Django Sign Up and login with confirmation Email | Python

Last Updated : 15 Sep, 2023

Django by default provides an *authentication system configuration*. **User** objects are the core of the authentication system. Today we will implement Django's authentication system.

Modules required: Django install, crispy_forms

Django Sign Up and Login with Confirmation Email

To install crispy_forms you can use the terminal command:

```
pip install --upgrade django-crispy-forms
```

Start a project with the following command –

```
django-admin startproject project
```

Change directory to project –

```
cd project
```

Start the server- Start the server by typing the following command in the terminal –

```
python manage.py runserver
```

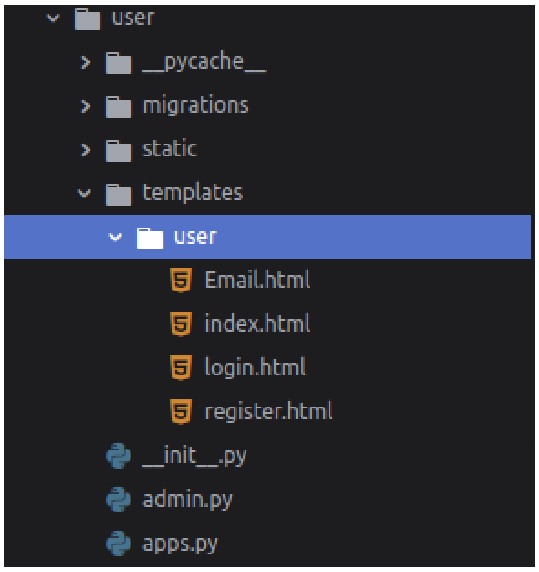
To check whether the server is running or not go to a web browser and enter *http://127.0.0.1:8000/* as URL. and to stop the server press keys

```
ctrl+c
```

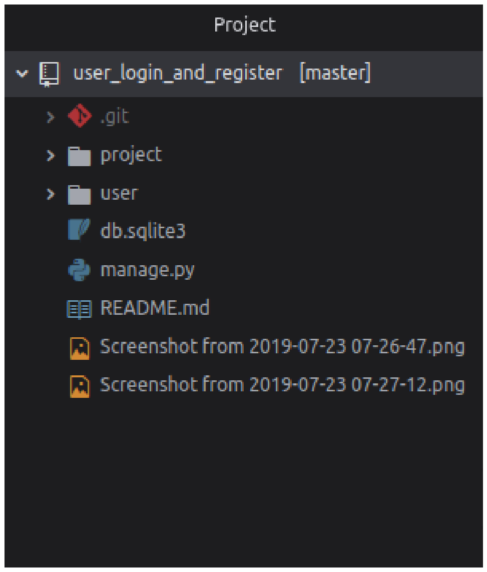
Let's create an app now called the "user".

```
python manage.py startapp user
```

Goto user/ folder by doing: *cd user* and create a folder *templates* with files *index.html, login.html, Email.html, register.html* files.


```
user
  __pycache__
  migrations
  static
  templates
    user
      Email.html
      index.html
      login.html
      register.html
  __init__.py
  admin.py
  apps.py
```

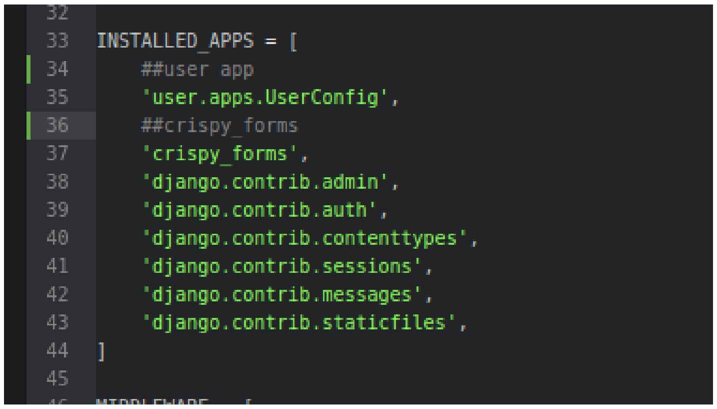
Open the project folder using a text editor. The directory structure should look like this :


```
Project
user_login_and_register [master]
  .git
  project
  user
  db.sqlite3
  manage.py
  README.md
  Screenshot from 2019-07-23 07-26-47.png
  Screenshot from 2019-07-23 07-27-12.png
```

Now add the "user" app and "crispy_form" in your todo_site in settings.py, and add

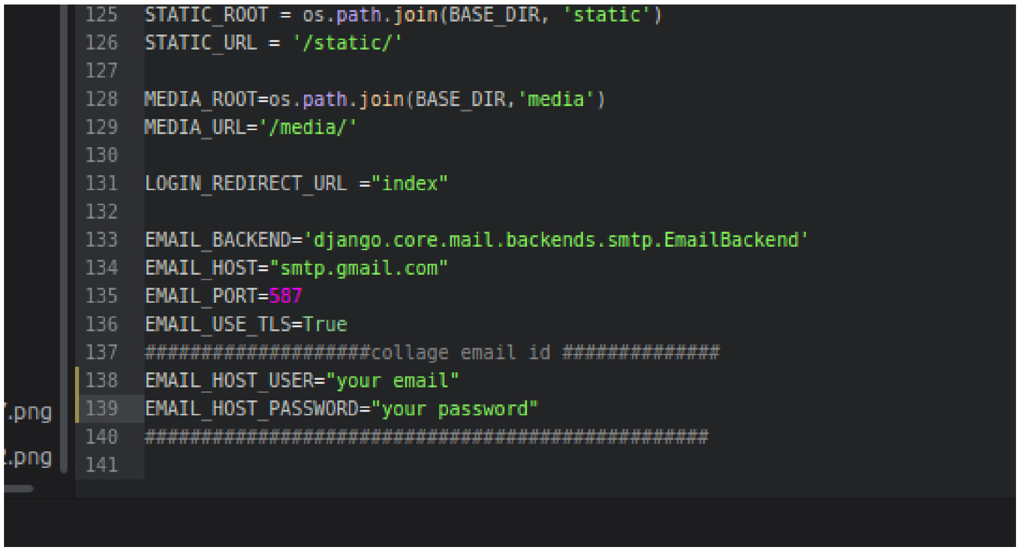
```
CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

at last of settings.py



```
INSTALLED_APPS = [
    ##user app
    'user.apps.UserConfig',
    ##crispy_forms
    'crispy_forms',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

configure *email settings* in setting.py:



```
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
STATIC_URL = '/static/'

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

LOGIN_REDIRECT_URL ="index"

EMAIL_BACKEND='django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST="smtp.gmail.com"
EMAIL_PORT=587
EMAIL_USE_TLS=True
####################collage email id ##############
EMAIL_HOST_USER="your email"
EMAIL_HOST_PASSWORD="your password"
##################################################
```

place your email and password here.

Edit *urls.py* file in project

In this file we provide the path for the login,logout ,register page and include the user.urls to the main project URL file.

Python3

```
from django.contrib import admin
from django.urls import path, include
from user import views as user_view
from django.contrib.auth import views as auth

urlpatterns = [

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

    ##### user related path##########################
    path('', include('user.urls')),
    path('login/', user_view.Login, name ='login'),
    path('logout/', auth.LogoutView.as_view(template_name ='user/index.html'), nam
    path('register/', user_view.register, name ='register'),

]
```

Edit urls.py in user

Here we provide the URL path for index view and these views are connected to the main project URL file.

Python3

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

urlpatterns = [
         path('', views.index, name ='index'),
]
```

Edit views.py in user

Now we will provide the logic and code for the Email system in the views of user app

Python3

```
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth import authenticate, login
from django.contrib.auth.decorators import login_required
from django.contrib.auth.forms import AuthenticationForm
from .forms import UserRegisterForm
from django.core.mail import send_mail
from django.core.mail import EmailMultiAlternatives
from django.template.loader import get_template
from django.template import Context


#################### index#######################################
def index(request):
    return render(request, 'user/index.html', {'title':'index'})

########### register here #####################################
def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            email = form.cleaned_data.get('email')
            ######################### mail system ################################
            htmly = get_template('user/Email.html')
            d = { 'username': username }
            subject, from_email, to = 'welcome', 'your_email@gmail.com', email
            html_content = htmly.render(d)
            msg = EmailMultiAlternatives(subject, html_content, from_email, [to])
            msg.attach_alternative(html_content, "text/html")
            msg.send()
            ##################################################################
            messages.success(request, f'Your account has been created ! You are nc
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'user/register.html', {'form': form, 'title':'register

################ login forms###################################################
def Login(request):
    if request.method == 'POST':

        # AuthenticationForm_can_also_be_used__

        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username = username, password = password)
        if user is not None:
            form = login(request, user)
            messages.success(request, f' welcome {username} !!')
            return redirect('index')
        else:
            messages.info(request, f'account done not exit plz sign in')
    form = AuthenticationForm()
    return render(request, 'user/login.html', {'form':form, 'title':'log in'})
```

Configure your email here.

Now create a forms.py in user

Now with help of django form we will create a Registration page for the new user to register and this will mail to registering gmail from the gmail we mention in the settings.py file of the project.

Python3

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

class UserRegisterForm(UserCreationForm):
    email = forms.EmailField()
    phone_no = forms.CharField(max_length = 20)
    first_name = forms.CharField(max_length = 20)
    last_name = forms.CharField(max_length = 20)
    class Meta:
        model = User
        fields = ['username', 'email', 'phone_no', 'password1', 'password2']
```

Navigate to *templates/user/* and edit files :

Index.html file

This file includes metadata, loads external CSS and JavaScript files (Bootstrap and Font Awesome), and uses Django template tags to handle dynamic content. The template features a navigation bar, displays alert messages, and adjusts the page content based on user authentication, showing a personalized welcome message or a login prompt. This code is designed for building user-friendly web interfaces within a Django project.

HTML

```
{% load static %}
{% load crispy_forms_tags %}
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="title" content="project">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <meta name="language" content="English">
  <meta name="author" content="vinayak sharma">

  <title>{{title}}</title>


  <!-- bootstrap file -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"><
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js
  <!-- bootstrap file-->

  <!-- jQuery -->
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome


  <!-- main css -->
  <link rel="stylesheet" type="text/css" href="{% static "index.css" %}" />


  <!-- message here -->

  {% if messages %}
  {% for message in messages %}

  <script>
    alert("{{ message }}");
  </script>

  {% endfor %}
  {% endif %}

  <!--_______________________________________________-->



</head>
```

```
<body class="container-fluid">


  <header class="row">

    <!-- navbar-->
    <nav class="navbar navbar-inverse navbar-fixed-top">
      <div class="container-fluid">
        <div class="navbar-header">
          <button class="navbar-toggle" data-toggle="collapse" data-target="#main
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" class="styleheader" href="{% url "index" %}">pr
        </div>
        <div class="collapse navbar-collapse" id="mainNavBar">
          <ul class="nav navbar-nav navbar-right">
            <li><a href="{% url "index" %}">Home</a></li>

            {% if user.is_authenticated %}
            <li><a href="{% url "logout" %}"><span class="glyphicon glyphicon-log
            {% else %}
            <li><a href="{% url "register" %}"><span class="glyphicon glyphicon-u
            <li><a href="{% url "login" %}"><span class="glyphicon glyphicon-log-
            {% endif %}

          </ul>
        </div>
      </div>
    </nav>
  </header>
  <br/>
  <br>
  <br>
  <div class="row">
    {% block start %}
    {% if user.is_authenticated %}
    <center><h1>welcome back {{user.username}}!</h1></center>
    {% else %}
    <center><h1>log in, plz . . .</h1></center>
    {% endif %}
    {% endblock %}
  </div>
</body>

</html>
```

Email.html

The provided HTML code is an email template for a registration confirmation message. It uses the Roboto font, has a centered thank-you message with user-specific content (username), and a horizontal line for separation. This template is designed to deliver a visually pleasing and informative confirmation email to users.

HTML

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title></title>
        <style>
            @import url('https://fonts.googleapis.com/css?family=Roboto:400,100,30
        </style>
    </head>
    <body style="background: #f5f8fa;font-family: 'Roboto', sans-serif;">
        <div style="width: 90%;max-width:600px;margin: 20px auto;background: #fff
            <section style="margin: 0 15px;color:#425b76;">
                <h2 style="margin: 40px 0 27px 0;text-align: center;">Thank you t
                <hr style="border:0;border-top: 1px solid rgba(66,91,118,0.3);max
                <p style="font-size:15.5px;font-weight: bold;margin:40px 20px 15p
            </section>
        </div>
    </body>
</html>
```

Login.html

Inside this block, it creates a centered login form with specific styling, including a black border, padding, and a rounded border. The form includes a CSRF token for security and uses the crispy filter to render form fields with enhanced formatting, along with a login button and a link to the registration page.

HTML

```
{% extends "user/index.html" %}
{% load crispy_forms_tags %}
{% block start %}

 <div class="content-section col-md-8 col-md-offset-2">
  <center>
  <form method="POST" style="border: 1px solid black; margin: 4%; padding:10%; bo
    {% csrf_token %}
    <fieldset class="form-group">
      {{ form|crispy}}
    </fieldset>
   <center>
    <button style="background: black; font-size: 2rem; padding:1%;" class="btn bt
  </center>
  <br/>
  <sub style="text-align: left;"><a href="{% url 'register' %}" style="text-decor
  </form>
</center>
 </div>
{% endblock start %}
```

Register.html

This file creates a centered sign-up form with specific styling, including a black border, padding, and rounded corners. The form includes a CSRF token for security and uses the crispy filter for enhanced form field rendering, along with a sign-up button and a link to the login page for users with existing accounts.

Python3

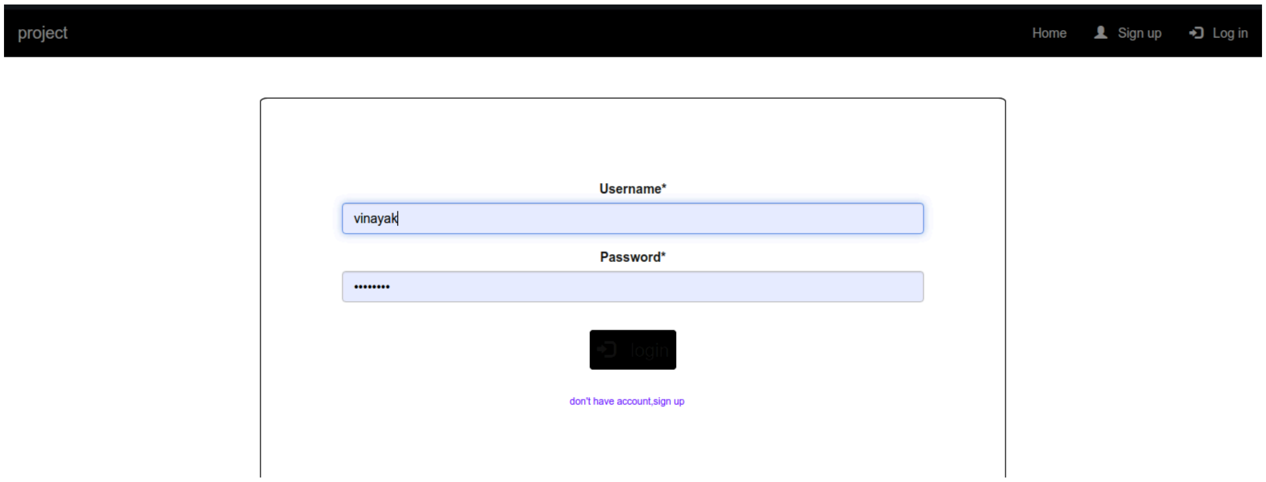
```
{% extends "user/index.html" %}
{% load crispy_forms_tags %}
{% block start %}

<div class="content-section col-md-8 col-md-offset-2">
  <form method="POST" style="border: 1px solid black; margin: 4%; padding:10%; bo
    {% csrf_token %}
    <fieldset class="form-group">
      {{ form|crispy}}
    </fieldset>
    <center>
      <button style="background: black; padding:2%; font-size: 2rem; color:white;'
    </center>
    <br />
    <sub><a href="{% url "login" %}" style="text-decoration: none; color: blue; pa
  </form>
</div>
{% endblock start %}
```

Make migrations and migrate them.

```
python manage.py makemigrations
python manage.py migrate
```

Now you can run the server to see your app.

```
python manage.py runserver
```



Are you ready to elevate your web development skills from foundational knowledge to advanced expertise? Explore our [Mastering Django Framework - Beginner to Advanced Course] on GeeksforGeeks, designed for aspiring developers and experienced programmers. This comprehensive course covers everything you need to know about Django, from the basics to advanced features. Gain practical experience through **hands-on projects** and real-world applications, mastering essential Django principles and techniques. Whether you're just starting or looking to refine your skills, this course will empower you itsvin... 38

Previous Article

Weather app using Django | Python

Next Article

ToDo webapp using Django

Similar Reads

Program to display Astrological sign or Zodiac sign for given date of birth

For given date of birth, this program displays an astrological sign or Zodiac sign.Examples : Input : Day = 10, Month = December Output : Sagittarius...

8 min read

Securing Django Admin login with OTP (2 Factor Authentication)

Multi factor authentication is one of the most basic principle when adding security for our applications. In this tutorial, we will be adding multi factor authentication...

2 min read

Django: Redirect to Previous Page After Login

Handling user navigation smoothly is crucial for a good user experience in web applications. One common requirement is to redirect users back to their previous...

4 min read

Email + Social Logins in Django - Step by Step Guide

This article revolves around a Django Project. It includes Email + Social Login integration in any Django project. We have used React as front end to...

14 min read

How to Send Email with Django

Django, a high-level Python web framework, provides built-in functionality to send emails effortlessly. Whether you're notifying users about account...

4 min read

Python | Counting sign change in list containing Positive and Negative…

Given a list containing Positive and Negative integers, We have to find number of times the sign(Positive or Negative) changes in the list. Input: [-1, 2, 3, -4, 5, -6, ...

5 min read

Compute the sign and natural logarithm of the determinant of an array in…

In this article, we will cover how to compute the sign and natural logarithm of the determinant of an array in Python using NumPy. numpy.linalg.slogdet() method...

3 min read

Login Application and Validating info using Kivy GUI and Pandas in Python

Prerequisites : Kivy, Pandas Kivy is a multiplatform GUI library, known for being responsive. It provides management of multiple screens in a single application. I...

5 min read

Adding Tags Using Django-Taggit in Django Project

Django-Taggit is a Django application which is used to add tags to blogs, articles etc. It makes very easy for us to make adding the tags functionality to our djang...

2 min read

How to customize Django forms using Django Widget Tweaks ?

Django forms are a great feature to create usable forms with just few lines of code. But Django doesn't easily let us edit the form for good designs. Here, we...

3 min read

Article Tags : Python Django-Projects Python Django

Practice Tags : python

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD

Inteview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects