



[Django Basics](#) [Django Projects](#) [Django Interview Question](#) [Flask](#) [Flask Projects](#) [Python API](#) [Torando](#) [Che](#)

Recipe Meal Planner using Django

Last Updated : 25 Sep, 2024

In this article, we will create the Recipe Meal Planner using Django step-by-step. Generally, we will implement the **CRUD** (Create, Read, Update, Delete) operations, allowing users to add the recipe name, day, and the recipe itself. Additionally, we will establish a login system, requiring users to register and log in before creating the recipe meal planner. Once the user has added all the daily recipe information, they simply need to click a single button. Subsequently, a PDF form will be generated, which the user can save for future reference.

Recipe Meal Planner using Django

Here, we will create the step-by-step Recipe Meal Planner using Django.

Create Project Folder

To start the project and app use this command

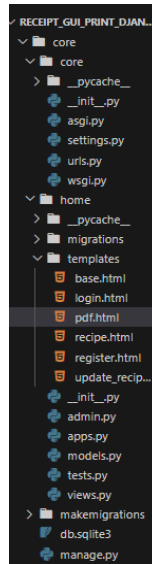
```
django-admin startproject core
cd core
python manage.py startapp home
```

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",
    "home",
```

]

File Structure



Building a **recipe meal planner** is a practical way to learn Django. If you want to enhance your skills and build more feature-rich applications, the [Django Web Development-Basics to Advance Course](#) is an ideal resource.

Setting Necessary Files

models.py: Here, the below code defines a Django model named `Recipe` with fields for user, day, name, and description. The user field is a foreign key to the built-in user model, allowing a null value on deletion. The default values for day, name, and description are set to 'something'.

Python

```
1 from django.db import models
2 from django.contrib.auth.models import User
3
4 #Create mode for receipt
5 class Recipe(models.Model):
6     user = models.ForeignKey(User,
7                               on_delete=models.SET_NULL, null=True, blank=True)
8     day = models.CharField(max_length=100,
9                             default='something')
```

```
8     name = models.CharField(max_length=100,  
    default='something')  
9     description = models.CharField(max_length=100,  
    default='something')
```

views.py: Here, the below code defines a Django application for recipe management, including functionalities for creating, updating, and deleting recipes. It also handles user authentication with login, registration, and logout features. Additionally, there is a basic PDF generation feature for recipes, with the ability to search and filter recipes by day.

Python

```
1  #import all libraries  
2  from django.shortcuts import render, redirect  
3  from .models import Recipe  
4  from django.http import HttpResponse, JsonResponse,  
    HttpResponseRedirect  
5  from django.contrib import messages  
6  from django.contrib.auth import login, authenticate  
7  from django.contrib.auth.decorators import login_required  
8  from django.contrib.auth.models import User  
9  from django.contrib.auth import logout  
10  
11 #create recipes page  
12 @login_required(login_url='/login/')  
13 def recipes(request):  
14     if request.method == 'POST':  
15         data = request.POST  
16         day = data.get('day')  
17         name = data.get('name')  
18         description = data.get('description')  
19         Recipe.objects.create(  
20             day = day,  
21             name=name,  
22             description=description,  
23         )  
24         return redirect('/')  
25  
26     queryset = Recipe.objects.all()  
27     if request.GET.get('search'):
```

```
28         queryset = queryset.filter(  
29             day__icontains=request.GET.get('search'))  
30  
31         context = {'recipes': queryset}  
32         return render(request, 'recipe.html', context)  
33  
34     #Update the recipes data  
35     @login_required(login_url='/login/')  
36     def update_recipe(request, id):  
37         queryset = Recipe.objects.get(id=id)  
38  
39         if request.method == 'POST':  
40             data = request.POST  
41             day = data.get('day')  
42             name = data.get('name')  
43             description = data.get('description')  
44  
45             queryset.day = day  
46             queryset.name = name  
47             queryset.description = description  
48             queryset.save()  
49             return redirect('/')  
50  
51         context = {'recipe': queryset}  
52         return render(request, 'update_recipe.html', context)  
53  
54     #delete the recipes data  
55     @login_required(login_url='/login/')  
56     def delete_recipe(request, id):  
57         queryset = Recipe.objects.get(id=id)  
58         queryset.delete()  
59         return redirect('/')  
60  
61     #login page for user  
62     def login_page(request):  
63         if request.method == "POST":  
64             try:  
65                 username = request.POST.get('username')  
66                 password = request.POST.get('password')  
67                 user_obj =  
User.objects.filter(username=username)  
68                 if not user_obj.exists():  
69                     messages.error(request, "Username not  
found")
```

```
70         return redirect('/login/')
71     user_obj = authenticate(username=username,
password=password)
72     if user_obj:
73         login(request, user_obj)
74         return redirect('recipes')
75     messages.error(request, "Wrong Password")
76     return redirect('/login/')
77 except Exception as e:
78     messages.error(request, "Something went wrong")
79     return redirect('/register/')
80 return render(request, "login.html")
81
82 #register page for user
83 def register_page(request):
84     if request.method == "POST":
85         try:
86             username = request.POST.get('username')
87             password = request.POST.get('password')
88             user_obj =
User.objects.filter(username=username)
89             if user_obj.exists():
90                 messages.error(request, "Username is
taken")
91                 return redirect('/register/')
92             user_obj =
User.objects.create(username=username)
93             user_obj.set_password(password)
94             user_obj.save()
95             messages.success(request, "Account created")
96             return redirect('/login')
97         except Exception as e:
98             messages.error(request, "Something went wrong")
99             return redirect('/register')
100     return render(request, "register.html")
101
102 #logout function
103 def custom_logout(request):
104     logout(request)
105     return redirect('login')
106
107 #Generate the Bill
108 @login_required(login_url='/login/')
109 def pdf(request):
```

```
110     if request.method == 'POST':
111         data = request.POST
112         day = data.get('day')
113         name = data.get('name')
114         description = data.get('description')
115
116         Recipe.objects.create(
117             day = day,
118             name=name,
119             description=description,
120
121         )
122         return redirect('pdf')
123     queryset = Recipe.objects.all()
124
125     if request.GET.get('search'):
126         queryset = queryset.filter(
127             day__icontains=request.GET.get('search'))
128
129     context = {'recipes': queryset}
130     return render(request, 'pdf.html', context)
```

Creating GUI

login.html: Below, HTML code is a concise Bootstrap-based login form for a job portal, featuring input fields for username and password. Success messages are displayed using Bootstrap's alert, and a link is included for users to create a new account.

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/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpUuCOML
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 <body><br><br><br><br>
14
15
16     <br><br>
17
18     <div class="container bg-white col-md-2 card shadow p-3
" id="log">
19         <div class="login-form">
20             {% if messages %}
21             {% for message in messages %}
22                 <div class="alert alert-success {{ message.tags
}} mt-4"
23                     role="alert">
24                     {{ message }}
25                 </div>
26             {% endfor %}
27             {% endif %}
28             <form action="" method="post">
29                 {% csrf_token %}
30                 <h4 > Login </h4>
31                 <div class="">
32                     <input type="text" name="username"
placeholder="Username" required
33
34                     >
35                 </div>
36                 <div class="mt-2">
37                     <input type="password" name="password"
placeholder="Password" required>
38
39                 </div>
40                 <div class="mt-2">
41                     <button >Login</button>
42                 </div>
43                 <br>
44             </form>

```

```

45         <p> <a href="{% url 'register' %}" >Create an
46             Account.</a></p>
47     </div>
48 </div>
49
50 </body>
51
52 </html>

```

register.html: The provided HTML code creates a registration form for a job portal using Bootstrap. It includes input fields for username and password, a registration button, and a link to the login page. Bootstrap's alert component is utilized for displaying success messages.

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/cs
s/bootstrap.min.css" rel="stylesheet"
9          integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCO
mLASjC" 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     <body>
16         <br> <br><br><br><br><br>
17
18     <div class="container bg-white mx-auto col-md-2
card shadow p-3">

```



```

19         <div class="login-form">
20             {% if messages %}
21             {% for message in messages %}
22                 <div class="alert alert-success {{
message.tags }}" role="alert">
23                     {{ message }}
24                 </div>
25             {% endfor %}
26             {% endif %}
27             <form action="" method="post">
28                 {% csrf_token %}
29                 <h4 > Register </h4>
30                 <div >
31                     <input type="text"
name="username" placeholder="Username" required>
32                 </div>
33
34                 <div class="mt-2">
35                     <input type="password"
name="password" placeholder="Password" required>
36                 </div>
37                 <div class="mt-2">
38                     <button >Register</button>
39                 </div>
40             </form>
41             <p ><a href="{% url 'login' %}">Log In
</a></p>
42         </div>
43     </div>
44
45 </body>
46
47 </html>

```

recipe.html: The Django template extends a base HTML file and presents a form for adding recipe data. It includes a button to generate a PDF recipe plan and displays a table of existing recipes with options to delete or update. The styling includes a hover effect for link color change.

HTML



```
1 {% extends "base.html" %}
```

```
2 {% block start %}
3
4 <link
  href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bo
  otstrap.min.css" rel="stylesheet">
5 <style>
6 .ok{
7     color: white;
8     text-decoration: none;
9 }
10 .ok:hover{
11     color: white;
12     text-decoration: none;
13 }
14
15 </style>
16
17 <div class="container mt-3 col-6">
18     <br><br>
19     <form class="col-6 mx-auto card p-3 shadow-lg"
  method="post" enctype="multipart/form-data">
20         {% csrf_token %}
21         <h4> Recipe </h4>
22         <hr>
23         <div class="form-group">
24             <label for="exampleInputEmail1">Day-Time </label>
25             <input type="text" name="day" required>
26         </div>
27         <div class="form-group">
28             <label for="exampleInputEmail1">Recipe </label>
29             <input name="name" type="text" required>
30
31
32         </div>
33         <div class="form-group">
34             <label for="exampleInputPassword1">Description
35 </label>
36             <!-- <input name="description" type="text"
  rows="10" cols="50" required> -->
37
38             <textarea name="description" type="text"
  rows="5" cols="30"></textarea>
39
40             </div>
41             <button type="submit" class="">Add Data</button>
```

```

40     </form>
41     <hr>
42     <div class="class mt-5">
43         <form action="">
44             <button > <a href="{% url 'pdf' %}">Generate Plan
45         </a></button>
46     </form>
47
48     <table class="table mt-6">
49         <thead>
50             <tr>
51                 <th scope="col">S.No. </th>
52                 <th scope="col">Day-Time </th>
53                 <th scope="col">Recipe Name </th>
54                 <th scope="col">Description </th>
55                 <th scope="col">Actions</th>
56             </tr>
57         </thead>
58         <tbody>
59             {% for recipe in recipes %}
60             <tr>
61                 <th scope="row">{{forloop.counter}}</th>
62                 <td>{{recipe.day}}</td>
63                 <td> {{recipe.name}}</td>
64                 <td>{{recipe.description}}</td>
65                 <td>
66                     <a href="/delete_recipe/{{recipe.id
67                     }}">Delete </a>
68                     <a href="/update_recipe/{{recipe.id
69                     }}">Update </a>
70                 </td>
71             </tr>
72             {% endfor %}
73         </tbody>
74     </table>
75 </div>
76
77 {% endblock %}

```

update_recipe.html: The Django template, extending a base HTML file, displays a form for updating recipe data. It pre-fills fields with existing data

and allows users to modify day, recipe name, and description. The styling uses Bootstrap, creating a centered card with a shadow effect.

HTML

```
1 {% extends "base.html" %}
2 {% block start %}
3
4 <link
5 href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bo
6 otstrap.min.css" rel="stylesheet">
7 <style>
8
9
10 <div class="container mt-5 col-5">
11
12     <form class="col-6 mx-auto card p-3 shadow-lg"
13     method="post" enctype="multipart/form-data">
14         {% csrf_token %}
15
16         <div class="form-group">
17             <label for="exampleInputEmail1">Day-Time </label>
18             <input type="text" name="day" value="{{recipe.day}}"
19             required>
20         </div>
21         <div class="form-group">
22             <label for="exampleInputEmail1">Recipe </label>
23             <input name="name" type="text" value="
24             {{recipe.description}}"
25             required>
26         </div>
27         <div class="form-group">
28             <label for="exampleInputPassword1">Description
29             </label>
30             <textarea name="description" type="text" rows="5"
31             cols="30" value="{{recipe.description}}"></textarea>
32             <br>
33             <br>
```

```
32     <button type="submit" >Update Data</button>
33 </form>
34
35
36 </div>
37
38 {% endblock %}
```

pdf.html: Below, HTML document defines a Recipe Meal Planner webpage with Bootstrap styling. It includes a table displaying recipe details and a button to generate a PDF using the html2pdf library. The styling features a clean layout with a card container and a green-themed table. JavaScript functionality is added to trigger PDF generation on button click.

HTML

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
7     <title>Recipe Meal Planner</title>
8
9     <!-- Add Bootstrap CSS Link -->
10    <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/css/
    bootstrap.min.css" rel="stylesheet">
11    <!-- Add html2pdf library -->
12    <script
    src="https://cdnjs.cloudflare.com/ajax/libs/html2pdf.js/0.10
    .1/html2pdf.bundle.js"></script>
13    <style>
14        body {
15            background-color: #f8f9fa;
16        }
17
18        .recipe-container {
19            padding: 20px;
20            margin-top: 30px;
21            background-color: #ffffff;
```

```

22         border-radius: 10px;
23         box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
24     }
25
26     .recipe-header {
27         color: black;
28     }
29
30     .recipe-table th,
31     .recipe-table td {
32         text-align: center;
33         border: 1px solid #dee2e6;
34         padding: 8px;
35     }
36
37     .recipe-table th {
38         background-color: #70e78c;
39         color: #fff;
40     }
41
42     .generate-pdf-btn {
43         margin-top: 20px;
44     }
45 </style>
46 </head>
47
48 <body>
49
50     <div class="container recipe-container col-md-8">
51         <div class="card">
52             <div class="card-body">
53                 <h2 class="recipe-header">Recipe Meal
Planner</h2>
54                 <br><br>
55                 <table class="table recipe-table">
56                     <thead class="recipe-table-head">
57                         <tr>
58                             <th>Day-Time</th>
59                             <th>Recipe Name</th>
60                             <th>Description</th>
61                         </tr>
62                     </thead>
63                     <tbody>

```

```

64         {% for recipe in recipes %}
65         <tr>
66             <td>{{recipe.day}}</td>
67             <td>{{recipe.name}}</td>
68             <td>{{recipe.description}}</td>
69         </tr>
70         {% endfor %}
71     </tbody>
72 </table>
73
74     <button class="btn btn-danger generate-pdf-
75 btn" onclick="generatePDF()">Generate PDF</button>
76 </div>
77 </div>
78
79 <!-- Add Bootstrap JS and Popper.js Scripts -->
80 <script src="https://code.jquery.com/jquery-
81 3.5.1.slim.min.js"></script>
82 <script
83 src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist
84 /umd/popper.min.js"></script>
85 <script
86 src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bo
87 otstrap.min.js"></script>
88
89 <script>
90     function generatePDF() {
91         var element = document.querySelector('.recipe-
92 container');
93         html2pdf(element);
94     }
95 </script>
96 </body>
97 </html>

```

base.html: The HTML template serves as a base for Django views, with a dynamic title based on the variable page. It includes a block for content rendering, allowing customization in extending templates.

HTML

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
7     <title>{{page}}</title>
8 </head>
9 <body>
10
11     {% block start %}
12     {% endblock %}
13
14     <script>
15         console.log('Hey Django')
16     </script>
17 </body>
18
19 </html>
```

admin.py : Here we are registering our models.

Python

```
1 from django.contrib import admin
2 from .models import *
3 from django.db.models import Sum
4
5 admin.site.register(Recipe)
```

urls.py : Here, the Django URL patterns include routes for user authentication (login, logout, register), recipe handling (view, update, delete), and a PDF generation endpoint. These paths are associated with corresponding views from the 'home' app.

Python

```
1 from django.contrib import admin
2 from django.urls import path
```



```
3  from home import views
4
5  urlpatterns = [
6      path('logout/', views.custom_logout, name="logout"),
7      path('pdf/', views.pdf , name='pdf'),
8      path('admin/', admin.site.urls),
9      path('login/' , views.login_page, name='login'),
10     path('register/', views.register_page, name='register'),
11
12     path('', views.recipes, name='recipes'),
13     path('update_recipe/<id>', views.update_recipe,
14         name='update_recipe'),
15     path('delete_recipe/<id>', views.delete_recipe,
16         name='delete_recipe'),
17 ]
```

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

Register

[Log In](#)

Login

[Create an Account.](#)

Recipe

Day-Time

Recipe

Description

[Generate Plan](#)

S.No.	Day-Time	Recipe Name	Description	Actions
-------	----------	-------------	-------------	---------

Day-Time

Recipe

Description

Update Data

Recipe Meal Planner

Day-Time	Recipe Name	Description
Monday-Lunch	Poha	Poha, Sev
Monday- Dinner	Dal-Chawal	Dal, Rice, Salt, Vessals etc.

Generate PDF

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 to build sophisticated web applications efficiently. Ready to enhance your web development journey? Enroll now and unlock your potential with Django!

S soura...



[Previous Article](#)[Next Article](#)[Portfolio Showcase using Django](#)

Similar Reads

Build a Recipe App using MVVM Architecture with Kotlin in Android

In this article, we will make a recipe app that displays a list of Indian recipes using the retrofit library and MVVM architecture. Model — View — ViewModel (MVV...

7 min read

Recipe Recommendation System Using Python

In this article, we will explore how to build a Recipe Recommendation System using Streamlit and OpenAI. We will create the GUI using the Streamlit library...

2 min read

Recipe Generator using Next.js

In this tutorial, we'll create a Recipe Generator App using Next.js, a React framework that allows users to generate random recipes. This type of applicatio...

4 min read

How to create a food recipe app using ReactJS ?

We are going to make a food recipe app using React.js. Pre-requisite:React hooksReact componentsJavaScript ES6API CSSApproach: Here in this app we...

4 min read

Recipe Finder using ReactJS

In this project article, we will be creating a recipe finder application using the React library. We have given the application the name "GeeksforGeeks Recipe...

5 min read

Daily Activity Planner App using MERN Stack

This project is a web-based Daily Planner application built using React for the frontend and Node.js with Express for the backend. It allows users to schedule...

7 min read

Workout Planner using MERN Stack

With a busy schedule, it is very important to prioritize our health and fitness. In this article, a walkthrough to creating a Workout Planner using MERN (Mongod...

15+ min read

Recipe Manager with MERN Stack

The Recipe Manager is a popular MERN Stack-based project used for managing various recipes and creating, editing, or deleting recipes. This article will cover t...

13 min read

Design a Recipe App in HTML CSS & JavaScript

We will create an attractive and styled application that gives the recipe of food, and dishes entered by the user. The user needs to enter the required food name...

6 min read

Project Idea | (Trip Planner)

This project is basically an application which helps friends, colleagues or relatives who live at far off places (generally in another country) plan a trip together to a...

2 min read

Article Tags :

[Django](#)

[Geeks Premier League](#)

[Project](#)

[Python](#)

[+2 More](#)

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
GeeksforGeeks Community

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

Languages

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

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

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

Social Science

English Grammar

Commerce

World GK

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++

Web Development

Data Science

CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved