Django Project

Project: Creating a form and integrate Google's reCaptcha system to the Django site/form.

Here's a brief overview of how you can integrate **reCAPTCHA** using the "**django-recaptcha**" library:

1. Installing, creating and activating a virtual environment.
2. Installation: Install the "django-recaptcha" library using pip:

   pip install django-recaptcha

   It provides wrapper functions and template tags that handle the rendering of the reCAPTCHA widget, validation of the user response, and communication with the Google reCAPTCHA API, making the integration process faster and simpler.
3. Registration on reCAPTCHA Admin Console: Register your site domain on the reCAPTCHA Admin Console to obtain your reCAPTCHA keys (site key and secret key).
4. Configuration in Django settings: Add your reCAPTCHA keys to your Django project settings file (settings.py).
5. Integration in HTML Template: Add the necessary reCAPTCHA script and input element to your HTML form template. The library typically provides template tags to include the reCAPTCHA widget easily.
6. Backend Validation: When the form is submitted, the reCAPTCHA response token is sent to the Google reCAPTCHA API for verification. You'll need to handle this verification process in your Django view.
7. Processing the Response: After the verification, Google reCAPTCHA API returns a response indicating whether the reCAPTCHA challenge was successful or not. You need to process this response in your backend logic.

Files used in this

1. forms.py: This is where you define your form. You'll use Django's form framework to create your form class, and include the reCAPTCHA field provided by the "django-recaptcha" library.
2. index.html: This is your webpage template. Here, you'll add the reCAPTCHA widget to your form using template tags provided by the "django-recaptcha" library. This widget will display the reCAPTCHA challenge to users.

3. views.py: In this file, you define the logic for handling form submissions. You'll write a view function that receives form data when submitted. Within this function, you'll validate the reCAPTCHA response token sent from the frontend to ensure it's from a real user and not a bot.

4. urls.py: This file manages the URL patterns for your Django project. You'll specify the URL route that corresponds to the view handling the form submission. When a user submits the form, Django knows which view function to call to process the data.

These files work together to add reCAPTCHA protection to your Django form, preventing spam submissions and ensuring that your website stays secure.

Step by Step Process

- ## 1st Step

Create a folder in any drive Ex. Python_With_Django_

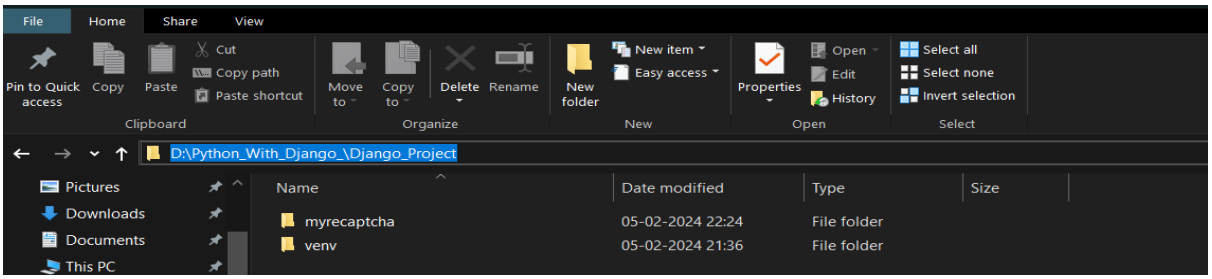
Again create a new folder named Django_Project within Python_With_Django_ folder.

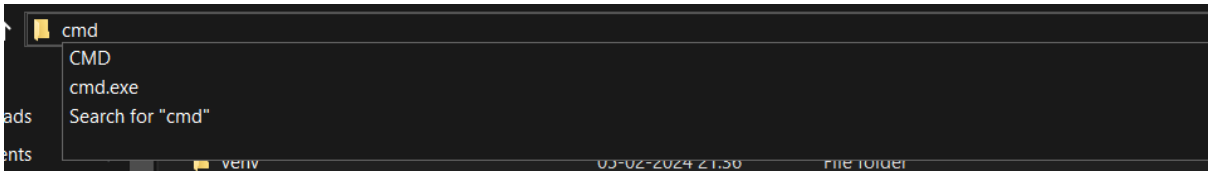
Path: D:\Python_With_Django_\Django_Project

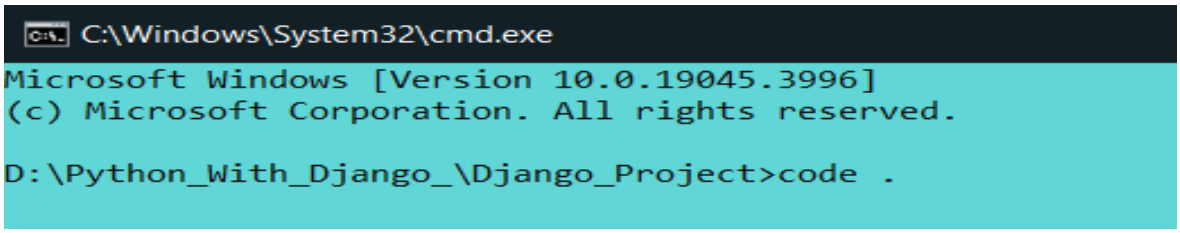
- ## 2nd Step

Open **D:\Python_With_Django_\Django_Project** (Path) on vs code.

1. **Using Cmd:** Open the original file location → Go to the path area (present above the screen) → Select the path and remove it and write cmd and press enter → In cmd write **"code ."**







or

2. **Using VS Code:** Open VS code → Go to File Tab → Select Open Folder → Select the folder **Django_Project**

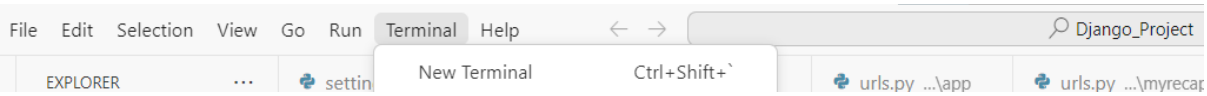
Also in VS code→ Go to File Tab → Select Auto Save option (This will auto save the programs)

Now we are going to add a virtual environment and a project under this Django_Project Folder

- ## Step 3

Setting up virtual environment.

Open terminal by pressing "**Ctrl + Shift + `**" OR



Make sure the selected path in the terminal is **D:\Python_With_Django_\Django_Project**

- **pip install virtualenv** #Download virtual environment
- **virtualenv venv** #Creating virtual environment named venv

OR

- **python -m venv venv**

Use any one process

virtualenv: is the most popular library to create isolated Python environment.

A Python virtual environment (venv) lets you manage separate package installations for different projects. It creates isolated Python environments, ensuring project dependencies do not conflict. Switching projects means creating new virtual environments, maintaining isolation.

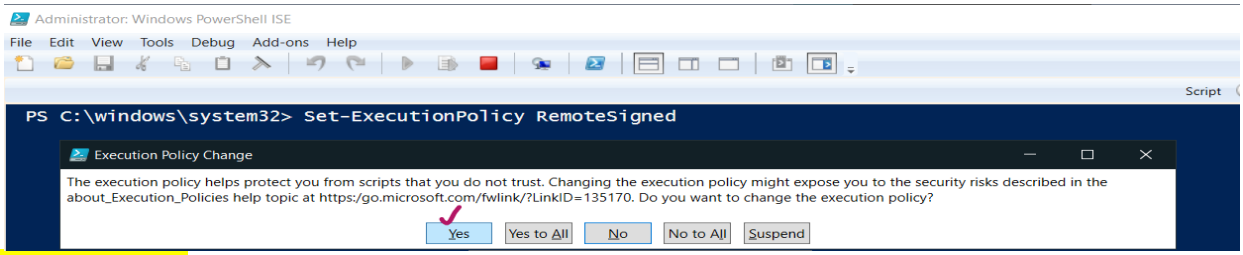
- ## Step 4

Activating the virtualenv.

Open **PowerShell** and "Run as administrator".

PowerShell's script execution policy, which is restricts the activation of scripts on the system. PowerShell has a security feature that controls the execution of scripts to prevent malicious activities. We need to enable this. Open a PowerShell window with administrator privileges and run the following command:

- **Set-ExecutionPolicy RemoteSigned**



Press Yes or Y

Go to VS code and in the terminal... type

- **venv\Scripts\activate**

This will activate the virtual environment

```
PS D:\Python_With_Django_\Django_Project> venv\Scripts\activate
(venv) PS D:\Python_With_Django_\Django_Project>
```

- ## Step 5

Install **django** → needs to be installed within venv

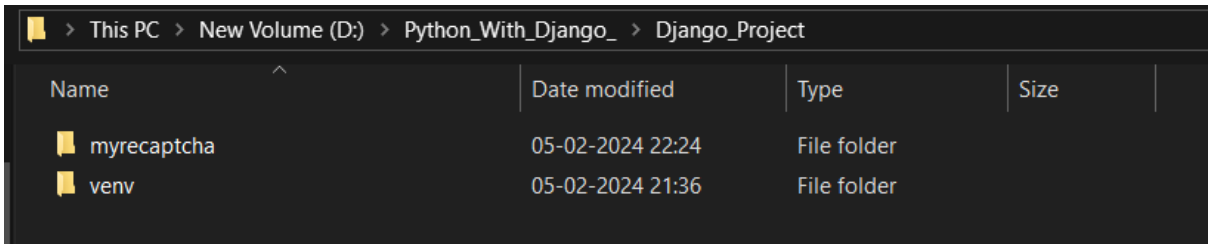
- **pip install Django**

- ## Step 6

Create a project under **D:\Python_With_Django_\Django_Project**

- **django-admin startproject myrecaptcha**

Here "**myrecaptcha**" is the project name.



Now two folders are created within **Django_Project**

- **myrecaptcha** is the project. Here we are going to make all the edits (adding and modifying files).
- **venv** is the virtual environment which will be running in the background while we work with myrecaptcha project. No edits will be made in venv folder.

- ## Step 7

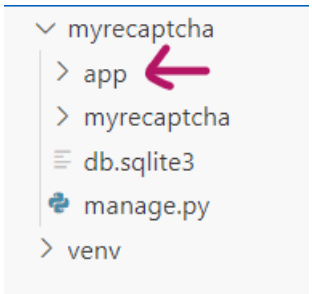
Move to myrecaptcha folder by using the cd command (change directory).

➢ **cd myrecaptcha**

Path: D:\Python_With_Django_\Django_Project\ myrecaptcha

```
(venv) PS D:\Python_With_Django_\Django_Project> cd myrecaptcha
(venv) PS D:\Python_With_Django_\Django_Project\myrecaptcha>
```

- ## Step 8



Now we need to create an app within the project (i.e, myrecaptcha)

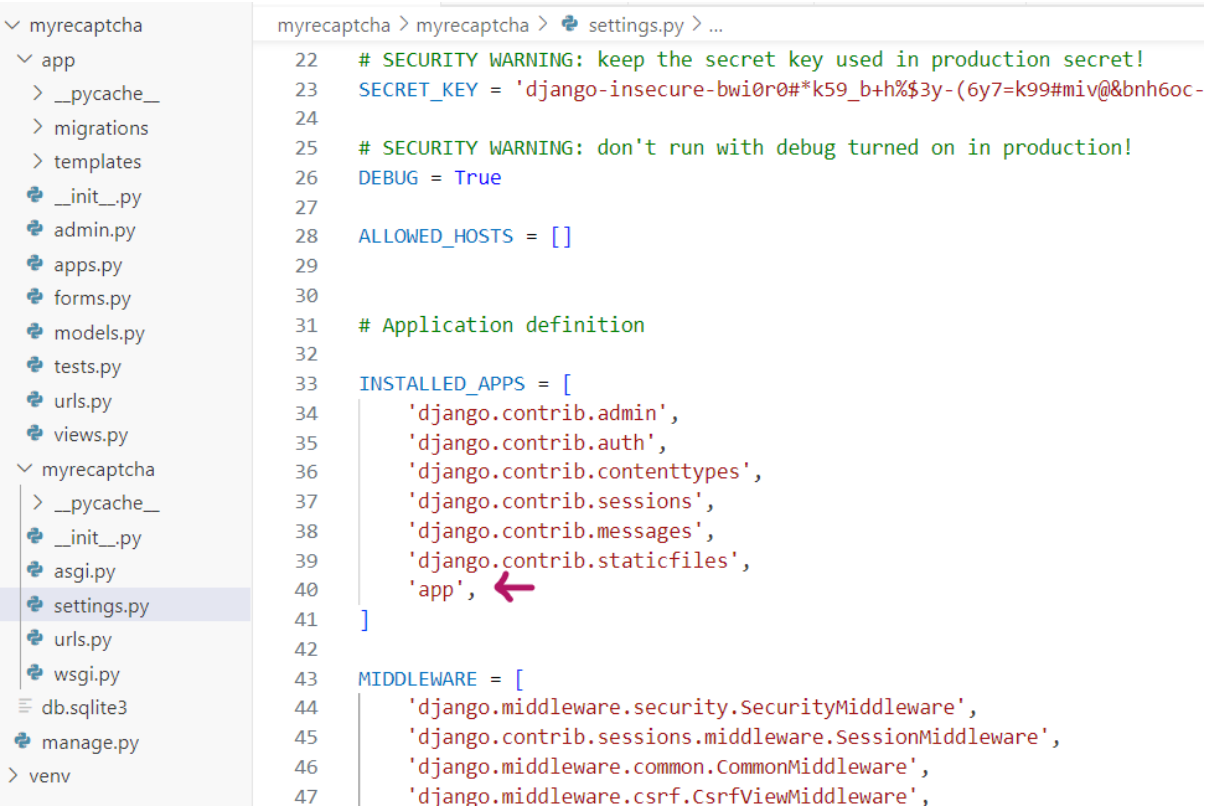
➢ **python manage.py startapp app**

Here **app name** is "**app**". We can also name it something else.

- ## Step 9

Adding the **app** to the **settings.py** file.

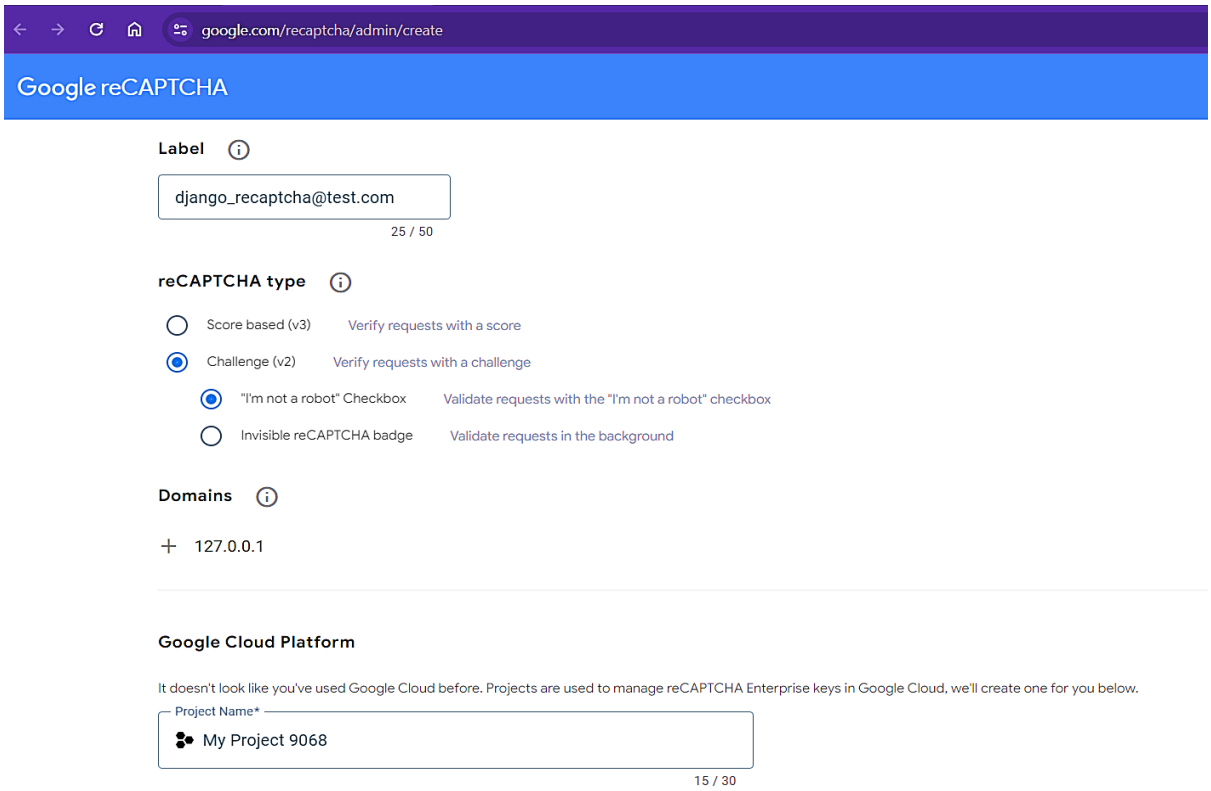
The `app` is added to `INSTALLED_APPS` to register it as a Django application, allowing its functionalities to be utilized.



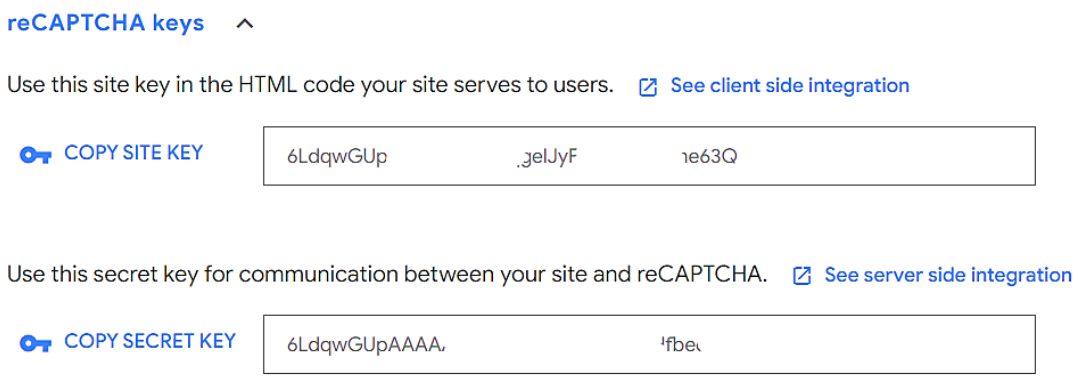
- ## Step 10

Registering to Google reCAPTCHA Admin Console.

First you need register your site on the [reCaptcha admin console](). In the domains section add **127.0.0.1** since we are testing it out locally.



Then click **I agree** and **Submit**



These are API keys (Application Programming Interface).

Copy the site key and secret key (Within double-inverted commas → “ ”) into **settings.py** as follows:

RECAPTCHA_PUBLIC_KEY = Your_Site_Key

RECAPTCHA_PRIVATE_KEY = Your_Secret_key

```
myrecaptcha
  > app
  > myrecaptcha
    > __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
    db.sqlite3
    manage.py
  > venv
```

```
myrecaptcha > myrecaptcha > settings.py > ...
121
122  # Default primary key field type
123  # https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field
124
125  DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
126
127  RECAPTCHA_PUBLIC_KEY = "6LdqwGUpAAAA[illegible]63Q"
128  RECAPTCHA_PRIVATE_KEY = "6LdqwGUpA[illegible]OL2z"
```

The **RECAPTCHA_PUBLIC_KEY** and **RECAPTCHA_PRIVATE_KEY** are configuration variables used for integrating Google reCAPTCHA with your Django application.

- **RECAPTCHA_PUBLIC_KEY:** It's used on the client-side to render the reCAPTCHA widget and verify the user's response.
- **RECAPTCHA_PRIVATE_KEY**: It's used on the server-side to communicate with Google's reCAPTCHA service and verify the user's response.

When you add these keys to your Django **settings.py** file, it allows your Django application to interact with the reCAPTCHA service. This integration helps protect your forms from spam and abuse by verifying that the form submissions are made by humans rather than automated bots. The reCAPTCHA service presents challenges (such as image recognition or checkbox ticking) to users, and based on their responses, it determines whether they are human or not.

- ## Step 11

Installing and using third-party library called **django-recaptcha** which simplifies integrating Google reCAPTCHA into Django projects for spam protection.

➢ **pip install django_recaptcha**

```
(venv) PS D:\Python_With_Django_\Django_Project\myrecaptcha> pip install django_recaptcha
```

- ## Step 12

Adding the **django_recaptcha** to the **settings.py** file.

```
myrecaptcha
  > app
  > myrecaptcha
    > __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  db.sqlite3
  manage.py
  > venv
```

myrecaptcha > myrecaptcha > settings.py > ...

```
30
31  # Application definition
32
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'app',
41      'django_recaptcha',
42  ]
43
```

We add "django_recaptcha" to "**INSTALLED_APPS**" in "settings.py" to enable its functionalities for integrating Google's reCAPTCHA service seamlessly into Django projects improved form security against bots.

- **Step 13**

Creating **forms.py** (file) under **app** (folder)

```
FOLDE...
myrecaptcha
  app
    __pycache__
    migrations
    templates
    __init__.py
    admin.py
    apps.py
    forms.py
    models.py
    tests.py
    urls.py
    views.py
```

forms.py | settings.py | index.html | urls.py ...\app | urls.py ...\myrecaptcha | views.py

myrecaptcha > app > forms.py > ...

```
1   # myapp/forms.py
2   from django import forms
3   from django_recaptcha.fields import ReCaptchaField
4   from django_recaptcha.widgets import ReCaptchaV2Checkbox
5
6   class MyForm(forms.Form):
7       name = forms.CharField(label='Your Name')
8       email = forms.EmailField(label='Your Email')
9       message = forms.CharField(label='Your Message', widget=forms.Textarea)
10      captcha = ReCaptchaField(widget=ReCaptchaV2Checkbox)
11
```

This will help us to create a "**Contact form**" in **forms.py** with Name, Email, Massage, and captcha as the fields.

The captcha field will be rendered as a checkbox field. If you set a different reCAPTCHA type in the admin console, adjust the widget attribute of ReCaptcha Field. By default, it's ReCaptchaV2Checkbox. Available widgets: ReCaptchaV2Checkbox, ReCaptchaV2Invisible, and ReCaptchaV3.

reCAPTCHA type

- ○ Score based (v3) — Verify requests with a score
- ◉ Challenge (v2) — Verify requests with a challenge
  - ◉ "I'm not a robot" Checkbox — Validate requests with the "I'm not a robot" checkbox
  - ○ Invisible reCAPTCHA badge — Validate requests in the background

Code to be used in forms.py:

```
# app/forms.py
from django import forms
from django_recaptcha.fields import ReCaptchaField
from django_recaptcha.widgets import ReCaptchaV2Checkbox

class MyForm(forms.Form):
    name = forms.CharField(label='Your Name')
    email = forms.EmailField(label='Your Email')
    message = forms.CharField(label='Your Message', widget=forms.Textarea)
    captcha = ReCaptchaField(widget=ReCaptchaV2Checkbox)
```

Explanation:

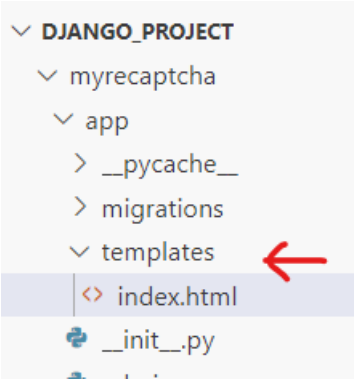
This code defines a Django form named **MyForm** in the file **forms.py** within the **app**.

1. **from django import forms**: Import the **forms** module from Django to create forms.
2. **from django_recaptcha.fields import ReCaptchaField**: Import the **ReCaptchaField** class from the **django_recaptcha** library, which integrates Google reCAPTCHA into Django forms.
3. **from django_recaptcha.widgets import ReCaptchaV2Checkbox**: Import the **ReCaptchaV2Checkbox** widget from the **django_recaptcha** library, specifically used for rendering reCAPTCHA as a checkbox.
4. Define the **MyForm** class, which inherits from **forms.Form**, representing a Django form.
5. **name**, **email**, and **message** are form fields defined using Django's form fields.
    - **forms.CharField** for the name field with a label "Your Name".
    - **forms.EmailField** for the email field with a label "Your Email".
    - **forms.CharField** for the message field with a label "Your Message", and customized with **widget=forms.Textarea** to render it as a textarea.
6. **captcha** is a form field defined using the **ReCaptchaField** class imported from **django_recaptcha.fields**.
    - It's configured to use the **ReCaptchaV2Checkbox** widget, which renders reCAPTCHA as a checkbox.
    - This provides a CAPTCHA challenge to verify that the form submission is done by a human.

This form will render fields for name, email, message, and a reCAPTCHA checkbox for the CAPTCHA challenge. When the form is submitted, it will include the reCAPTCHA response along with other form data for validation.

- **Step 14**

Click on **app** (Select app) → Create folder→ Name it **templates** →select templates→ create new file named **index.html** (index is the name)




```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Contact</title>
    <style>
        body {
            font-family: Times New Roman; /* Change font family */
            background-color: #000000;/* Black background */
            color: #000000; /* Black text color */
            padding: 15px;
            overflow: scroll; /* Prevent background overflow */
            font-size: 20px;
            letter-spacing: 0.1em;
            font-weight: bold;
        }
        h2 {
            color: #00FFD8; /* Green heading color */
            font-size: 60px;
            background-color: rgba(0.8, 0.6, 0.8, 0.7);
            padding: 10px;
            margin-top: 15px;
            font-weight: bold;
            text-align: center; /* Center align the text */
            word-spacing: 0.5em; /* Double space between words */
            letter-spacing: 0.25em; /* Double space between letters */
            text-shadow: 1px 1px 0 #000, -1px -2px 0 #000, 1px -2px 0 #000,
-3px -2px -2px #fff; /* Shadow effect to mimic border */
        }
        form {
            margin-top: -20px;
```

```
            background-color: rgba(139, 247, 244,0.7); /* Semi-transparent
black for the form */
            padding: 25px;
            border-radius: 20px;
        }

        input[type="text"],
        input[type="email"], textarea {
            font-family: Times New Roman;
            font-size: 20px;
            padding: 0.5px;
            margin-bottom: 15px;
            border: 2px solid #fff; /* White border for input fields */
            border-radius: 4px;
            box-sizing: border-box;
            background-color: rgba(0.8, 0.6, 0.8, 0.8); /* Semi-transparent
black background */
            color: #fff; /* White text color */
            position: relative;
            margin: 5px 0;
            border-bottom: 2px solid #fff;
            width: 1300px; /* Adjust the width as needed */
            height: 50px; /* Adjust the height as needed */
        }
        button[type="submit"] {
            background-color: #1cdaab; /* Bright GREEN submit button */
            color: 0000;
            padding: 10px 30px;
            border: 10px;
            border-radius: 15px;
            cursor: pointer;
            font-size: 20px;
            font-weight: bold; /* Making the word "submit" bold */
        }
        button[type="submit"]:hover {
            background-color: #f7ff58; /* yellow on hover */
        }
        .g-recaptcha {
            margin-top: -40px; /* Move the reCAPTCHA box up */
            margin-left: 120px;
            display: inline-grid; /* Ensure it's displayed inline */
            left: 170px;
            up:140px;
            border-radius: 15px; /* Rounded corners */
            border-radius: 7px; /* Rounded corners */
            border: 4px solid black; /* Green border color */

        }
```

```
/* Shaded balls background */
.background-balls {
    position: fixed;
    top: 0;
    left: 0;
    width: 150%;
    height: 150%;
    z-index: -1;
    pointer-events: none;
}
.background-balls span {
    position: absolute;
    width: 145px;
    height: 145px;
    background: #35d4c4; /* Sky blue color */
    border-radius: 50%;
    opacity: 0.4; /* Brighter blue balls */
    animation: balls 25s linear infinite;
}
.background-balls span:nth-child(odd) {
    background: #c43960; /* Bright pink color */
}
.background-balls span:nth-child(1) { left: 10%; top: 20%; }
.background-balls span:nth-child(2) { left: 20%; top: 50%; }
.background-balls span:nth-child(3) { left: 30%; top: 70%; }
.background-balls span:nth-child(4) { left: 40%; top: 30%; }
.background-balls span:nth-child(5) { left: 50%; top: 60%; }
.background-balls span:nth-child(6) { left: 60%; top: 40%; }
.background-balls span:nth-child(7) { left: 70%; top: 80%; }
.background-balls span:nth-child(8) { left: 80%; top: 10%; }
.background-balls span:nth-child(9) { left: 90%; top: 50%; }
.background-balls span:nth-child(10) { left: 5%; top: 80%; }
.background-balls span:nth-child(11) { left: 15%; top: 40%; }
.background-balls span:nth-child(12) { left: 25%; top: 20%; }
.background-balls span:nth-child(13) { left: 35%; top: 60%; }
.background-balls span:nth-child(14) { left: 45%; top: 90%; }
.background-balls span:nth-child(15) { left: 55%; top: 30%; }
.background-balls span:nth-child(16) { left: 65%; top: 70%; }
.background-balls span:nth-child(17) { left: 75%; top: 10%; }
.background-balls span:nth-child(18) { left: 85%; top: 50%; }
.background-balls span:nth-child(19) { left: 95%; top: 80%; }
@keyframes balls {
    0% {
        transform: translateY(0) scale(1);
        opacity: 0.4; /* Brighter blue balls */
    }
    50% {
        transform: translateY(-300px) scale(1.2);
```

```
                opacity: 0.7; /* Brighter blue balls */
            }
            100% {
                transform: translateY(-600px) scale(1);
                opacity: 0.4; /* Brighter blue balls */
            }
        }
    </style>
</head>
<body>
    <h2>CONTACT     FORM</h2>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Submit</button>
    </form>
    <!-- Shaded balls background -->
    <div class="background-balls">
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
    </div>
</body>
</html>
```

- **Step 15**

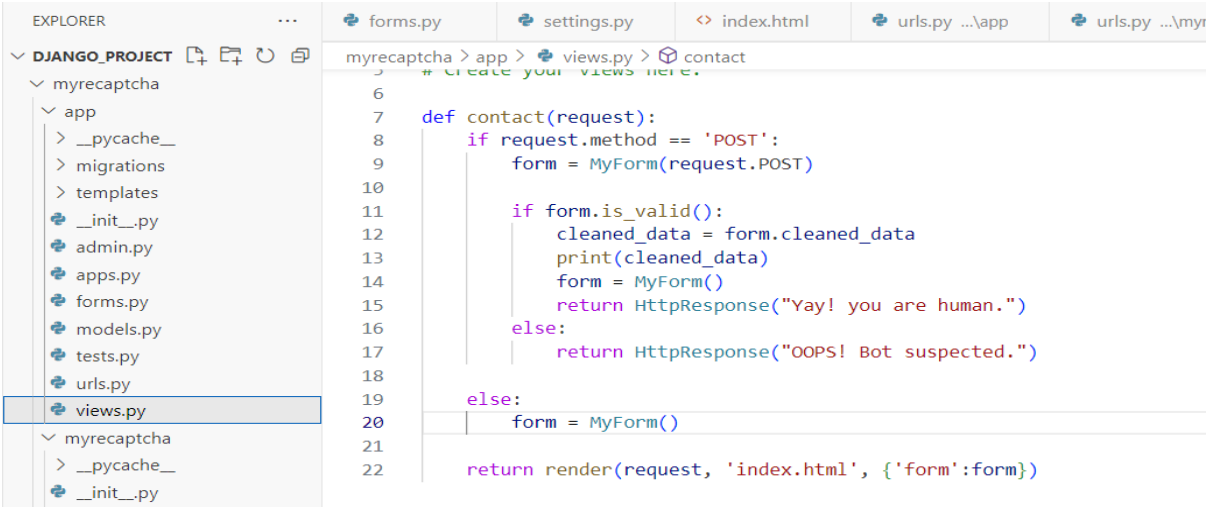
In app → **views.py** → **Put the code**

```
from django.shortcuts import render
from django.http import HttpResponse
from .forms import MyForm

# Create your views here.

def contact(request):
    if request.method == 'POST':
        form = MyForm(request.POST)

        if form.is_valid():
            cleaned_data = form.cleaned_data
            print(cleaned_data)
            form = MyForm()
            return HttpResponse("Yay! you are human.")
        else:
            return HttpResponse("OOPS! Bot suspected.")

    else:
        form = MyForm()

    return render(request, 'index.html', {'form':form})
```

In the views.py file of a Django app, we define a contact function to handle form submissions.

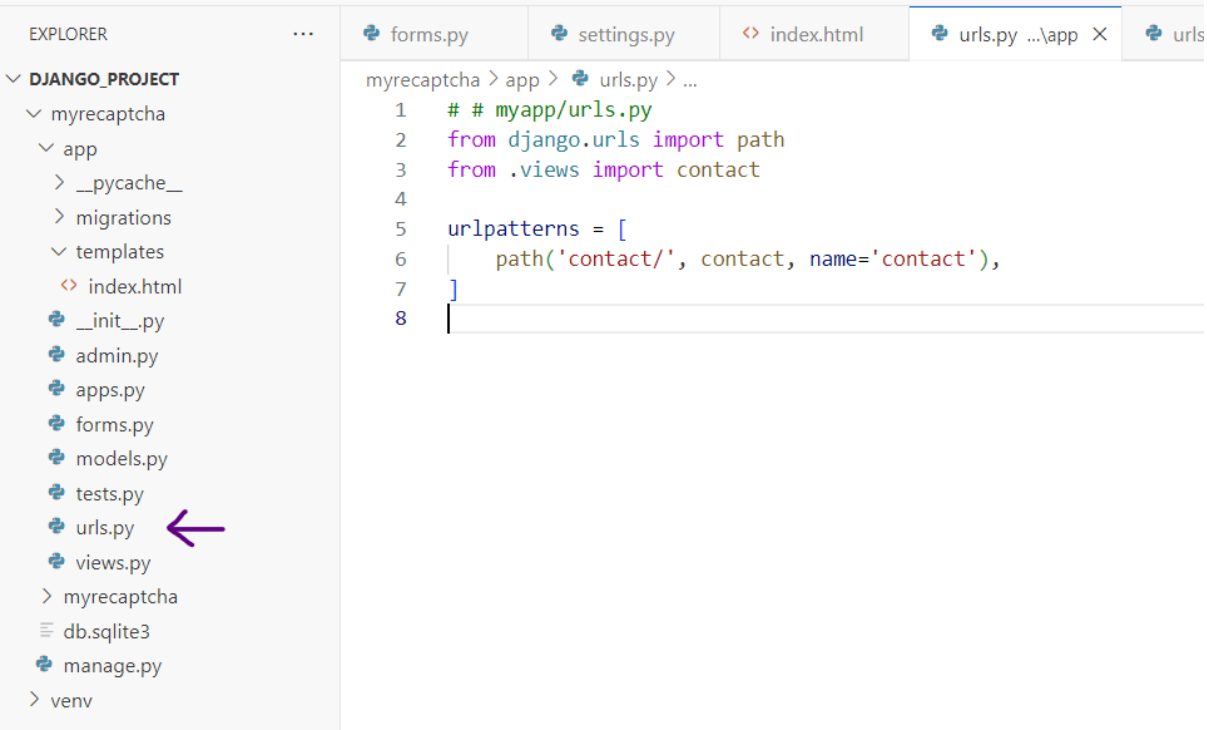
- If the request method is POST, it processes form data using MyForm. If the form is valid, it prints the cleaned data and returns a success message.
- If not, it returns a bot suspicion message. Otherwise, it renders the index.html template with the form.

This code ensures form submission handling and basic bot detection in Django projects.



- **Step 16**

Within app → create file named urls.py → Enter code



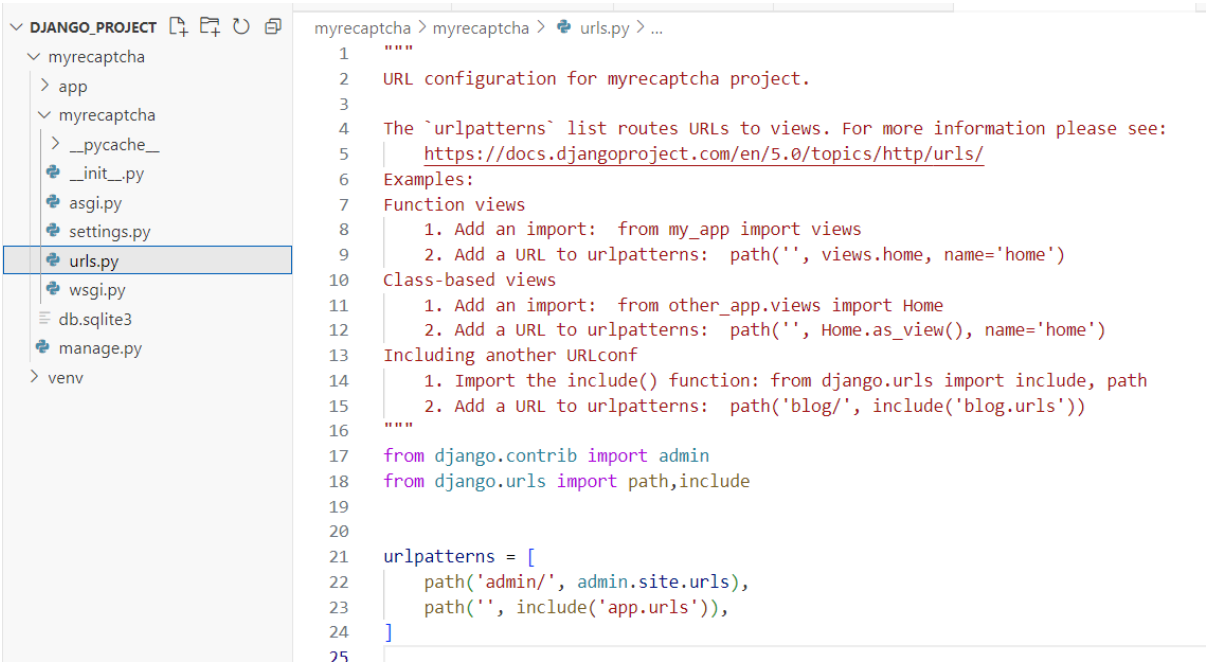

```python
# # myapp/urls.py
from django.urls import path
from .views import contact

urlpatterns = [
    path('contact/', contact, name='contact'),
]
```

In this `**urls.py**` file, URL patterns for the app are defined. It imports the `contact` view from `**views.py**`. A single URL pattern is created for the `contact` view, mapping it to the '**/contact/**' endpoint with a named URL pattern 'contact'.

- **Step 17**

Link App- urls.py with myrecaptcha- urls.py (This is the main body)

Go to myrecaptcha → urls.py→ add the code




```
from django.contrib import admin
from django.urls import path,include


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('app.urls')),
]
```

This `**urlpatterns**` list in the project's main `urls.py` file maps URLs to view functions. The '/admin/' URL leads to the Django admin site. The empty path ' ' includes URLs from the 'app' application, directing requests to its respective `urls.py` file for further processing.

Steps to Reactivate the virtual environment in VS Code

1. **D:\Python_With_Django_> cd Django_Project**
2. **D:\Python_With_Django_\Django_Project> venv\Scripts\activate**
3. **D:\Python_With_Django_\Django_Project>cd myrecaptcha**
4. **D:\Python_With_Django_\Django_Project\ myrecaptcha> python manage.py runserver**

```
PS D:\Python_With_Django_\Django_Project> venv\Scripts\activate
(venv) PS D:\Python_With_Django_\Django_Project> cd myrecaptcha
(venv) PS D:\Python_With_Django_\Django_Project\myrecaptcha> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
```

Click + Ctrl on the generate link i.e,

http://127.0.0.1:8000/contact/

Here we need to add contact because we are running it locally.