



Build a Django Application to Perform CRUD Operations

Last Updated : 24 Sep, 2024

This project provides a comprehensive system for managing recipe data, including the ability to create, view, edit, and delete recipes. It demonstrates the fundamental operations of a [CRUD](#) application for recipe management using [Django](#), typically used in web applications for organizing and maintaining recipe collections.

CRUD Operations In Django

the "Recipe Update" project is a part of a CRUD (Create, Read, Update, Delete) application for managing recipes. Here's a summary of its key functionality:

- **Create:** The project enables users to create new recipes by providing a name, description, and an image upload.
- **Read:** Users can view a list of recipes with details, including names, descriptions, and images. They can also search for recipes using a search form.
- **Update:** Users can update existing recipes by editing their names, descriptions, or images. This functionality is provided through a form that populates with the recipe's current details.
- **Delete:** The project allows users to delete recipes by clicking a "Delete" button associated with each recipe entry in the list.

CRUD operations are fundamental to any Django project. To fully master Django's potential and handle more complex operations, the [Django Web Development - Basics to Advance](#) is a comprehensive course that can take your skills to the next level.

Starting the Project Folder

To install Django follow these [steps](#). To start the project use this command

```
django-admin startproject core
cd core
```

To start the app use this command

```
python manage.py startapp recipe
```

Setting up Necessary Files

setting.py: After creating the app we need to register it in **settings.py** in the `installed_apps` section like below

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.staticfiles",
    "recipe",
]
```

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

models.py: The code defines a [Django model](#) named `Recipe` that represents recipes. It includes fields for the user who created the recipe, the recipe's name, description, image, and the number of times the recipe has been viewed. The user field is linked to the built-in user model and can be null, allowing for recipes without a specific user.

Python



```
from django.db import models
from django.contrib.auth.models import User
class Recipe(models.Model):
    user = models.ForeignKey(User, on_delete=models.SET_NULL, null=True,
blank=True)
    recipe_name = models.CharField(max_length=100)
    recipe_description = models.TextField()
    recipe_image = models.ImageField(upload_to="recipe")
    recipe_view_count = models.PositiveIntegerField(default=1)
```

views.py: The code is part of a Django web application for managing recipes. It includes functions for:

1. Displaying and creating recipes, with the ability to filter recipes by name.
2. Deleting a specific recipe.
3. Updating an existing recipe, including the option to change the recipe's image.

These functions are responsible for various recipe-related actions in the application.

Python



```
1 from django.shortcuts import render, redirect
2 from .models import Receipe # Assuming 'Receipe'
  is the correct model name
3 from django.http import HttpResponse
4
5
6 def receipes(request):
7     if request.method == 'POST':
8         data = request.POST
9
10        receipe_image = request.FILES.get('receipe_image')
11        receipe_name = data.get('receipe_name')
12        receipe_description =
data.get('receipe_description')
13
14        Receipe.objects.create(
15            receipe_image=receipe_image,
16            receipe_name=receipe_name,
17            receipe_description=receipe_description,
18        )
19        return redirect('/')
20
21    queryset = Receipe.objects.all()
22
23    if request.GET.get('search'):
24        queryset = queryset.filter(
25            receipe_name__icontains=request.GET.get('search'))
26
```

```
context = {'receipes': queryset}
28 return render(request, 'receipes.html', context)
29
30
31 def delete_receipe(request, id):
32     queryset = Receipe.objects.get(id=id)
33     queryset.delete()
34     return redirect('/')
35
36
37 def update_receipe(request, id):
38     queryset = Receipe.objects.get(id=id)
39
40     if request.method == 'POST':
41         data = request.POST
42
43         receipe_image = request.FILES.get('receipe_image')
44         receipe_name = data.get('receipe_name')
45         receipe_description =
46 data.get('receipe_description')
47
48         queryset.receipe_name = receipe_name
49         queryset.receipe_description = receipe_description
50
51         if receipe_image:
52             queryset.receipe_image = receipe_image
53
54         queryset.save()
55         return redirect('/')
56
57     context = {'receipe': queryset}
58     return render(request, 'update_receipe.html', context)
```

admin.py: We register the models in admin.py file

Python

```
1 from django.contrib import admin
2 from .models import *
3 from django.db.models import Sum
4
```

```
admin.site.register(Receipe)
```

Creating GUI for app

recipes.html: First we created the recipes.html file this Django template code is a part of a web page for adding, searching, displaying, updating, and deleting recipes. Here's a simplified explanation.

HTML

```
1 {% extends "base.html" %}
2 {% block start %}
3
4
5
6 <link
  href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bo
  otstrap.min.css" rel="stylesheet">
7 <style>
8   .text{
9     color: green;
10    font-weight: bold;
11  }
12 </style>
13
14 <div class="container mt-5">
15
16   <form class="col-6 mx-auto card p-3 shadow-lg"
  method="post" enctype="multipart/form-data">
17     {% csrf_token %}
18     <h2 class="text text-center"> GeeksforGeeks </h2>
19     <br>
20     <h3>Add Receipe</h3>
21     <hr>
22     <div class="form-group">
23       <label for="exampleInputEmail1">Receipe
  name</label>
24       <input name="receipe_name" type="text"
  class="form-control" required>
25     </div>
26     <div class="form-group">
```

```

        <label for="exampleInputPassword1" >Receipe
description</label>
28        <textarea name="receipe_description" class="form-
control" required ></textarea>
29    </div>
30    <div class="form-group">
31        <label for="exampleInputPassword1">Receipe
Image</label>
32        <input name="receipe_image" type="file"
class="form-control" >
33    </div>
34
35    <button type="submit" class="btn btn-success">Add
Receipe</button>
36 </form>
37
38 <hr>
39 <div class="class mt-5">
40     <form action="">
41
42         <div class="max-auto col-6">
43             <div class="form-group">
44                 <label for="exampleInputEmail1">Search
Food</label>
45                 <input name="search" type="text" class="form-
control">
46             </div>
47             <button type="submit" class="btn btn-primary ">
Search</button>
48         </form>
49     </div>
50 </div>
51 <table class="table mt-5">
52     <thead>
53
54         <tr>
55             <th scope="col">#</th>
56             <th scope="col">Receipe name</th>
57             <th scope="col">Receipe Desc</th>
58             <th scope="col">Image</th>
59             <th scope="col">Actions</th>
60         </tr>
61     </thead>
62     <tbody>
63         {% for receipe in receipes %}

```

```

65         <th scope="row">{{forloop.counter}}</th>
66         <td>{{receipe.receipe_name}}</td>
67         <td>{{receipe.receipe_description}}</td>
68         <td>
69              </td>
70         <td>
71             <a href="/delete_receipe/{{receipe.id
72             }}<td>
73                 <a href="/update_receipe/{{receipe.id
74                 }}</td>
75             </td>
76         </tr>
77     {% endfor %}
78 </tbody>
79 </table>
80 </div>
81 {% endblock %}

```

update_receipe.html: This template provides a user-friendly interface for editing the details of a recipe, including its name, description, and image. When a user submits the form, it likely sends the updated data to a [Django view](#) for processing and updating the database record.

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  .text{
9  color: green;
10 font-weight: bold;
11 }
12 </style>

```

```

13     <div class="container mt-5">
14         <form class="col-6 mx-auto card p-3 shadow-lg"
15             method="post" enctype="multipart/form-data">
16             {% csrf_token %}
17             <h2 class="text text-center"> GeeksforGeeks </h2>
18             <h>Update Receipe</h>
19             <hr>
20             <div class="form-group">
21                 <label for="exampleInputEmail1">Receipe name</label>
22                 <input name="receipe_name" value="
23                 {{receipe.receipe_name}}" type="text" class="form-control"
24                 required>
25             </div>
26             <div class="form-group">
27                 <label for="exampleInputPassword1">Receipe
28                 description</label>
29                 <textarea name="receipe_description" value=""
30                 class="form-control"
31                 required>{{receipe.receipe_description}}</textarea>
32             </div>
33             <div class="form-group">
34                 <label for="exampleInputPassword1">Receipe
35                 Image</label>
36                 <input name="receipe_image" type="file" class="form-
37                 control">
38             </div>
39             <button type="submit" class="btn btn-success">Update
40             Receipe</button>
41         </form>
42     </div>
43 {% endblock %}

```

base.html: It is the base HTML file which is extended by all other HTML files.

HTML



```
1 <!DOCTYPE html>
```




```
<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
9     <style>
10         table {
11             width: 80%;
12             margin: 20px auto;
13             border-collapse: collapse;
14         }
15
16         th,
17         td {
18             padding: 10px;
19             text-align: left;
20             border: 1px solid #ccc;
21         }
22
23         th {
24             background-color: #f2f2f2;
25         }
26
27         tr:nth-child(even) {
28             background-color: #f2f2f2;
29         }
30
31         tr:hover {
32             background-color: #ddd;
33         }
34     </style>
35 </head>
36
37 <body>
38
39     {% block start %}
40     {% endblock %}
41
42     <script>
43         console.log('Hey Django')
```

```
        </script>
45    </body>
46
47    </html>
```

urls.py: Setting up all the paths for our function.

Python



```
1  from django.contrib import admin
2  from django.urls import path
3  from receipe import views
4
5  urlpatterns = [
6      path('admin/', admin.site.urls),
7      path('', views.receipes),
8      path('update_receipe/<id>', views.update_receipe,
9           name='update_receipe'),
10     path('delete_receipe/<id>', views.delete_receipe,
11         name='delete_receipe'),
12 ]
```

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

GeeksforGeeks

Add Receipe

Receipe name

Receipe description

Receipe Image

Choose File

No file chosen

Add Receipe

GeeksforGeeks

Update Receipe

Receipe name

dfm

Receipe description

sds dg

Receipe Image

Choose File

No file chosen

Update Receipe

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!



prath...



[Previous Article](#)[Next Article](#)

How to Filter ForeignKey Choices in a Django ModelForm

Similar Reads

Build a To-Do application Using Django, React and Tailwind

This article will guide you in creating a To-Do application using React and Tailwind with the Django Framework. We'll explore the integration of Django,...

6 min read

Build a Clock Application using Django and React

The clock project aims to develop a simple yet functional clock application using modern web technologies. It combines the frontend capabilities of React.js for...

4 min read

How to Perform Query Filtering in Django Templates

Sometimes we may want to filter or modify the list of objects within the template itself to tailor the data being displayed. While filtering should generally be done...

5 min read

Build a Calculator with React, Tailwind, and Django

This article will guide you in creating a calculator using React and Tailwind with the Django Framework. We'll explore the integration of Django, React, and...

6 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

Build a Contact form Using Django, React and tailwind

This article will guide you in creating a Contact form Using Django, React, and tailwind. We'll explore the integration of Django, React, and Tailwind, and go...

5 min read

How to build a URL Shortener with Django ?

Building a URL Shortener, Is one of the Best Beginner Project to Hone your Skills. In this article, we have shared the steps to build a URL shortener using Django...

4 min read

Django Form | Build in Fields Argument

We utilize Django Forms to collect user data to put in a database. For various purposes, Django provides a range of model field forms with various field...

3 min read

Embed pygal charts in Django Application

Suppose we are developing a web application using the Django framework and we have some data and want to visualize it on the webpage We can embed it in...

6 min read

Deploy an ASGI Django Application

ASGI, which stands for Asynchronous Server Gateway Interface, is a big deal in Django. It basically helps Django handle lots of things at once, like requests fro...

5 min read

Article Tags :

[Django](#)[Geeks Premier League](#)[Geeks Premier League 2023](#)

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