Contents

Django Mail SetUp

Sending email using Django is pretty easy and requires less configuration. In this tutorial, we will send email to provided email.

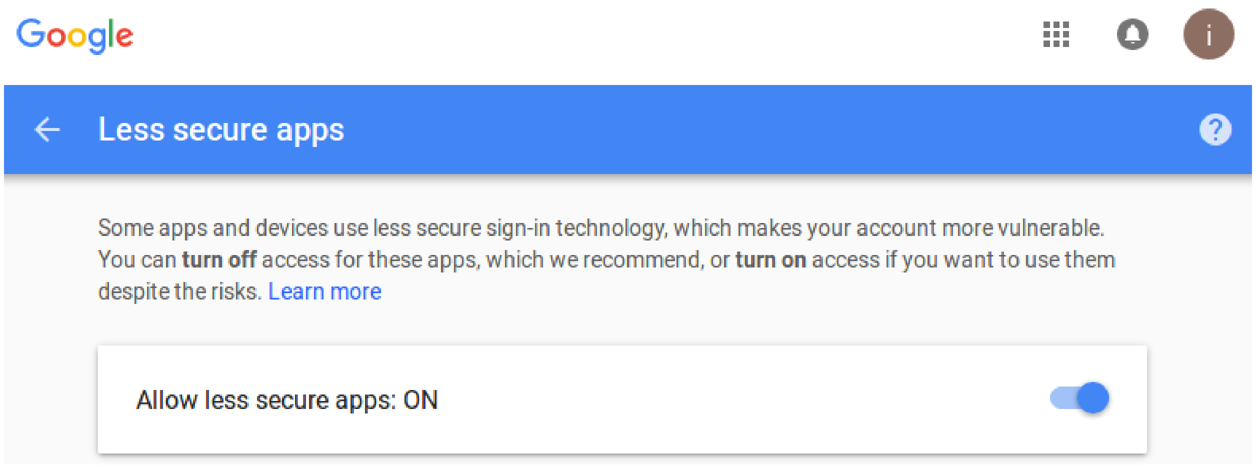
For this purpose, we will use Google's SMTP and a Gmail account to set the sender.

Django provides a built-in mail library **django.core.mail** to send email.

Before sending an email, we need to make some changes in our Gmail account because for security reasons Google does not allow direct access (login) by any application. So, login to the Gmail account and follow the urls. It will redirect to the Gmail account settings where we need to allow less secure apps but toggle the button. See the below screenshot.

6.1 Enable Less Secure Apps Option in Google Account Settings

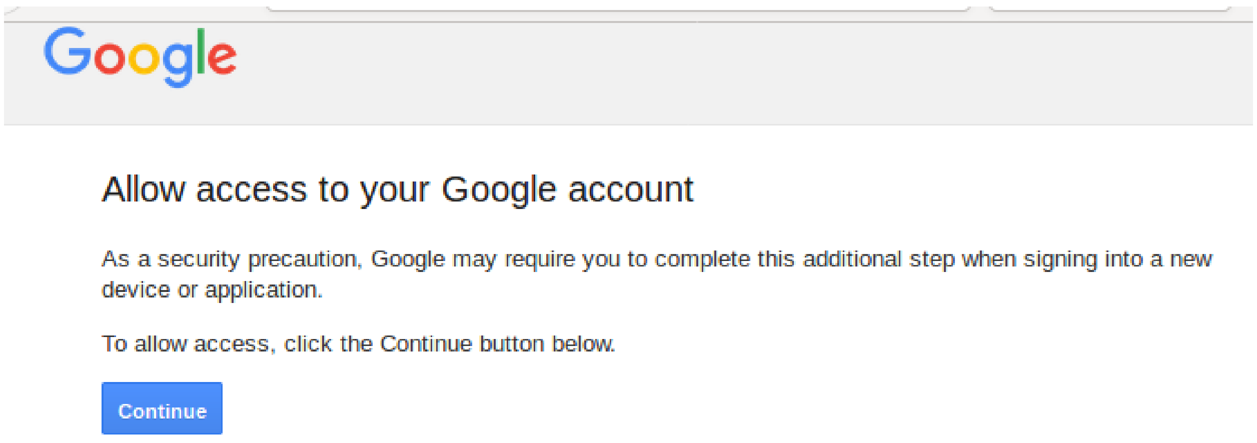
- Google Account Settings
  - Security
    - Turn on Less secure app access



Some apps and devices use less secure sign-in technology, which makes your account more vulnerable. You can **turn off** access for these apps, which we recommend, or **turn on** access if you want to use them despite the risks. Learn more

Allow less secure apps: ON

After that follow this url that is a additional security check to verify the make security constraint


Allow access to your Google account

As a security precaution, Google may require you to complete this additional step when signing into a new device or application.

To allow access, click the Continue button below.

Click on continue and all is setup

Note : After completion of email sending turn off less secure apps option

6.2 Django Configuration

- EMAIL_USE_TLS to be enabled for security reasons. TLS (Transport Layer Security) encrypts all the SMTP commands.
- EMAIL_HOST is for mentioning which server is to be connected i.e server location.
- EMAIL_PORT for giving port number of the gmail server i.e 587
- EMAIL_HOST_USER and EMAIL_HOST_PASSWORD are from which user email id is going to be send and it's password

```
EMAIL_USE_TLS = True
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = 'sender_username@gmail.com' # Your gmail ID
EMAIL_HOST_PASSWORD = '********' # Password of your gmail account
```

Import Library

```
from django.core.mail import send_mail
```

Now, write a view function that uses a built-in mail function to send mail. See the example

6.3 Sample Example For Mail Sending

This example contains the following files

// views.py

```
from django.http import HttpResponse
from djangpapp import settings
from django.core.mail import send_mail



def mail(request):
    subject = "Greetings"
    msg     = "Congratulations for your success"
    to      = "irfan.sssit@gmail.com"
    res     = send_mail(subject, msg, settings.EMAIL_HOST_USER, [to])
    if(res == 1):
        msg = "Mail Sent Successfuly"
    else:
        msg = "Mail could not sent"
    return HttpResponse(msg)
```

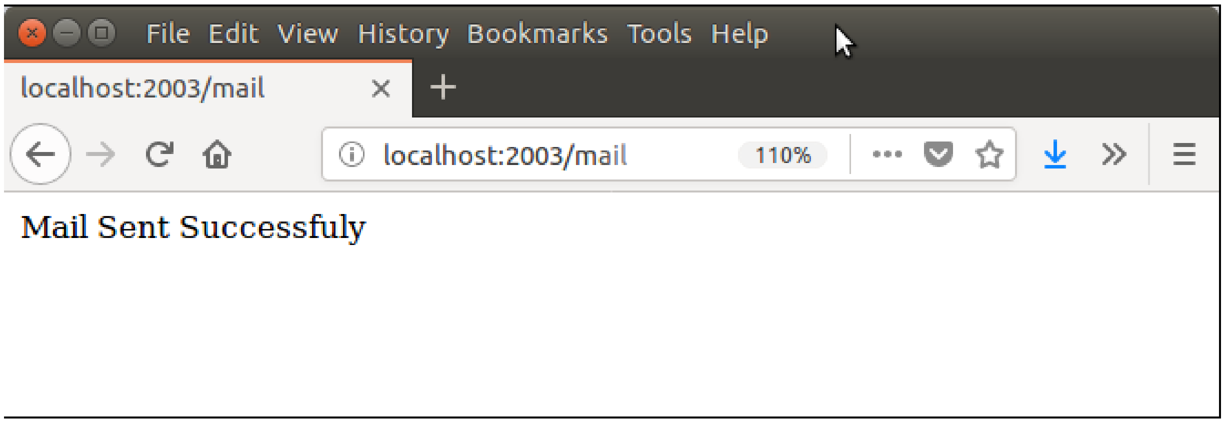
// urls.py

Put the following url into the urls.py file.

```
path('mail',views.mail)
```

Run Server and access it in the browser, see the output.

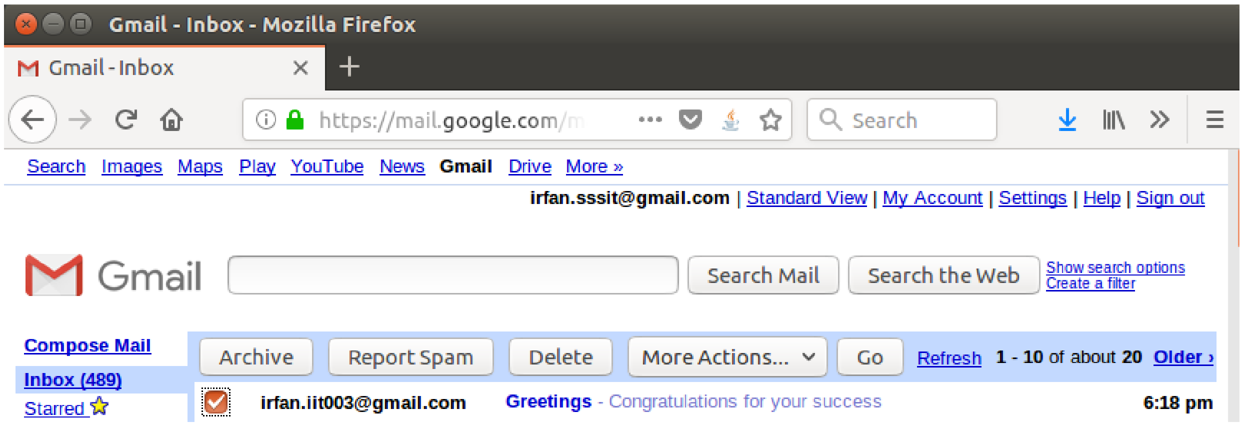
Here, both email



Mail Sent Successfuly

Here, both email ids are mine, so I can verify the email by login to the account.

And after login, here we go!! I got the mail.


Gmail - Inbox - Mozilla Firefox
Gmail - Inbox
https://mail.google.com/m
Search
Search Images Maps Play YouTube News Gmail Drive More »
irfan.sssit@gmail.com | Standard View | My Account | Settings | Help | Sign out
Gmail
Search Mail
Search the Web
Show search options
Create a filter
Compose Mail
Inbox (489)
Starred
Archive
Report Spam
Delete
More Actions...
Go
Refresh
1 - 10 of about 20
Older ›

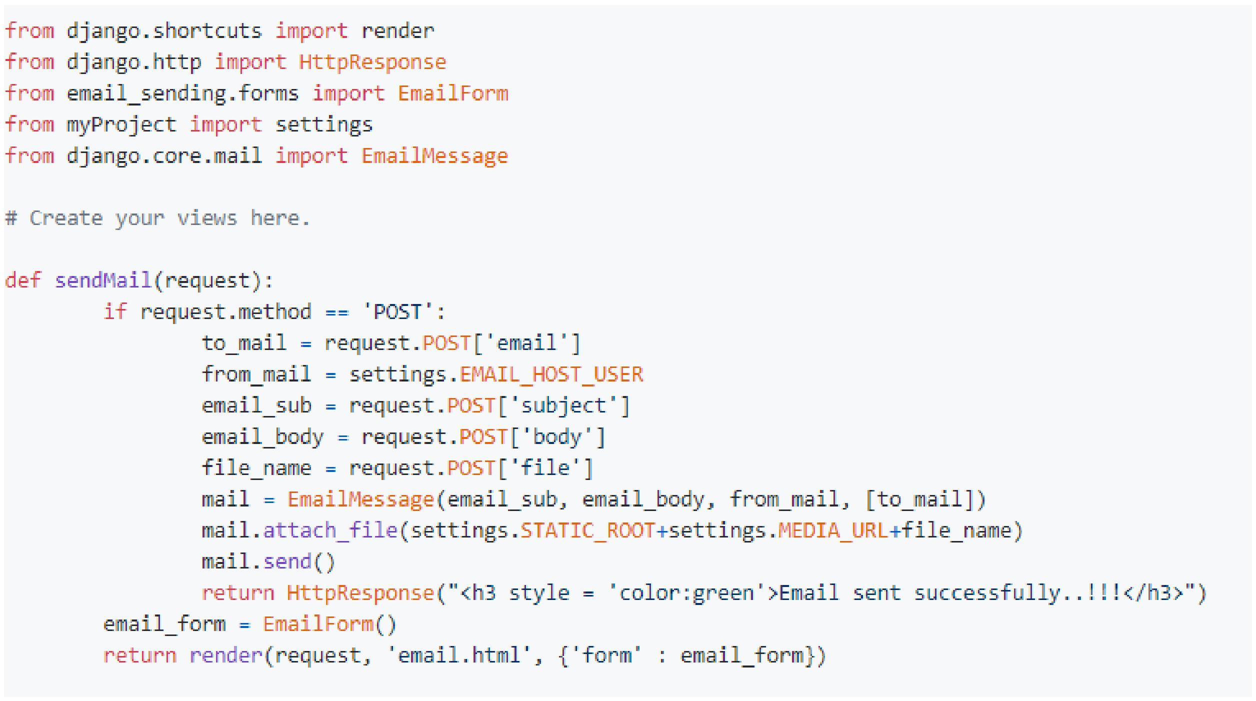
irfan.iit003@gmail.com **Greetings** - Congratulations for your success 6:18 pm

Well, same as, we can send mail using other smtp server configurations if we have.

6.4 Example with image And File Attachments

This example contains the following files

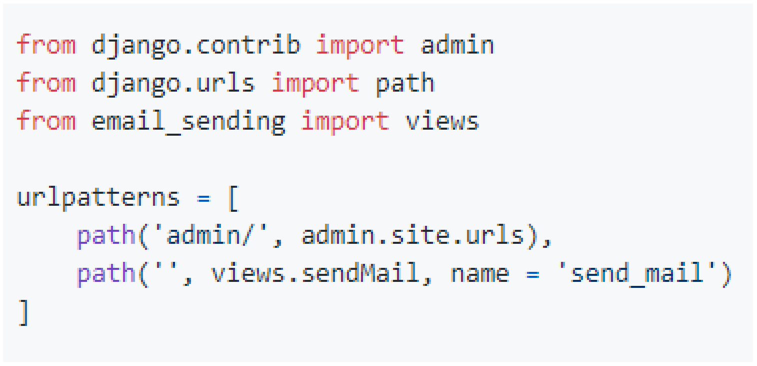
// views.py



```python
from django.shortcuts import render
from django.http import HttpResponse
from email_sending.forms import EmailForm
from myProject import settings
from django.core.mail import EmailMessage

# Create your views here.

def sendMail(request):
        if request.method == 'POST':
                to_mail = request.POST['email']
                from_mail = settings.EMAIL_HOST_USER
                email_sub = request.POST['subject']
                email_body = request.POST['body']
                file_name = request.POST['file']
                mail = EmailMessage(email_sub, email_body, from_mail, [to_mail])
                mail.attach_file(settings.STATIC_ROOT+settings.MEDIA_URL+file_name)
                mail.send()
                return HttpResponse("<h3 style = 'color:green'>Email sent successfully..!!!</h3>")
        email_form = EmailForm()
        return render(request, 'email.html', {'form' : email_form})
```

// urls.py

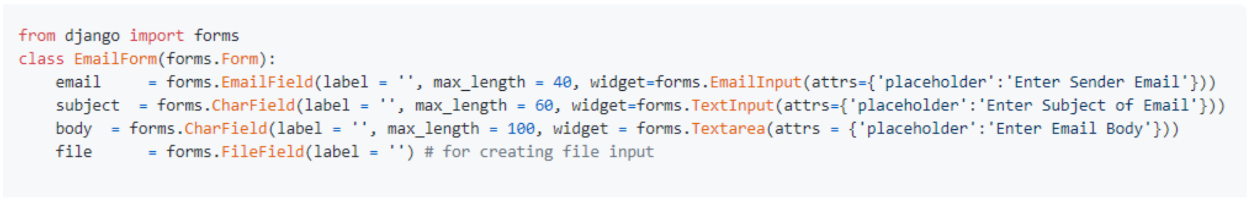
You have to include your url in urls.py file



```python
from django.contrib import admin
from django.urls import path
from email_sending import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.sendMail, name = 'send_mail')
]
```

// Forms.py

Create forms.py file in app folder for generating a form with email, subject, body, file fields by using below lines of code



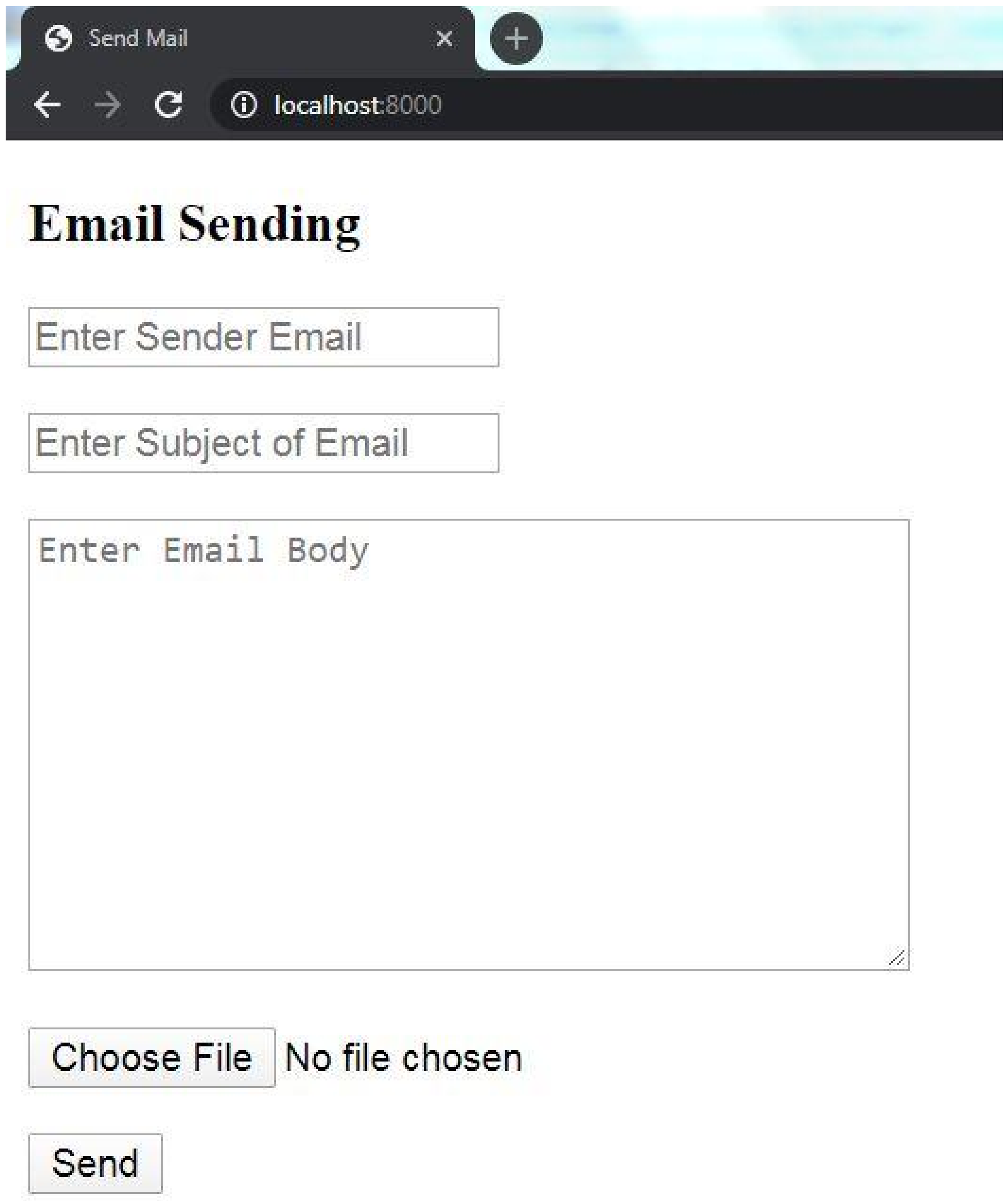
```python
from django import forms
class EmailForm(forms.Form):
    email     = forms.EmailField(label = '', max_length = 40, widget=forms.EmailInput(attrs={'placeholder':'Enter Sender Email'}))
    subject  = forms.CharField(label = '', max_length = 60, widget=forms.TextInput(attrs={'placeholder':'Enter Subject of Email'}))
    body  = forms.CharField(label = '', max_length = 100, widget = forms.Textarea(attrs = {'placeholder':'Enter Email Body'}))
    file      = forms.FileField(label = '') # for creating file input
```

// email.html

Create email.html file in templates folder existed in app folder with below lines of html code

```html
<!DOCTYPE html>
<html>
<head>
        <title>Send Mail</title>
</head>
<body>
<h3>Email Sending</h3>
<form action="{% url 'send_mail' %}" method="POST">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Send</button>
</form>
</body>
</html>
```

Run the server and check in browser will get an form for entering details to send an email as shown below



Enter information to send an email in the form and click on send, So that your email is sent successfully

Confirm that the receiver got an email along with attachment, Thus how we can send an email using Django. For more information about this email sending can check the below references given

Reference

https://docs.djangoproject.com/en/3.0/topics/email/

6.5 Django File Upload

File upload to the server using Django is a very easy task. Django provides built-in libraries and methods that help to upload a file to the server.

The forms.FileField() method 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.

Let's see an example of uploading a file to the server. This example contains the following files.

6.5.1 SetUp

Open the command line and navigate to which directory you want to create the project. First,we need to install a pillow which is a Python image process library Django relies on for image files. For non-image file uploads, a pillow is not needed.

```
pip install pillow
```

// Models.py

Starting with the database model is a good choice.In our case our model image upload has an image field including other fields. We'll also include a str method below so that the firstName and image appears in our Django admin later on.

```
#imageupload/models.py
from django.db import models

# Create your models here.
class upload(models.Model):
        gender_vals = [('Male','Male'),('FeMale','FeMale')]
        firstName = models.CharField(max_length=100)
        lastName = models.CharField(max_length=100)
        emailId = models.EmailField(null = True)
        phoneNo = models.CharField(max_length=10)
        age = models.IntegerField(null=True)
        gender = models.CharField(max_length=10,choices=gender_vals )
        date_of_birth = models.DateField(null=True)
        image = models.ImageField(upload_to='photos/')

        def __str__(self):
                return self.firstName+" "+str(self.image)
```

The location of the uploaded image will be in MEDIA_ROOT/photos. In Django, the MEDIA_ROOT setting is where we define the location of all user uploaded items. We'll set that now.
If we want to use a regular file here the only difference could be to change ImageField to FileField.
The files uploaded to FileField or ImageField are not stored in the database but in the filesystem. FileField and ImageField are created as a string field in the database (usually VARCHAR), containing the reference to the actual file.

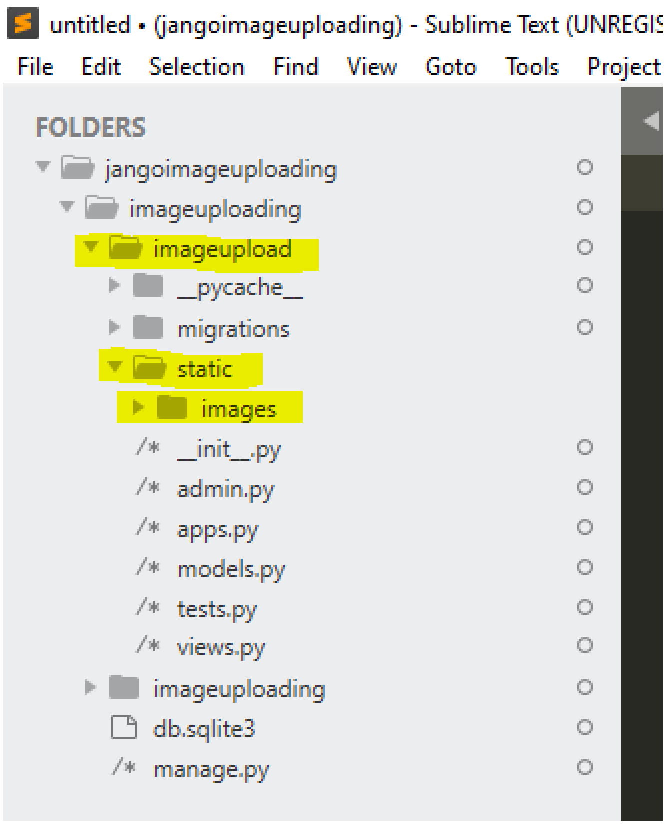
If you delete a model instance containing FileField or ImageField, Django will not delete the physical file, but only the reference to the file.

6.5.2 MEDIA_ROOT and MEDIA_URL

Open up imageuploading/settings.py in your text editor. We will add two new configurations. By default MEDIA_URL and MEDIA_ROOT are empty and not displayed so we need to configure them:

- MEDIA_ROOT is the absolute filesystem path to the directory for user-uploaded files
- MEDIA_URL is the URL we can use in our templates for the files

Now let us store all images in the static folder.For that create a static folder inside our app.After that create an images folder inside static folder

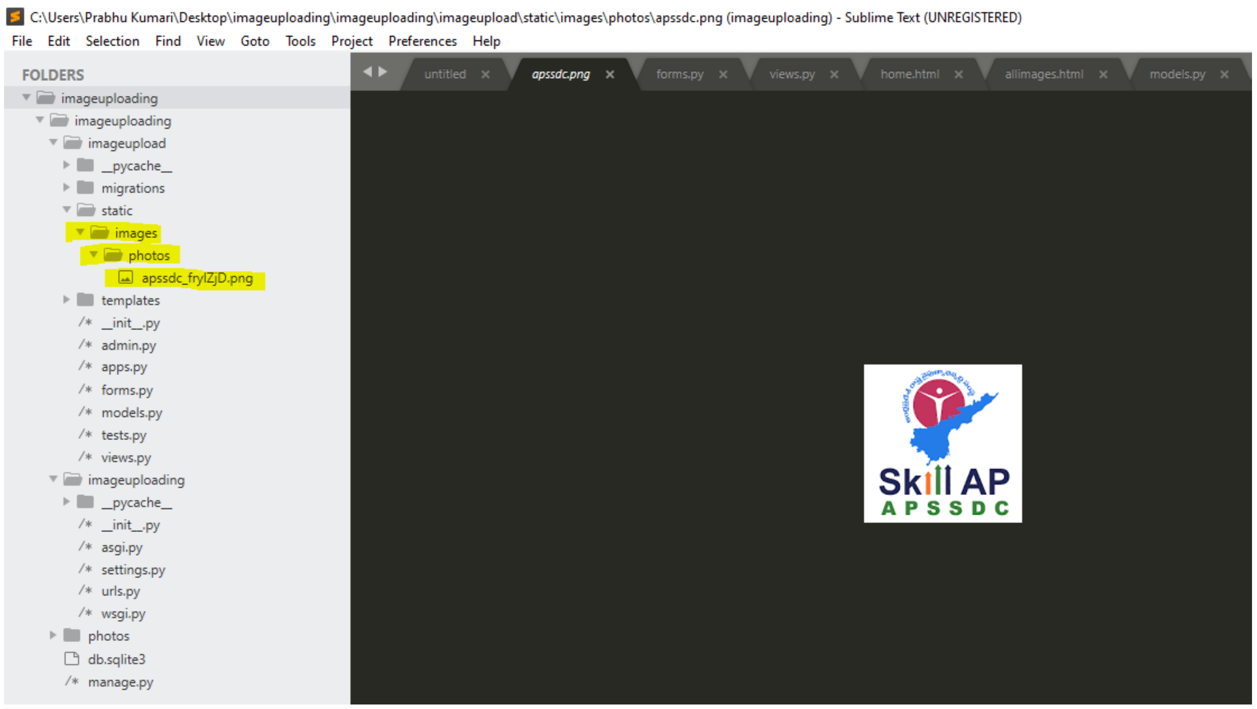


Now let us add the MEDIA_ROOT and MEDIA_URL to our settings.py

```
#imageuploading/settings.py
MEDIA_ROOT=os.path.join(BASE_DIR,'imageupload/static/images')

MEDIA_URL='/photos/'
```

In MEDIA_ROOT we are giving our file path i,e upload/static/images all user uploaded files will configure to this path. In MEDIA_URL we have given photos i,e all the user uploaded files will be created under the photos which are created in the images folder that you will see later on.

If you look within the local folder in your project you'll see under images there is now the apssdc_frylZjD.png image file under photos that is what MEDIA_URL would do.


C:\Users\Prabhu Kumari\Desktop\imageuploading\imageuploading\imageupload\static\images\photos\apssdc.png (imageuploading) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
untitled
apssdc.png
forms.py
views.py
home.html
allimages.html
models.py

```
imageuploading
  imageuploading
    imageupload
      __pycache__
      migrations
      static
        images
          photos
            apssdc_fryIZjD.png
      templates
      __init__.py
      admin.py
      apps.py
      forms.py
      models.py
      tests.py
      views.py
    imageuploading
      __pycache__
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
    photos
    db.sqlite3
    manage.py
```

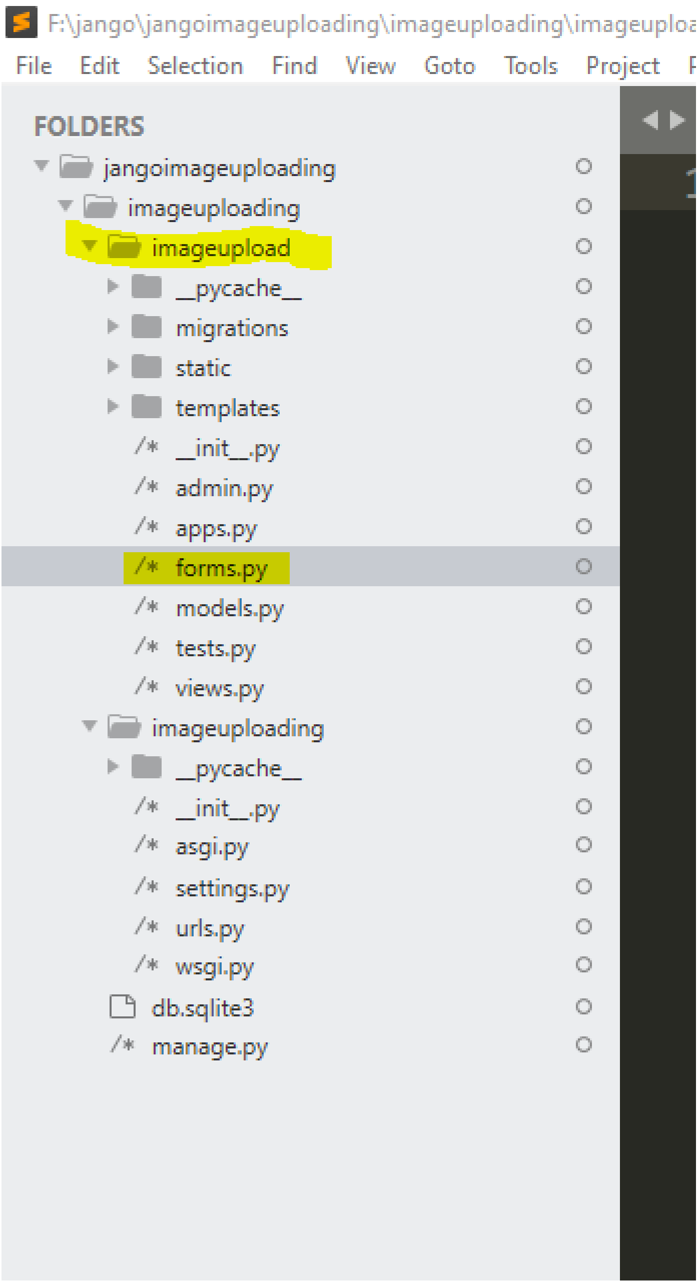
// Urls.py

We'll need two urls.py file updates. First at the project-level imageuploading/urls.py files we need to add imports for settings and static and views.Then define a route for the image upload app. Note,we also need to add the MEDIA_URL if settings are in DEBUG mode, otherwise we won't be able to view uploaded images locally.

```
#imageuploading/urls.py

from django.contrib import admin
from django.urls import path
from django.conf import settings #new
from django.conf.urls.static import static #new
from imageupload import views #new

urlpatterns = [
    path('admin/', admin.site.urls),
    path('home/', views.home,name="home"), #new
]

if settings.DEBUG: #new
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

// Forms.py

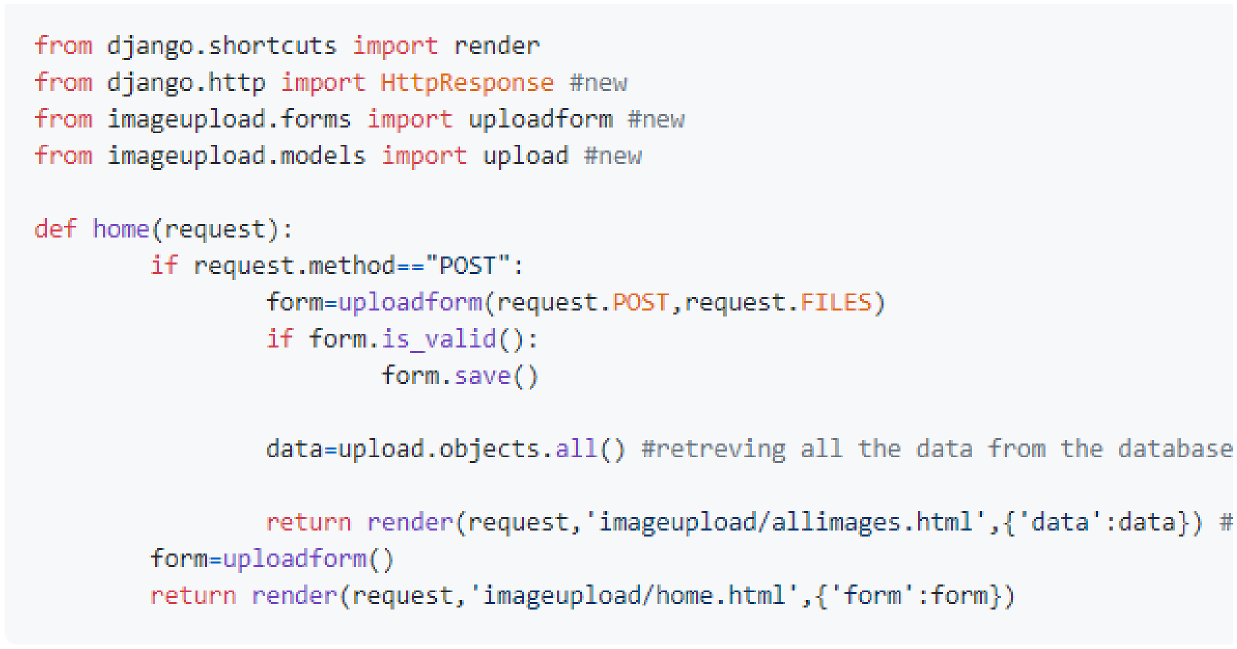
Now we can add a form so regular users, who wouldn't have access to the admin, can also upload photos.We can extend Django's built-in ModelForm in forms.py or we can create our own html form in templates Let us proceed using Django's built-in ModelForm.Create a new file within the app imageupload/forms.py and create a Modelform by importing the model that you have created.



```
#imageupload/forms.py
from django.forms import ModelForm
from imageupload.models import upload
class uploadform(ModelForm):
        class Meta:
                model=uploadform
                fields='__all__'
```

// Views.py

create a home view that uses the model and and templates

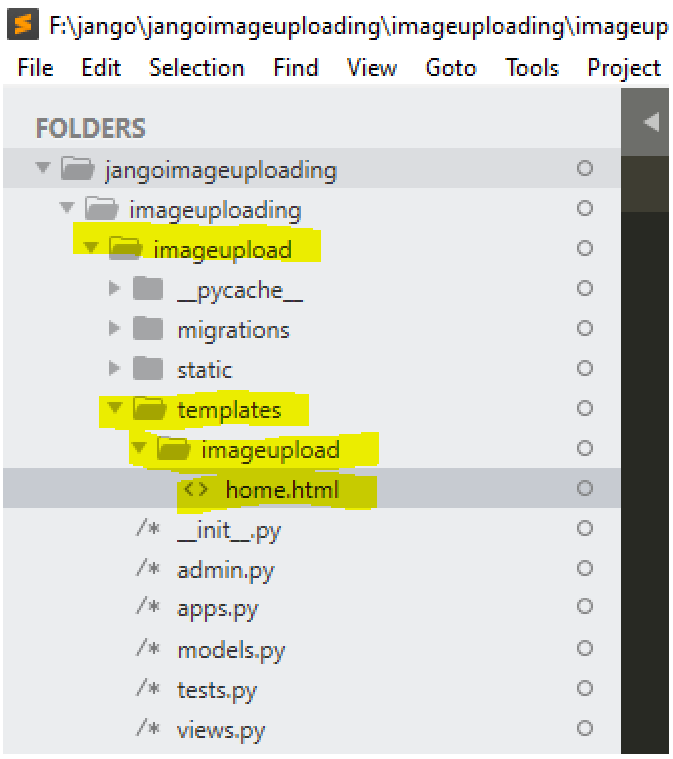


```
from django.shortcuts import render
from django.http import HttpResponse #new
from imageupload.forms import uploadform #new
from imageupload.models import upload #new

def home(request):
        if request.method=="POST":
                form=uploadform(request.POST,request.FILES)
                if form.is_valid():
                        form.save()

                data=upload.objects.all() #retreving all the data from the database

                return render(request,'imageupload/allimages.html',{'data':data}) #
        form=uploadform()
        return render(request,'imageupload/home.html',{'form':form})
```

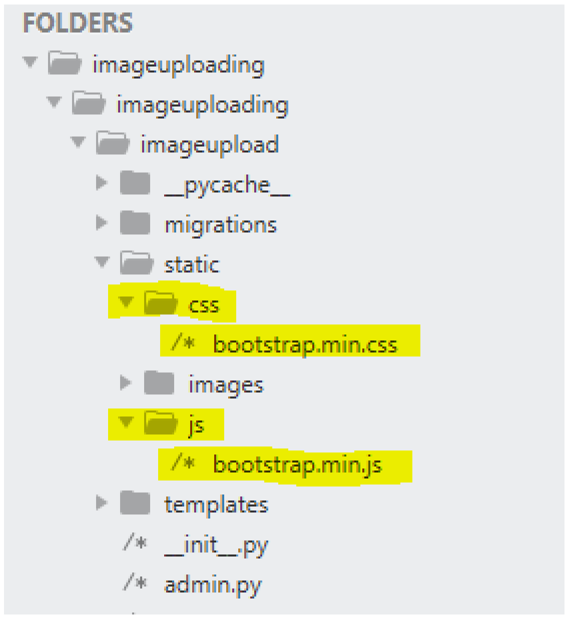
When files are submitted to the server, the file data ends up placed in request.FILES.The request.FILES is a dictionary-like object. Each key in request.FILES is the name from the <input type="file" name="" /> that you can see below html code.Each value in request.FILES is an UploadedFile instance.

// Templates.py

Here we used two templates in order to upload the file home.htmland view all the images all images that we have uploaded Now create a template home.html, within the app imageupload/templates/imageupload/home.html and and write the html code



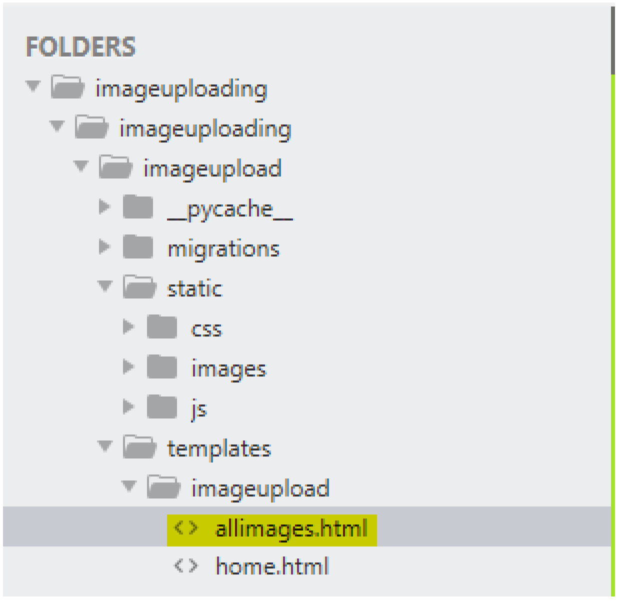
In case if you want to apply bootstrap to your template you can add.For that,you need to create css and js folders in a static folder and add css and js files if you want to use it offline.



Here you have to write home code inside home.html page

```
<!DOCTYPE html>
<html>
<head>
        {% load static %}
        <title>Register Here..</title>
        <link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.min.css' %}"
        <script type="text/javascript" src="{% static 'js/bootstrap.min.js' %}"></script>

</head>
<body>
        <div  class="row justify-content-center">
        <form action="{% url 'home' %}" method="POST" enctype="multipart/form-data" >
                <div class="card">
                        <div class="card-header bg-info">Register Here</div>
                        <div class="card-body">

                {% csrf_token %}
                {{ form.as_p }}
        </div>
        <div class="card-footer">
                <button type="submit" class="btn btn-primary"> Submit</button>
        </div>

        </form>
</div>
</body>
</html>
```

It is mandatory for the HTML form to have the attribute enctype="multipart/form-data" set correctly. Otherwise the request.FILES will be empty.
Also create a template allimages.html and write down the html code



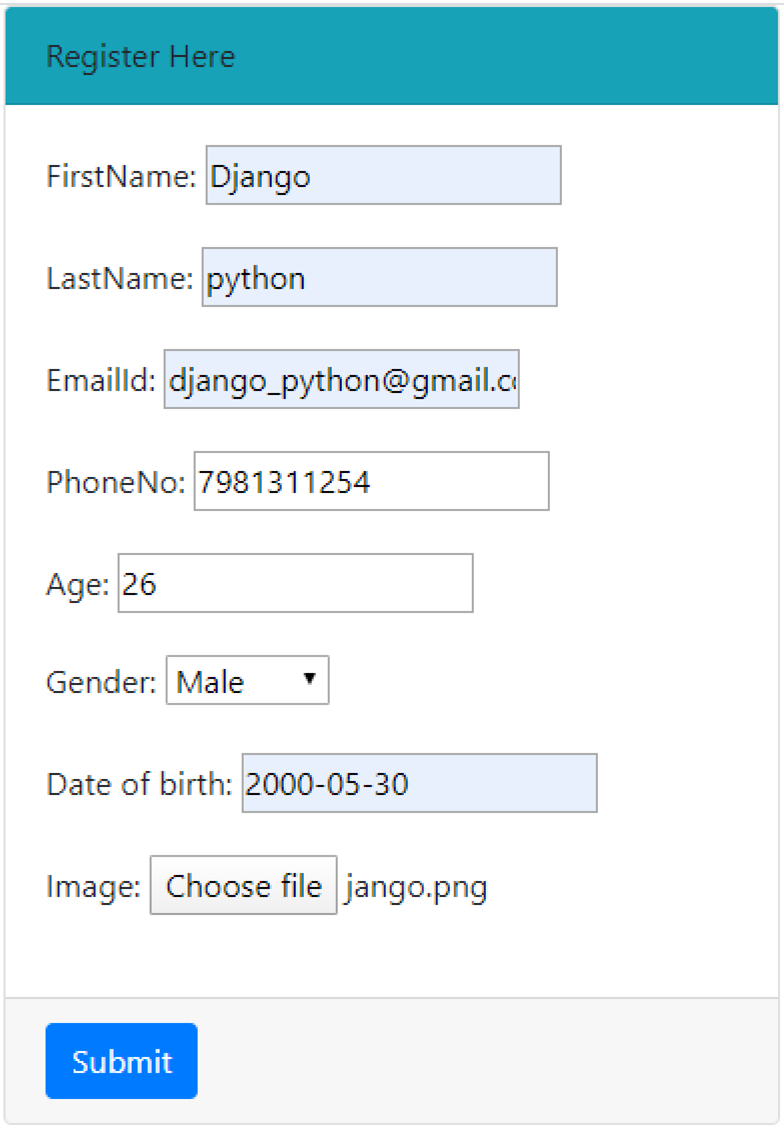
```
FOLDERS
▼ imageuploading
  ▼ imageuploading
    ▼ imageupload
      ▶ __pycache__
      ▶ migrations
      ▼ static
        ▶ css
        ▶ images
        ▶ js
      ▼ templates
        ▼ imageupload
            <> allimages.html
            <> home.html
```

Here you have to write your allimages.html code

```
<!DOCTYPE html>
<html>
<head>
        <title>Allimages</title>
</head>
<body>
<table border="1">
        <thead>
                <tr>
                        <th>Name</th>
                        <th>Image</th>
                </tr>

        </thead>
        <tbody>
        {% for row in data %}
        <tr>
        <td> {{row.firstName}} </td>
    <td><img src="/photos/{{ row.image }}" height=150px width=150px></td>
    </tr>
{% endfor %}
</tbody>
</table>
</body>
</html>
```

Ok, that's it! Make sure the server is running with the python manage.py runserver command and navigate to our homepage at http://127.0.0.1:8000/home. Refresh the page if needed.

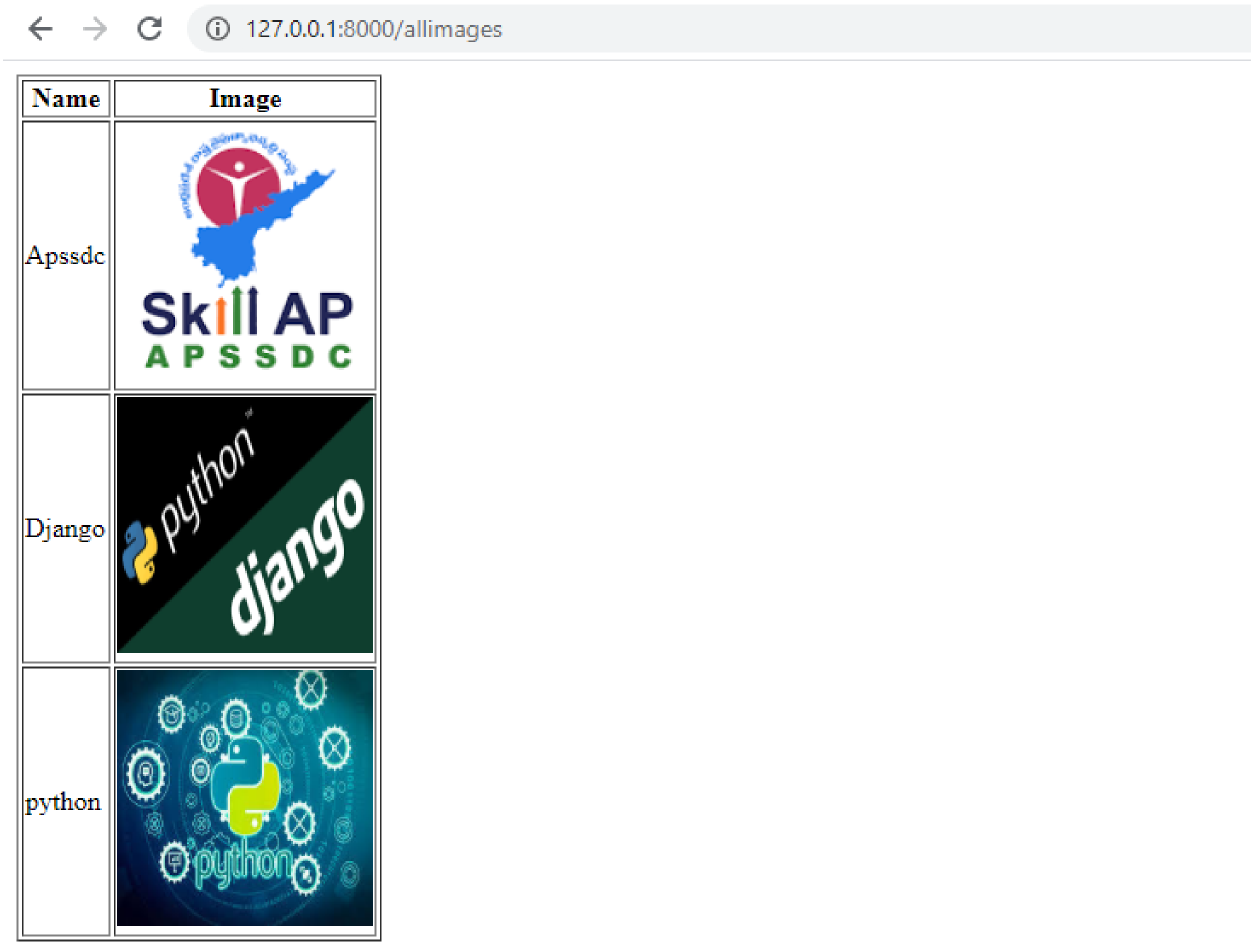
Output



Register Here

FirstName:	Django
LastName:	python
EmailId:	django_python@gmail.c
PhoneNo:	7981311254
Age:	26
Gender:	Male
Date of birth:	2000-05-30
Image:	Choose file jango.png

After giving the details,click on upload.You will be redirected to the allimages.html which you can see all the image that you have uploaded



127.0.0.1:8000/allimages

Name	Image
Apssdc	
Django	
python	