



Note-taking App using Django

Last Updated : 24 Sep, 2024

In this article, we will explore a note-making app. In this article, we will create a note-making app using [Django](#). We will also use the [Django authentication system](#). Users need to create an account on the web to access the note-making app. After that, users will have access to the note-making app. This means users can create notes, update them with a simple click of a button, and also delete notes with a single click. Additionally, users have the option to log out by clicking on a button.

Note-taking App using Django

Users can create notes, update them with a simple click of a button, and also delete notes with a single click. Additionally, users have the option to log out by clicking on a button.

To install Django follow these [steps](#).

Starting the Project Folder

To start the project use this command

```
django-admin startproject notes
cd notes
```

To start the app use this command

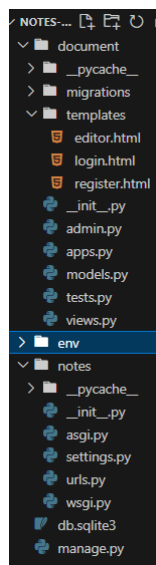
```
python manage.py startapp document
```

Now add this app to the 'settings.py'

```
INSTALLED_APPS = [
    'django.contrib.admin',
```

```
'django.contrib.auth',  
'django.contrib.contenttypes',  
'django.contrib.sessions',  
'django.contrib.messages',  
'django.contrib.staticfiles',  
'document'  
]
```

File Structure



File Structure

Building a **note-taking app** is a great way to apply your Django skills. For those looking to gain a comprehensive understanding of Django, the [Django Web Development Course](#) offers in-depth knowledge.

Setting up Necessary Files

models.py: This code defines a Django model for a document with fields for title, content, creation date, and modification date. The documents are ordered by their titles.

Python

```
1 from django.db import models  
2  
3 class Note(models.Model):  
4     title = models.CharField(max_length=255)
```

```

        content = models.TextField(blank=True, null=True)
6
7        created_at = models.DateTimeField(auto_now_add=True)
8        modified_at = models.DateTimeField(auto_now=True)
9
10       class Meta:
11           ordering = ('title',)

```

views.py: Below are the explanation of the each function:

- **editor Function:** Allows authenticated users to create, edit, and save notes. It also displays existing notes. Users must be logged in to access this page.
- **delete_document Function:** Handles note deletion by taking the docid as a parameter.
- **login_page Function:** Manages user login and provides error messages for incorrect credentials
- **register_page Function:** Manages user registration and provides success/error messages.
- **custom_logout Function:** Handles user logout.

In summary, it's a note-making app with user authentication and registration

Python

```

1  from django.shortcuts import render, redirect
2  from .models import Note
3  from django.http import HttpResponse, JsonResponse,
  HttpResponseRedirect
4  from django.contrib import messages
5  from django.contrib.auth import login, authenticate
6  from django.contrib.auth.decorators import login_required
7  from django.contrib.auth.models import User
8  from django.contrib.auth import logout
9
10
11  # create editor page
12  @login_required(login_url='/login/')
13  def editor(request):
14      docid = int(request.GET.get('docid', 0))
15      notes = Note.objects.all()

```

```
17     if request.method == 'POST':
18         docid = int(request.POST.get('docid', 0))
19         title = request.POST.get('title')
20         content = request.POST.get('content', '')
21
22         if docid > 0:
23             note = Note.objects.get(pk=docid)
24             note.title = title
25             note.content = content
26             note.save()
27
28             return redirect('/?docid=%i' % docid)
29         else:
30             note = Note.objects.create(title=title,
31 content=content)
32
33             return redirect('/?docid=%i' % note.id)
34
35     if docid > 0:
36         note = Note.objects.get(pk=docid)
37     else:
38         note = ''
39
40     context = {
41         'docid': docid,
42         'notes': notes,
43         'note': note
44     }
45
46     return render(request, 'editor.html', context)
47
48 # create delete notes page
49
50 @login_required(login_url='/login/')
51 def delete_note(request, docid):
52     note = Note.objects.get(pk=docid)
53     note.delete()
54
55     return redirect('/?docid=0')
56
57
```

```

# login page for user
59 def login_page(request):
60     if request.method == "POST":
61         try:
62             username = request.POST.get('username')
63
64             user_obj =
        User.objects.filter(username=username)
65             if not user_obj.exists():
66                 messages.error(request, "Username not
        found")
67                 return redirect('/login/')
68             user_obj = authenticate(username=username,
        password=password)
69             if user_obj:
70                 login(request, user_obj)
71                 return redirect('editor')
72                 messages.error(request, "Wrong Password")
73                 return redirect('/login/')
74             except Exception as e:
75                 messages.error(request, "Something went wrong")
76                 return redirect('/register/')
77             return render(request, "login.html")
78
79
80 # register page for user
81 def register_page(request):
82     if request.method == "POST":
83         try:
84             username = request.POST.get('username')
85             password = request.POST.get('password')
86             user_obj =
        User.objects.filter(username=username)
87             if user_obj.exists():
88                 messages.error(request, "Username is
        taken")
89                 return redirect('/register/')
90             user_obj =
        User.objects.create(username=username)
91             user_obj.set_password(password)
92             user_obj.save()
93             messages.success(request, "Account created")
94             return redirect('/login')
95         except Exception as e:

```

Django Basics

Django Projects

Django Interview Question

Flask

Flask Projects

Python API

Torando

Chk

```
        messages.error(request, "Something went wrong")
97         return redirect('/register')
98     return render(request, "register.html")
99
100
101 # logout function
102 def custom_logout(request):
103     logout(request)
104     return redirect('login')
```

Creating GUI

login.html: This is an HTML template for a login page with a minimalistic design. It includes a form for users to enter their username and password, along with a "Login" button. If the login is successful, a success message is displayed. Users can also navigate to a registration page using a provided link. The styling is done using Bootstrap and Font Awesome.

HTML



```
1  <!doctype html>
2  <html lang="en">
3
4  <head>
5      <!-- Required meta tags -->
6      <meta charset="utf-8">
7      <meta name="viewport" content="width=device-width,
initial-scale=1">
8      <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/
bootstrap.min.css" rel="stylesheet"
9          integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIIm9Nao0Yz1ztcQTWfSpd3yD65VohhpUuCOmL
ASjC" crossorigin="anonymous">
10     <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">
11     <title>Job Portal</title>
12 </head>
13 <style>
14     .text{
```

```

        color: green;
16         font-weight: bold;
17         font-family: 'Times New Roman', Times, serif;
18     }
19
20 </style>
21 <body><br><br><br><br>
22
23
24     <br><br>
25
26     <div class="container mt-4 bg-white col-md-3 card
shadow p-3 " id="log">
27         <div class="login-form">
28             {% if messages %}
29             {% for message in messages %}
30                 <div class="alert alert-success {{ message.tags
31 }} mt-4" role="alert">
32                     {{ message }}
33                 </div>
34             {% endfor %}
35             {% endif %}
36             <form action="" method="post">
37                 {% csrf_token %}
38                 <h3 class="text text-center"> GeeksforGeeks
39 </h3>
40                 <h4 class="text-center" style="font-family:
'Times New Roman', Times, serif;"> Login </h4>
41                 <div class="form-group">
42                     <input type="text" class="form-control"
name="username" placeholder="Username" required
43                     style="background-color: #fff;
border: 1px solid #ddd; border-radius: 5px; padding: 10px;">
44                     </div>
45                     <div class="form-group mt-2">
46                         <input type="password" class="form-
control" name="password" placeholder="Password" required
47                         style="background-color: #fff;
border: 1px solid #ddd; border-radius: 5px; padding: 10px;">
48                         </div>
49                         <div class="form-group mt-2">
50                             <button class="btn btn-success btn-
block" style="margin-left: 138px;">Login ????</button>
51                             </div>
52                     <br>

```

```

        </form>
52         <p class="text-center" style="color: #555;"><a
href="{% url 'register' %}" style="color: #007bff;">Create
an
53             Account.</a></p>
54     </div>
55 </div>
56
57 </body>
58
59 </html>

```

register.html: This HTML template is designed for a registration page with a clean and simple layout. Users can input their username and password, and upon successful registration, a success message is shown. The page features styling using Bootstrap and Font Awesome icons, providing a visually appealing user experience.

HTML

```

1  <!doctype html>
2  <html lang="en">
3
4  <head>
5      <!-- Required meta tags -->
6      <meta charset="utf-8">
7      <meta name="viewport" content="width=device-width,
initial-scale=1">
8      <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/
bootstrap.min.css" rel="stylesheet"
9          integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpuaCOML
ASjC" crossorigin="anonymous">
10     <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">
11     <title>Job Portal</title>
12 </head>
13
14 <body>
15

```



```

17         <style>
18             .text{
19                 color: green;
20                 font-weight: bold;
21                 font-family: 'Times New Roman', Times, serif;
22             }
23         </style>
24
25         <body>
26             <br> <br><br><br><br><br>
27
28             <div class="container mt-4  bg-white mx-auto col-
md-3 card shadow p-3">
29                 <div class="login-form">
30                     {% if messages %}
31                     {% for message in messages %}
32                         <div class="alert alert-success {{
message.tags }}" role="alert">
33                             {{ message }}
34                         </div>
35                     {% endfor %}
36                     {% endif %}
37                     <form action="" method="post">
38                         {% csrf_token %}
39                         <h3 class="text text-center">
GeeksforGeeks </h3>
40                         <h4 class="text-center" style="font-
family: 'Times New Roman', Times, serif;"> Register </h4>
41                         <div class="form-group">
42                             <input type="text" class="form-
control" name="username" placeholder="Username" required>
43                         </div>
44                         <div class="form-group mt-2">
45                             <input type="password" class="form-
control" name="password" placeholder="Password" required>
46                         </div>
47                         <div class="form-group mt-2">
48                             <button class="btn btn-success btn-
block" style="margin-left: 117px;">Register ???</button>
49                         </div>
50                         <br>
51                     </form>
52                     <p class="text-center"><a href="{% url
'login' %}">Log In </a></p>

```

```

        </div>
54    </div>
55
56    </body>
57
58 </html>

```

editor.html: This HTML template is for a web page with a header for creating and managing notes. It uses the Bulma CSS framework for styling and features a navigation bar with options to create a new document and log out. The main content area allows users to enter a title and content for their notes and provides options to save and delete them. The page uses a custom font style for a unique look and feel.

HTML

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width,
        initial-scale=1">
6      <title>Notes</title>
7      <link rel="stylesheet"
        href="https://cdn.jsdelivr.net/npm/bulma@0.9.0/css/bulma.min
        .css">
8  </head>
9  <style>
10     .text{
11         color: green;
12         font-weight: bold;
13         margin-top: -10%;
14         margin-left: 200px;
15         font-size: 30px;
16         font-family: 'Times New Roman', Times, serif;
17     }
18 </style>
19
20 <body>
21     <nav class="navbar is-white" > <!-- Changed header color
        to primary -->

```

```

23         <div class="navbar-brand">
24             <a href="{% url 'editor' %}" class="navbar-item
has-text-black ok" style="font-size: 25px; font-weight:bold;
font-family:'Times New Roman', Times, serif; ">Notes</a>
25             <a href="{% url 'editor' %}?docid=0"
class="button is-primary" style=" font-weight:bold; font-
family:'Times New Roman', Times, serif; margin-top:5%;
margin-right:10px">New Note</a>
26
27             <a href="{% url 'logout' %}" class="button is-
danger" style=" font-weight:bold; font-family:'Times New
Roman', Times, serif; margin-top:5%">Logout</a>
28
29
30         </div>
31         <div class="navbar-menu">
32
33
34
35     </div>
36 </nav>
37 <hr>
38
39
40
41 <section class="section">
42
43     <div class="columns">
44         <div class="column is-2">
45             <aside class="menu">
46                 <p class="menu-label" style="font-size:
20px; font-weight:bold; font-family:'Times New Roman',
Times, serif">Notes</p>
47                 <ul class="menu-list">
48                     {% for doc in notes %}
49                         <li>
50                             <a href="{% url 'editor' %}?
docid={{ doc.id }}">{{ doc.title }}</a>
51                         </li>
52                     {% endfor %}
53                 </ul>
54             </aside>
55         </div>
56

```

```

        <div class="column is-5 mt-3 pt-4" >
58             <h3 class="text text-center"> GeeksforGeeks
        </h3>
59             <br>
60             <div class="box" >
61
62                 <form method="post" action="{% url
        'editor' %}">
63                     {% csrf_token %}
64                     <input type="hidden" name="docid"
        value="{{ docid }}">
65                     <div class="field">
66                         <label class="label"
        style="font-size: 22px; font-family:'Times New Roman',
        Times, serif">Title</label>
67                         <div class="control">
68                             <input type="text"
        class="input" name="title" placeholder="Title" {% if note %}
        value="{{ note.title }}" {% endif %}>
69                         </div>
70                     </div>
71                     <div class="field">
72                         <label class="label"
        style="font-size: 20px; font-family:'Times New Roman',
        Times, serif">Content</label>
73                         <div class="control">
74                             <textarea class="textarea"
        name="content" placeholder="Content">{% if note %}{{
        note.content }}{% endif %}</textarea>
75                         </div>
76                     </div>
77                     <div class="field is-grouped">
78                         <div class="control">
79                             <button class="button is-
        primary">Save</button>
80                         </div>
81                         {% if note %}
82                             <div class="control">
83                                 <a href="{% url
        'delete_note' note.id %}" class="button is-
        danger">Delete</a>
84                             </div>
85                         {% endif %}
86                     </div>
87                 </form>
88             </div>

```

```
        </div>
90    </div>
91    </section>
92 </body>
93 </html>
```

admin.py :Here we are registering the models in the admin file.

Python



```
1 from django.contrib import admin
2 from .models import Note
3
4 admin.site.register(Note)
```

urls.py: This Django code configures URL patterns, linking specific URLs to corresponding views or functions. It handles user authentication, custom logout, an editor page, and document deletion. The "admin/" URL is reserved for Django's admin site. Each pattern has a symbolic name for easy reference and URL generation.

Python



```
1 from django.contrib import admin
2 from django.urls import path
3
4 from home.views import *
5
6 urlpatterns = [
7     path('login/' , login_page, name='login'),
8     path('register/' , register_page, name='register'),
9     path('custom_logout/' , custom_logout, name='logout'),
10    path('', editor, name='editor'),
11    path('delete_note/<int:docid>/' , delete_note,
12         name='delete_note'),
13    path('admin/' , admin.site.urls),
14 ]
```

Deployment of the Project

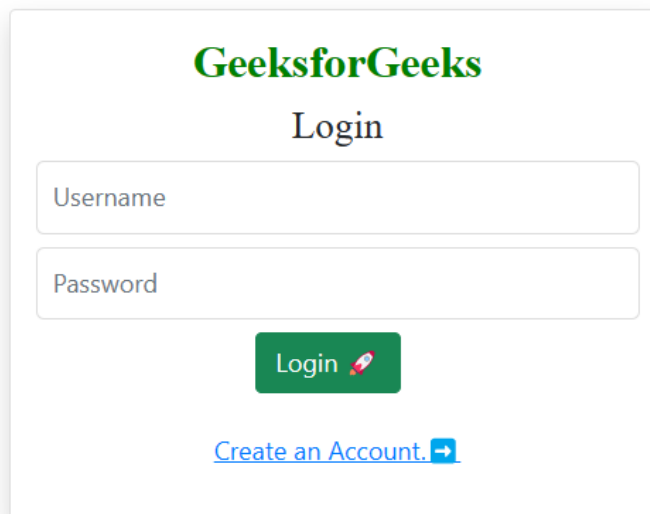
Run these commands to apply the migrations:

```
python3 manage.py makemigrations  
python3 manage.py migrate
```

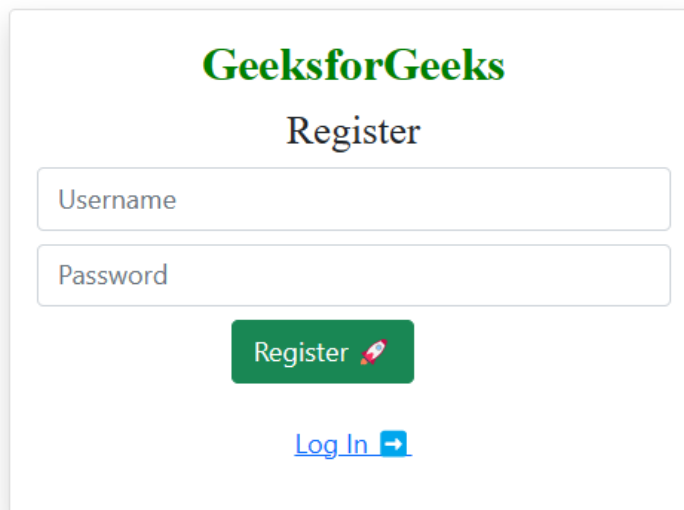
Run the server with the help of following command:

```
python3 manage.py runserver
```

Output



The image shows a web form titled "GeeksforGeeks Login". It features a green header with the "GeeksforGeeks" logo. Below the header, the word "Login" is centered. There are two input fields: "Username" and "Password". Below the "Password" field is a green button with the text "Login" and a rocket icon. At the bottom, there is a blue link that says "Create an Account." followed by a right-pointing arrow icon.



The image shows a web form titled "GeeksforGeeks Register". It features a green header with the "GeeksforGeeks" logo. Below the header, the word "Register" is centered. There are two input fields: "Username" and "Password". Below the "Password" field is a green button with the text "Register" and a rocket icon. At the bottom, there is a blue link that says "Log In" followed by a right-pointing arrow icon.

Notes

New Document

Logout

GeeksforGeeks

DOCUMENTS

GeeksforGeeks

Title

Content

GFG full form is GeeksforGeeks.

Save Delete

Looking to dive into the world of programming or sharpen your Python skills? Our [Master Python: Complete Beginner to Advanced Course](#) is your ultimate guide to becoming proficient in Python. This course covers everything you need to build a solid foundation from fundamental programming concepts to advanced techniques. With **hands-on projects**, real-world examples, and expert guidance, you'll gain the confidence to tackle complex **coding challenges**. Whether you're starting from scratch or aiming to enhance your skills, this course is the perfect fit. Enroll now and master Python, the language of the future!



prath...



1

Next Article

[Language Learning App using Django](#)

Similar Reads

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 django...

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

Integrating Django with Reactjs using Django REST Framework

In this article, we will learn the process of communicating between the Django Backend and React js frontend using the Django REST Framework. For the sake...

15+ min read

Taking Screenshots using pyscreenshot in Python

Python offers multiple libraries to ease our work. Here we will learn how to take a screenshot using Python. Python provides a module called pyscreenshot for th...

2 min read

Weather app using Django | Python

In this tutorial, we will learn how to create a Weather app that uses Django as backend. Django provides a Python Web framework based web framework that...

2 min read

Wikipedia search app Project using Django

Django is a high-level Python based Web Framework that allows rapid development and clean, pragmatic design. It is also called batteries included...

2 min read

Translator App Project using Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it...

2 min read

Create Word Counter app using Django

In this article, we are going to make a simple tool that counts a number of words in text using Django. Before diving into this topic you need to have some basic...

3 min read

Deploying a Django App to Heroku Using Github Repository

Heroku is a free hosting cloud service provider. We can use our free dynos to deploy our applications on the cloud. The only disadvantage is that it loses all t...

2 min read

Build Calculator App Using Django

In this article, we will guide you through the process of creating a calculator app using Django. Our goal is to develop a versatile application capable of performin...

3 min read

Article Tags : [Django](#) [Python](#) [Geeks Premier League](#) [Geeks Premier League 2023](#)

Practice Tags : [python](#)



Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305



Company

[About Us](#)
[Legal](#)
[In Media](#)
[Contact Us](#)
[Advertise with us](#)
[GFG Corporate Solution](#)
[Placement Training Program](#)

Languages

[Python](#)
[Java](#)
[C++](#)
[PHP](#)
[GoLang](#)
[SQL](#)
[R Language](#)

[GeeksforGeeks Community](#)[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](#)

Web Technologies

[HTML](#)[CSS](#)[JavaScript](#)[TypeScript](#)[ReactJS](#)[NextJS](#)[Bootstrap](#)[Web Design](#)

Computer Science

[Operating Systems](#)[Computer Network](#)[Database Management System](#)[Software Engineering](#)[Digital Logic Design](#)[Engineering Maths](#)[Software Development](#)[Software Testing](#)

System Design

[High Level Design](#)[Low Level Design](#)[UML Diagrams](#)[Interview Guide](#)[Design Patterns](#)[OOAD](#)[System Design Bootcamp](#)[Interview Questions](#)

School Subjects

[Mathematics](#)[Physics](#)[Chemistry](#)[Biology](#)

Data Science & ML

[Data Science With Python](#)[Data Science For Beginner](#)[Machine Learning](#)[ML Maths](#)[Data Visualisation](#)[Pandas](#)[NumPy](#)[NLP](#)[Deep Learning](#)

Python Tutorial

[Python Programming Examples](#)[Python Projects](#)[Python Tkinter](#)[Web Scraping](#)[OpenCV Tutorial](#)[Python Interview Question](#)[Django](#)

DevOps

[Git](#)[Linux](#)[AWS](#)[Docker](#)[Kubernetes](#)[Azure](#)[GCP](#)[DevOps Roadmap](#)

Interview Preparation

[Competitive Programming](#)[Top DS or Algo for CP](#)[Company-Wise Recruitment Process](#)[Company-Wise Preparation](#)[Aptitude Preparation](#)[Puzzles](#)

GeeksforGeeks Videos

[DSA](#)[Python](#)[Java](#)[C++](#)

Social Science	Web Development
English Grammar	Data Science
Commerce	CS Subjects
World GK	

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved