register Successfully"
                return render(request,"app/login.html",{'msg':message})
            else:
                message = "Password and Confirm Password doesnot Match"
                return render(request, "app/register.html",{'msg':message})
```

36- Now create new url path under urls.py file of our 'app' . Write the below code for set path.

```
from django.urls import path,include
from . import views

urlpatterns = [
    path("",views.RegisterPage,name="registerpage"),
    path("register/",views.UserRegister,name="register"),
    ]
```

Run the XAMP server each and every time when you start to work on Django Software.

37- Now pass view name i.e. ‘register’ in <form action=…..> (update register.html file with given code, ignore the code which is cut)

```
<div class="main-w3layouts wrapper">
        <h1>SignUp Form</h1>
        <div class="main-agileinfo">
            <div class="agileits-top">
                <form action="{% url 'register' %}" method="post">
                    {% csrf_token %}
```

38- Now if we want to show the messages of server side validation that means sent through created view. This message that we have sent also need to be show here means register.html page. So, for this update the above register.html file. (Ignore the code written in cut form)

```
<div class="main-w3layouts wrapper">
        <h1>SignUp Form</h1>
        <center><h3 style="color: red;">{{msg}}</h3></center>
        <div class="main-agileinfo">
            <div class="agileits-top">
                <form action="{% url 'register' %}" method="post">
                    {% csrf_token %}
```

39- Now create new template ‘login.html’ (under ‘app’->’template’->’app’->’login.html’

40- Now type this code in login.html page to diaplsy this message <h1>Login Page </h1>.

41- Now run the django server and Fill the asked information in Form and then click on SIGNUP button and we get message ‘Login Page’ on new page on browser. To

Run the XAMP server each and every time when you start to work on Django Software.

42- Now, refresh the Php admin page of xamp server and we see the entered data is displayed with password field also in mysql database.

User Login with Server-Side validation in Django

43- In last means (42) register the user, but in this topic, we will give the permission to Login the registered user means we give permission to access.

Q-What does User Login means?

Ans- User Login is Used To manage The User Profile.

Q- How does server-side validation work?

Ans- In Server-Side Validation, the user's input is transferred to the server and validation using one of the server-side programming languages such as ASP.net, PHP, and others. Following the server-side Validation, the feedback is given to the client via a new dynamically produced web page.

Q- What is the purpose of using client-side validation Vs Server-side validation?

Ans- Using both is the best solution. The gateway into the rest of the system, server-side validation, treats all

Run the XAMP server each and every time when you start to work on Django Software.

incoming input as untrusted. Client-side validation aims to make the user's experience as smooth as possible while also reducing server burden.

44- Before working on this topic we will start our xamp server. Now click on admin button of php mysql server on xamp. And we will take any data from user which is 'register' and will login from login page of Django.

45- We will copy the whole code of register.html file and paste in login.html file.

46- We will update this pasted code of login.html file. (See the code below and delete extra code from login.html file)

```
<!--
Author: Colorlib
Author URL: https://colorlib.com
License: Creative Commons Attribution 3.0 Unported
License URL: http://creativecommons.org/licenses/by/3.0/
-->
{% load static %}
<!DOCTYPE html>
<html>
<head>
<title>SignUp Form</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="application/x-javascript"> addEventListener("load", function() {
setTimeout(hideURLbar, 0); }, false); function hideURLbar(){ window.scrollTo(0,1); }
</script>
<!-- Custom Theme files -->
<link href="{% static '/css/style.css' %}" rel="stylesheet" type="text/css" media="all" />
<!-- //Custom Theme files -->
```

Run the XAMP server each and every time when you start to work on Django Software.

```
<!-- web font -->
<link href="//fonts.googleapis.com/css?family=Roboto:300,300i,400,400i,700,700i"
rel="stylesheet">
<!-- //web font -->
</head>
<body>
    <!-- main -->
    <div class="main-w3layouts wrapper">
        <h1>SignIn Form</h1>
        <center><h3 style="color: red;">{{msg}}</h3></center>
        <div class="main-agileinfo">
            <div class="agileits-top">
                <form action="{% url 'register' %}" method="post">
                    {% csrf_token %}

                    <input class="text email" type="email" name="email"
placeholder="Email" required="">
                    <input class="text" type="password" name="password"
placeholder="Password" required="">
                    <input type="submit" value="SIGNIN">
                </form>
                <p>Don't have an Account? <a href="#"> Login Now!</a></p>
            </div>
        </div>
        <ul class="colorlib-bubbles">
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
        </ul>
    </div>
    <!-- //main -->
</body>
</html>
```

47- Now don’t have an account, we will run our login page and for that we will to create a new view in views.py file (See the below code and write in views.py file and don’t delete the previous code)

Run the XAMP server each and every time when you start to work on Django Software.

Creating LogInPage() View

```
def LoginPage(request):
    return render(request,"app/login.html")
```

48- Now, we have to make url of this above view by adding this below code in urls.py file of our 'app'.

```
path("loginpage/",views.LoginPage,name="loginpage"),
```

49- Now, we will run django server by command python manage.py runserver and click on the given link and will call the login page from url (like 127.0.0.1:8000/loginpage) and press enter key.

50. Now, after pressing the enter key our SIGNIN FORM page is ready. But right now, we only have a SIGNIN FORM we don't have permission to Login to existing user.

51- For login to existing user, we will create a new view in views.py file of our 'app'. (Add the below code in views.py file and don't delete the previous code)

```
# Creating LoginUser() view
def LoginUser(request):
    if request.method == "POST":
        email = request.POST['email']
        password = request.POST['password']

        #Checking the emailid with database
        user = User.objects.get(Email=email)

        if user:
```

Run the XAMP server each and every time when you start to work on Django Software.

```
            if user.Password == password:
                # We are getting user data in session
                request.session['Firstname'] = user.Firstname
                request.session['Lastname'] = user.Lastname
                request.session['Email'] = user.Email
                return render(request,"app/home.html")
            else:
                message = "Password does not match"
                return render(request,"app/login.html",{'msg':message})
        else:
            message = "User does not exist"
            return render(request, "app/register.html",{'msg':message})
```

52- Now, to call this view, we have to set new path in urls.py file of our 'app'. So update the code not delete the previous code.

```
path("loginuser/",views.LoginUser,name="login"),
```

53- Now update the login.html file with below code. (Ignore the code which is already cut and save every code after writing)

```
<h1>SignIn Form</h1>
        <center><h3 style="color: red;">{{msg}}</h3></center>
        <div class="main-agileinfo">
            <div class="agileits-top">
                <form action="{% url 'login' %}" method="post">
                    {% csrf_token %}
```

54- Now, we will create new file home.html file in our created templates means go to (app->template->app->home.html)

Run the XAMP server each and every time when you start to work on Django Software.

55- Now, we just want to show some information on webpage like Firstname, Lastname, and Emailid of the valid user which has just login by writing some code in home.html file. So, write the below code in home.html file.(don't forget to save the any updated code and then views.py and then urls.py file also of our 'app')

```
<center>
<h1>Home Page</h1>

Firstname : {{request.session.Firstname}}
<br>
Lastname : {{request.session.Lastname}}
<br>
Email : {{request.session.Email}}
</center>
```

56- Now, we will again call the login page or refresh the django server and will call the login page from url of browser by typing the 127.0.0.1:8000/loginpage

And enter the valid user email id and password and after that click on SIGNIN button and home.html page output will be shown on browser.

Run the XAMP server each and every time when you start to work on Django Software.

Image Uploading Using Django And How to Display the Image on Browser that has been already Uploaded on our Database

1- To upload the image and store the image on database, we will use 'pillow' library.

2- To Upload the image, we have to use models.imageField() method which is used to create a file input and submit the file to the server. While working with files, make sure the HTML form tag contains enctype="multipart/form-data" property.

For Image Uploading

->The very first step is to add below code in the settings.py file.(Don't write the code which is cut because this code which is cut is used here for reference(the exact location) of the code which is to write.

```
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True
```

Run the XAMP server each and every time when you start to work on Django Software.

```
USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/
import os

STATIC_URL = '/static/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL ='/media/'
```

➔ In above code MEDIA_ROOT is for server path to store files in the computer.

➔ MEDIA_URL is the reference URL for browser to access the files over Http.

To implement the IMAGE UPLOADING CONCEPT follow the given instructions below-

1-Create a new folder in 'VS Code' with any name in my case I have taken 'Image Uploading with Django'.

2-And In VS Code Go to terminal and choose the command prompt.

3- D:\Image Uploading with Django>pip install virtualenv "press enter"

4- D:\Image Uploading with Django>virtualenv myenv "press enter"

Run the XAMP server each and every time when you start to work on Django Software.

5-D:\Image Uploading with Django> myenv\Scripts\activate “press enter”

6- D:\ImageUploadingwithDjango> pip install django “press enter”

7- D:\ImageUploadingwithDjango>django-admin startproject project . “press enter”

8- D:\ImageUploadingwithDjango>python manage.py startapp app “press enter”

9- Now install the ‘app’ in settings.py file of ‘project’ folder in installed app.(For reference see the starting few page of this notes)

10- Go to urls.py file of ‘project’ and update the urls.py file and save it.

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

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

11- Now we will create new file urls.py file in created ‘app’ Write the below code in urls.py file

```
from django.urls import path,include

urlpatterns = [
]
```

12- Now, We have to create a ‘templates’ folder in our created ‘app’ and again create a new folder inside

Run the XAMP server each and every time when you start to work on Django Software.

‘template’ folder with name ‘app’ and inside ‘app’ folder create a new file name ‘index.html’ i.e. app->templates->app->index.html

13- Write the below code in index.html file

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <center>
        <h1>Image Uploading Form</h1>
        <form action="" method="post">
            <table>
                <tr>
                    <td>Image Name<input type="text" name="imgname" id=""></td>
                </tr>
                <tr>
                    <td>Upload Image<input type="file" name="image" id=""></td>
                </tr>
                <tr>
                    <td>
                        <input type="submit" value="Upload">
                    </td>
                </tr>
            </table>
        </form>
    </center>
</body>
</html>
```

14- Now we will write the view to call this index.html file in views.py file.

```
from django.shortcuts import render

# Create your views here

# Index File View
def IndexPage(request):
```

Run the XAMP server each and every time when you start to work on Django Software.

```
    return render(request,"app/index.html")
```

15- Now we go to urls.py file of 'app' folder and set the path to call Indexpage view from views.py file.

```
from django.urls import path,include
from . import views
urlpatterns = [
       path("",views.IndexPage,name="index"),
```

16- Now we will migrate our python manage.py file and then run the django server

D:\ImageUploadingwithDjango>python manage.py migrate "press enter"

17- D:\ImageUploadingwithDjango>python manage.py runserver "press enter"

18- Now we want to store the form-data on the server-side so for that we have to go to xamp-server and then click on start button against Apache and MySQL server and then click on Admin button of MySQL server to go on phpMyAdmin.

19- Now, We will create a new database with name 'djangoimg' and click on create button.

20- After this we will configure this phpMyAdmin Server database in Django App.

21- Now we will go to settings.py file and copy this code. (Ignore the code which is cut)

```
# Database
```

Run the XAMP server each and every time when you start to work on Django Software.

```
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
'default':
    {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'djangoimg',    #Your database name
        'HOST': 'localhost', #Your localhost name
        'PORT': '3306',
        'USER': 'root',    # Your Database Username
        'PASSWORD': '', # Your Database User Password
    }
}


# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators
```

22- Now, after this we have to install mysqlclients in our django.

D:\ImageUploadingwithDjango>

pip install - -only- -binary :all: mysqlclients "press enter"

23- Now we will migrate it to upload the default models of django in phpMyAdmin server database

D:\ImageUploadingwithDjango>python manage.py migrate "press enter"

24- And we will go to phpMyAdmin Server and when we click on 'djangoimg' database then our default tables name is shown.

25- Now we have to make model in models.py file of our create 'app'. So models.py file and write this code

```
from django.db import models
```

Run the XAMP server each and every time when you start to work on Django Software.

```
# Create your models here.

class ImageData(models.Model):
    Imagename = models.CharField(max_length=50)
    Image = models.ImageField(upload_to="img/")
```

26-

Now apply 'makemigratons' on django.

D:\ImageUploadingwithDjango>python manage.py makemigrations "press enter"

27- When we apply makemigrations then we get error that we have not install 'pillow' because we are going to upload image on database. So type the below command

D:\ImageUploadingwithDjango>pip install pillow "press enter"

28- Now apply makemigrations again.

D:\ImageUploadingwithDjango>python manage.py makemigrations "press enter"

29- Now we will apply migrations through 'migrate' command.

D:\ImageUploadingwithDjango>python manage.py migrate "press enter"

Run the XAMP server each and every time when you start to work on Django Software.

30- Now we will refresh phpMyAdmin server database and when we click on 'djangoimg' database then we show our model name in the form of table name 'imagedata' and its attribute in the form of column name i.e. id, imagename, image

31- Now, after migrations we have to insert data(means image) in the display page of form. It will insert our data in database. For this we have to create a new view in our existing views.py file. (Don't delete the previous code of this file)

```
def UploadImage(request):

    if request.method=="POST":
        imagename = request.POST['imgname']
        pics = request.FILES['image']

        newimg = ImageData.objects.create(Imagename=imagename,Image=pics)
        return render(request,"app/show.html")
```

32- Now we have to create a new path to call this new view in urls.py file of our created 'app'.

```
from django.urls import path,include
from . import views
urlpatterns = [
        path("",views.IndexPage,name="index"),
        path("upload/",views.UploadImage,name="imageupload"),
```

Run the XAMP server each and every time when you start to work on Django Software.

Now we copy the name ‘imageupload’ from above path and paste in index.html file’s form action section. (Ignore the code which is cut)

```
<center>

        <h1>Image Uploading Form</h1>
        <form action="{% url 'imageupload' %}" method="post" enctype="multipart/form-data">
            {% csrf_token %}
```

33- Now after inserting the image in form (means entering data in index.html file page) when we click on upload button then show.html file should call. So for this we have to create a new file show.html file under our created ‘app’. Follow below structure to create show.html file.

app->templates->app->show.html

34- Now I will write only single line code in this show.html file.

<h1>Show Page</h1>

35- Now we will run our server

D:\ImageUploadingwithDjango>python manage.py runserver “press enter”

36- Now click on given link in the same page and enter the data(means image) in the form and click on upload and show.html file will be called and again go to phpAdmin server database and we can see thata the image is uploaded.

Run the XAMP server each and every time when you start to work on Django Software.

Display Image in Django

37- Now, we just want to fetch the data first, to display the image on our browser which we have saved in database in last step.

So, To implement display image concept, we have to do some changes in settings.py file. See the code carefully to find the exact location inside the settings.py file (ignore the code which is cut)

```
TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/
import os

STATIC_URL = '/static/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL ='/media/'

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

To call this above settings, we have to go to urls.py file of our created project with name 'project' (means project->urls.py file) and update this urls.py file. (Ignore the code which is cut)

```
from django.contrib import admin
```

Run the XAMP server each and every time when you start to work on Django Software.

```
from django.urls import path,include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path("",include('app.urls')),
]
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

38- Now, Question is arisen that why have we written this code in urls.py file of 'project' because this code helps only our image will come after fetching means MEDIA_URL, and MEDIA_ROOT that we have kept in settings.py file this both things will be called at time of DEBUGING and this will get call only during DEBUG then only our image will get fetched.

39- After this go to views.py file and create new view. Don't delete the previous code. (ignore the code which is cut)

```
from django.shortcuts import render,redirect
from . models import *

# Create your views here



# Index File View
def IndexPage(request):
    return render(request,"app/index.html")
```

Run the XAMP server each and every time when you start to work on Django Software.

```
# Upload Image View
def UploadImage(request):
    if request.method=="POST":
        imagename = request.POST['imgname']
        pics = request.FILES['image']

        newimg = ImageData.objects.create(Imagename=imagename,Image=pics)
        return redirect('show')

# Image Fetching View
def ImageFetch(request):
    all_img = ImageData.objects.all()
    return render(request,"app/show.html",{'key1':all_img})
```

Now we have sent our data through ‘key1’. So in show.html file this ‘key1’ should fetch also. To fetch ‘key1’, we have to update show.htm file. Don’t delete previous code of show.html file and also write this below code in show.html again.

```
<h1>Show Page </h1>

{% if key1 %}
{% for i in key1 %}
Image Name: {{i.Imagename}}
<br>
<img src="{{MEDIA_URL}}{{i.Image.url}}" alt="" srcset="">

{% endfor %}
{% endif %}
```

Now save all above code again and again of files (means urls.py file, views.py file of our create ‘app’ and urls.py file of our created project ‘project’ also. Then run the server and enter the data and click on upload button and see the

Run the XAMP server each and every time when you start to work on Django Software.

image is saved in database or not.

```
D:\ImageUploadingwithDjango>python manage.py runserver “press enter”
```