Making Geocoder App using Django Framework and Google Map API

Prerequisites:

Python, PostgreSQL and Sublime Text.

1. Creating Django Geocoder project

Step 1: Install Python

If you don't have Python installed, download and install it from the official Python website.

Step 2: Set Up a Virtual Environment

A virtual environment keeps your project dependencies isolated from the global Python environment. This helps to manage packages and avoid conflicts.

1. Open your terminal or command prompt.
2. Navigate to the directory where you want to create your Django project.
3. Create a virtual environment:

```
# On Windows
python -m venv myenv

# On macOS and Linux
python3 -m venv myenv
```

4. Activate the virtual environment:

```
# On Windows
myenv\Scripts\activate

# On macOS and Linux
source myenv/bin/activate
```

Step 3: Install Django

With your virtual environment active, install Django using pip:

```
pip install django
```

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

```
user@user-virtualbox: ~/geocoder_project
user@user-virtualbox:~$ python3 -m venv myenv
user@user-virtualbox:~$ source myenv/bin/activate
(myenv) user@user-virtualbox:~$ pip install django
```

Step 4: Create a Django Project

Create a new Django project and navigate to the project directory:

```
   django-admin startproject geocoder_project
   cd geocoder_project
```

Step 5: Run the Development Server

1. To open the Django app on PC, run the following commands:

```
ifconfig
ls
cd geocoder_project/
ls
nano settings.py
```

2. Check the ip address and edit the settings.py files as:

```
ALLOWED_HOSTS = ['your_ip_address, 'localhost']
```

Make changes and save.

```
^C(myenv) user@user-virtualbox:~/geocoder_project$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.21.246.83  netmask 255.255.240.0  broadcast 172.21.255.255
        inet6 fe80::e65a:f29e:b2d5:1663  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:59:46:26  txqueuelen 1000  (Ethernet)
        RX packets 182128  bytes 225546574 (225.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 158542  bytes 14479070 (14.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 12986  bytes 6112180 (6.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12986  bytes 6112180 (6.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

(myenv) user@user-virtualbox:~/geocoder_project$ ls
db.sqlite3  geocoder_project/  manage.py*
(myenv) user@user-virtualbox:~/geocoder_project$ cd geocoder_project/
(myenv) user@user-virtualbox:~/geocoder_project/geocoder_project$ ls
```

3. Now, start the development server:

```
python manage.py runserver 0:8000
```

Every time after editing code, save the changes by pressing Ctrl + S

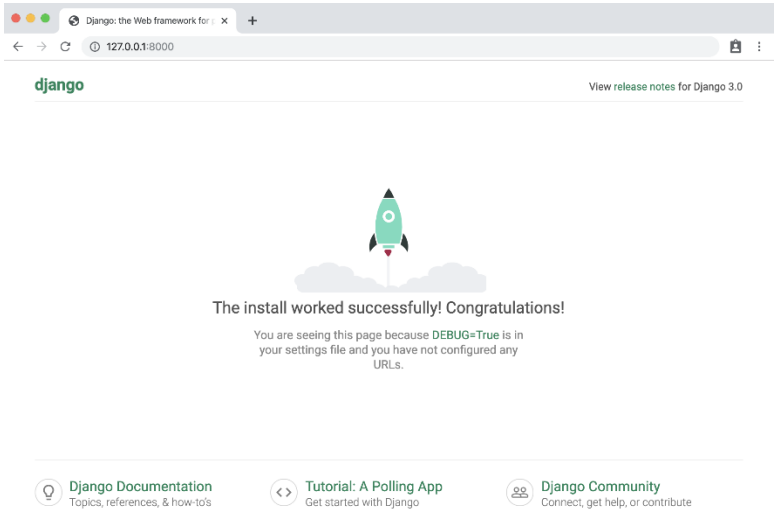
All the text in red need to be modified according to you.

```
(myenv) user@user-virtualbox:~/geocoder_project$ python manage.py runserver 0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, content
types, sessions.
Run 'python manage.py migrate' to apply them.
August 31, 2023 - 05:10:00
Django version 4.2.4, using settings 'geocoder_project.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

4. Open your web browser on PC and go to http://your_ip_address:8000/. You should see the default Django welcome page.



Step 6: Create a Django App

1. Create a new app within your project:

```
python manage.py startapp geocoder_app
```

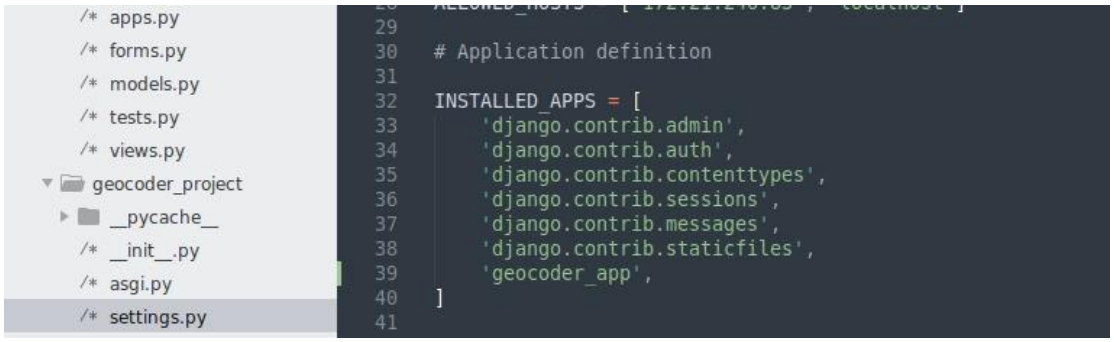
```
(myenv) user@user-virtualbox:~/geocoder_project$ python manage.py startapp geocoder_app
(myenv) user@user-virtualbox:~/geocoder_project$
```

2. Open `geocoder_project` file in Sublime Text, go to `geocoder_project /settings.py` and add your new app to the INSTALLED_APPS list:

```
# geocoder_project /settings.py
INSTALLED_APPS = [
    # ...
    'geocoder_app',
]
```

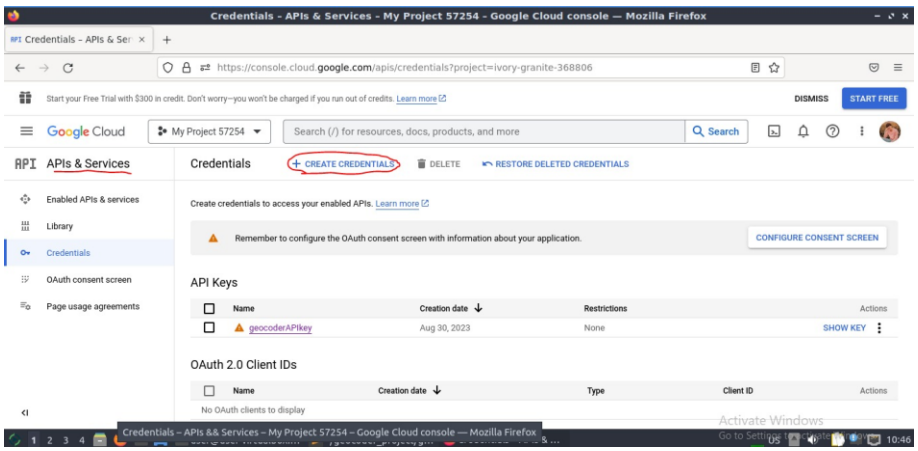
Every time after editing code, save the changes by pressing Ctrl + S

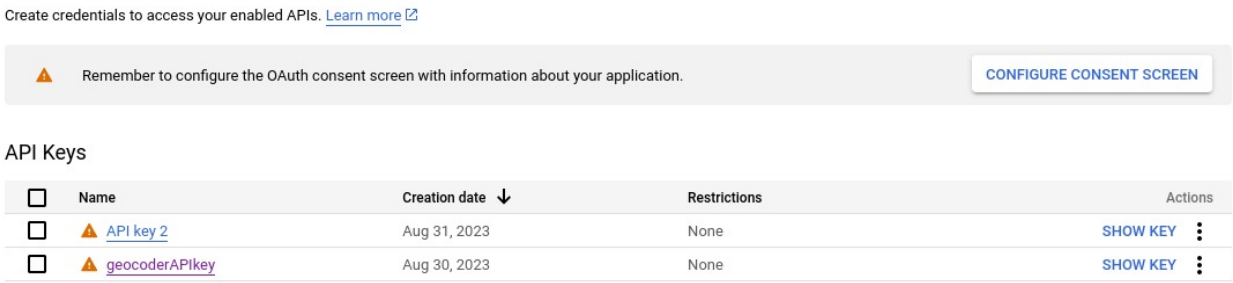
All the text in red need to be modified according to you.



Step 7: Get a Google Maps API Key

1. Go to the Google Cloud Console.
2. Create a new project or select an existing one.
3. Enable the "Geocoding API" for your project.
4. Create an API Key:
    - Go to "APIs & Services" > "Credentials".
    - Click on "Create credentials" and select "API Key".
    - Your API key will be generated. Keep it secure and do not share it publicly.
    - Copy your key.





Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

Step 8: Google Maps API for geocoding service

```
pip install googlemaps
pip install geopy
```

Step 9: Integrate Google Maps Geocoding API

1. Open the settings.py file of your Django project (`geocoder_project/settings.py`).
2. Add your Google Maps API key to the settings.py file:

```
GOOGLE_MAPS_API_KEY = 'your_api_key_here'
```

Step 10: Creating the Address Table/Model

Define a model to store the user's address information and the corresponding latitude and longitude:

```
# geocoder_app/models.py
from django.db import models

class UserAddress(models.Model):
    zipcode = models.CharField(max_length=10)
    house_number = models.CharField(max_length=10)
    street = models.CharField(max_length=100)
    city = models.CharField(max_length=100)
    country = models.CharField(max_length=100)
    latitude = models.DecimalField(max_digits=20,
decimal_places=6, null=True, blank=True)
    longitude = models.DecimalField(max_digits=20,
decimal_places=6, null=True, blank=True)
```

Step 11: Create an address input form

Create a Django form to handle user input for address details in new file `'geocoder_app\forms.py'`:

```
# geocoder_app\forms.py
from django import forms
from .models import UserAddress

class UserAddressForm(forms.ModelForm):
    class Meta:
        model = UserAddress
        fields = ['zipcode', 'house_number', 'street', 'city',
'country']
```

Step 12: Geocoding Logic:

Implement a function to perform geocoding using the googlemaps package and update the latitude and longitude fields in the model in `'geocoder_app\views.py'` :

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

```
# geocoder_app\views.py

from django.shortcuts import render

import googlemaps
from geopy.geocoders import Nominatim
from django.shortcuts import render
from .forms import UserAddressForm
from .models import UserAddress


# geocoding
def get_coordinates(address):
    # Use googlemaps or Nominatim geocoder (geopy) to get latitude
and longitude
    gmaps = googlemaps.Client(key='your_google_map_api')
    geolocator = Nominatim(user_agent='geocoder_app')
    location = geolocator.geocode(address)

    if location:
        return location.latitude, location.longitude
    else:
        return None, None

def user_address_view(request):
    if request.method == 'POST':
        form = UserAddressForm(request.POST)
        if form.is_valid():
            address = f"{form.cleaned_data['house_number']}
{form.cleaned_data['street']}, {form.cleaned_data['city']},
{form.cleaned_data['country']}"
            latitude, longitude = get_coordinates(address)
            form.instance.latitude = latitude
            form.instance.longitude = longitude
            form.save()
    else:
        form = UserAddressForm()

    return render(request, 'geocoder_app/user_address.html',
{'form': form})
```

Step 13: Create a Geocoder Page Template:

Create a template (HTML file) to display the form and show the map. For this, first create a folder named 'templates' in '`geocoder_app`' folder, in which make another folder of same name as app i.e. decoder_app. Now create two html files named, 'base.html' and 'user_address.html' in geocoder_app folder that's inside template folder.

```
# geocoder_app/templates/geocoder_app/base.html
```

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

```
<!DOCTYPE html>
<html>
<head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-
scale=1">
      <title></title>
      <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootst
rap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">

    <!-- for using leaflet map -->
    <link
href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
      <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/leaflet@1.7.1/dist/leaflet.css"
/>
    <script
src="https://cdn.jsdelivr.net/npm/leaflet@1.7.1/dist/leaflet.js"><
/script>

      <!-- for google map apis-->
      <script
src="https://maps.googleapis.com/maps/api/js?key=YOUR_GOOGLE_MAPS_
API_KEY&callback=initMap" async defer></script>
      {% if title %}
            <title>Geocoder App - {{ title }}</title>
      {% else %}
            <title>Geocoder App</title>
      {% endif%}
</head>

<body>
      <!-- Main Body -->
      <div class="body-main">
         {% if messages %}
            {% for message in messages %}
                  <div class="alert alert-{{ message.tags }}">
                        {{ message }}
                  </div>
            {% endfor %}
```

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

```
          {% endif %}
        {% block content %}{% endblock %}
    </div>
</body>
</html>
```

 # geocoder_app/templates/geocoder_app/user_address.html

```
{% extends "geocoder_app/base.html" %}

{% block content %}
  <style>
    .grid{
      padding: 50px 100px;
      display: grid;
      grid-template-columns: 1fr 1fr 1fr;
      grid-template-rows: 2fr;
      grid-template-areas:
       "AddressSection AddressSection AddressSection"

    }

    .AddressSection {
      grid-area: AddressSection;
      margin-bottom: 30px !important;
      margin-right: 20px !important;
      background-color: #e0eefd63;
      width: 100%;
      height: 750px;
      padding: 20px 20px;
      box-sizing: border-box;
      border-radius: 5px;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    }

    .AddressSection h1 {
      text-align: center;
      padding-bottom: 20px;
      font-size: 25px;
      color: rgba(38, 45, 55, 0.9);
    }

    .addressForm {
        margin-bottom: 20px;
    }

    form {
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
        color: rgba(38, 45, 55, 0.9);
        font-size: 12px;
```

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

```
        margin-left: 25px;
    }

    form label {
        width: 40%;
    }

    form input[type="text"],
    form input[type="email"] {
        width: 60 px;
        padding: 8px 40px;
        margin-bottom: 10px;
        border: 1px solid #ccc;
        border-radius: 3px;
        box-sizing: border-box;
    }

    input[type="submit"] {
      background-color: rgba(57, 41, 51, 0.8);
      color: #fff;
      border-color: #fff;
      font-weight: bold;
      padding: 5px 10px;
      margin-top: 30px;
      border: none;
      border-radius: 3px;
      cursor: pointer;
      width: 80px;
      height: 30px;
      display: block;
    }

    input[type="submit"]:hover{
      background-color: rgba(57, 41, 51, 1);
      color: #fff;
      border-color: #326ba8;
      font-weight: bold;
      padding: 5px 20px;
      border: none;
      border-radius: 3px;
      cursor: pointer;
      width: 80px;
      height: 30px;
      box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    }

    .MapArea {
      width: 1050px;
      height: 400px;
      grid-area: AddressSection;
```

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

```
    margin:  0px 30px;
  }

  #map{
    width: 100%;
    height: 100%;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

  .latLong{
    display: flex;
    align-items: center;
    justify-content: space-between;
    background-color: #fff;
    color: black;
    font-weight: bold;
    text-align: left;
    padding: 10px 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  }

  .latLong p{
    margin: 0;
    font-size: 16px;
  }
  .MapArea button {
    font-size: 12px;
    background-color: rgba(57, 41, 51, 0.8);
    color: #fff;
    border-color: #fff;
    font-weight: bold;
    padding: 5px 20px;
    border: none;
    border-radius: 3px;
    cursor: pointer;
    width: 80px; /* Adjust the width as needed */
    height: 30px;
  }

  .MapArea button:hover {
    font-size: 12px;
    background-color: rgba(57, 41, 51, 1);
    color: #fff;
    border-color: #326ba8;
    font-weight: bold;
    padding: 5px 20px;
    border: none;
    border-radius: 3px;
    cursor: pointer;
    width: 80px;
```

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

```
      height: 30px;
      box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    }
  </style>


  <div class="grid">
    <div class="AddressSection">
      <h1>Enter Your Address</h1>
      <div class="addressForm">
          <form method="post">
              {% csrf_token %}
              {{ form.as_p }}
              <input type="submit" value="Submit">
          </form>
      </div>
      {% if form.instance.latitude and form.instance.longitude %}
        <div class="MapArea">
          <div class="latLong">
            <p>Latitude: {{ form.instance.latitude }}, Longitude:
{{ form.instance.longitude }}</p>
            <button id="copyCoordinatesButton">Copy</button>
          </div>
          <div class="map" id="map"></div>
        </div>
      {% endif %}
    </div>

    <script>
      // Display map with the resulting coordinates
      const osm =
L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
        maxZoom: 19,
        attribution: '&copy; <a
href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'
      });
      const map = L.map('map', {
        center: [{{ form.instance.latitude }}, {{
form.instance.longitude }}],
        zoom: 15,
        layers: [osm]
      });
      // Add a marker and bind a popup to it
      const marker = L.marker([{{ form.instance.latitude }}, {{
form.instance.longitude }}]).bindPopup('Latitude: ' + {{
form.instance.latitude }} + ', Longitude: ' + {{
form.instance.longitude }}).addTo(map);


      // Add a function to copy coordinates to clipboard
```

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

```
      const copyCoordinatesButton =
document.getElementById('copyCoordinatesButton');
      copyCoordinatesButton.addEventListener('click', function() {
        const latitude = {{ form.instance.latitude }};
        const longitude = {{ form.instance.longitude }};
        const coordinatesString = `${latitude}, ${longitude}`;
        copyToClipboard(coordinatesString);
      });

      // Function to copy text to clipboard
      function copyToClipboard(text) {
        const textarea = document.createElement('textarea');
        textarea.value = text;
        document.body.appendChild(textarea);
        textarea.select();
        document.execCommand('copy');
        document.body.removeChild(textarea);
        alert('Coordinates copied to clipboard: ' + text);
      }

      </script>
  </div>
{% endblock content %}
```

Step 14: URLs Configuration:

Add a URL mapping to `urls.py` file that's inside geocoder_project:

```
# geocoder_project/urls.py
from django.contrib import admin
from django.urls import path
from geocoder_app.views import user_address_view

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', user_address_view, name='user_address'),
]
```

Step 15: Save Changes:

1. Open a terminal or command prompt to run the makemigrations command using python manage.py makemigrations and apply migrations to database:

```
python manage.py makemigrations
python manage.py migrate
python manage.py runserver 0:8000
```

Every time after editing code, save the changes by pressing Ctrl + S

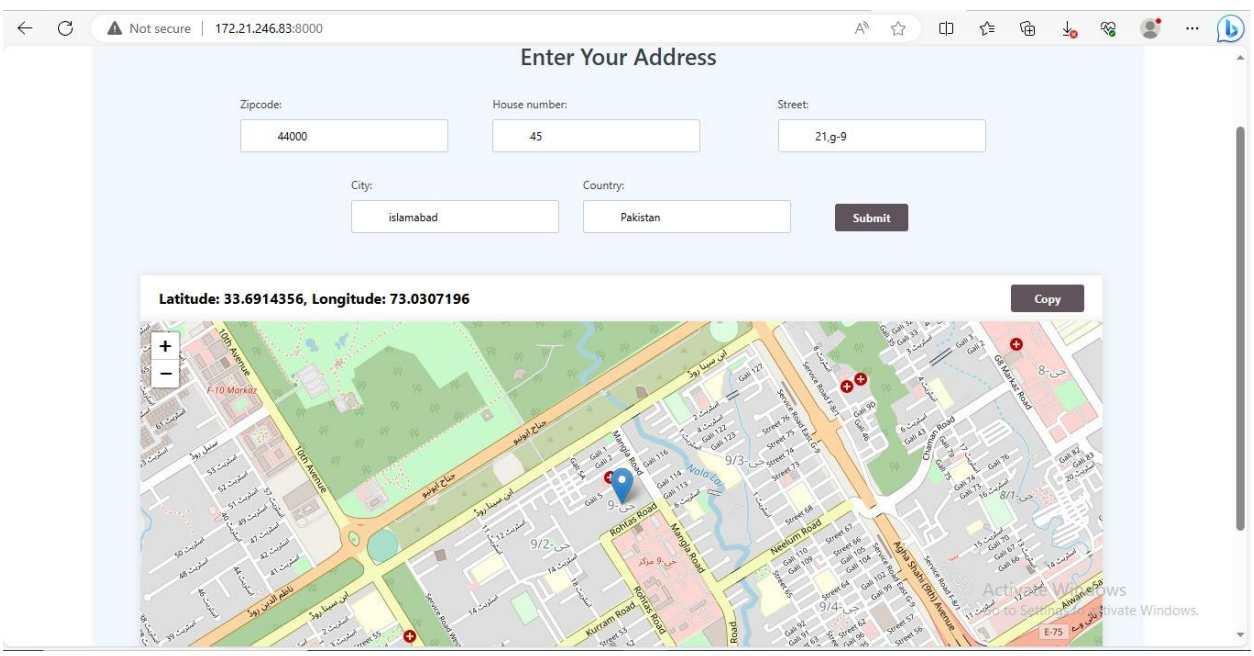
All the text in red need to be modified according to you.

```
(myenv) user@user-virtualbox:~/geocoder_project$ python manage.py makemigrations
Migrations for 'geocoder_app':
  geocoder_app/migrations/0001_initial.py
    - Create model UserAddress
(myenv) user@user-virtualbox:~/geocoder_project$
```

```
(myenv) user@user-virtualbox:~/geocoder_project$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, geocoder_app, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying geocoder_app.0001_initial... OK
  Applying sessions.0001_initial... OK
(myenv) user@user-virtualbox:~/geocoder_project$
```

```
(myenv) user@user-virtualbox:~/geocoder_project$ python manage.py runserver 0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 31, 2023 - 06:13:42
Django version 4.2.4, using settings 'geocoder_project.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```



2. Connect and Configure PostgreSQL in Django Project

Step 1: Create the PostgreSQL Database

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

Make sure you have PostgreSQL installed and running. Create a new database and note down the database name, username, port, and password.

Step 2: Install the Required Packages

Open the terminal and install psychopg2 to connect PostgreSQL to your Django project using command:

```
pip install psycopg2-binary
```

Step 3: Configure Database Postgres Settings

Open your Django project's **settings.py** file and navigate to the DATABASES setting. By default, Django is configured to use SQLite, but we'll change that to use PostgreSQL.

Replace the DATABASES setting with the following code(you can get the name, user, host and port by right clicking on the database and going to properties):

```
# geocoder_project /settings.py

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.postgresql',

        'NAME': 'your_database_name',

        'USER': 'your_database_user',

        'PASSWORD': 'your_database_password',

        'HOST': 'localhost',

        'PORT': '',          # Leave empty to use the default
PostgreSQL port (usually 5432)

    }

}
```

Every time after editing code, save the changes by pressing ctrl + S

All the text in red need to be modified according to you.

Step 4: Apply Django Migrations for Database Tables

Run following command to create necessary tables in PostgreSQL DB:

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

```
(myenv) user@user-virtualbox:~/geocoder_project$ python manage.py makemigrations
No changes detected
(myenv) user@user-virtualbox:~/geocoder_project$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, geocoder_app, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying geocoder_app.0001_initial... OK
  Applying sessions.0001_initial... OK
(myenv) user@user-virtualbox:~/geocoder_project$
```

Step 5: Test Connection

```
python manage.py runserver 0:8000
```

Step 6: Create Trigger Function to add Geometry

1. Create a trigger function to update the geometry automatically upon adding new address each time in `geocoder_app_useraddress` table.

```
create extension postgis;
```

```
SELECT * FROM pg_available_extensions;
```

```
-- creating geom column

ALTER TABLE geocoder_app_useraddress

ADD COLUMN geom geometry(Point, 4326);
```

```
-- Creating function

CREATE OR REPLACE FUNCTION add_geometry()
```

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

```
RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

        NEW.geom = ST_SetSRID(ST_MakePoint(NEW.longitude,
NEW.latitude), 4326);

       RETURN NEW;

END;

$$;
```

```
-- creating trigger

CREATE OR REPLACE TRIGGER insert_geometry_trigger

BEFORE INSERT ON geocoder_app_useraddress

FOR EACH ROW

EXECUTE FUNCTION add_geometry();
```

```
-- viewing all rows in geocoder_useraddress table

select * from geocoder_app_useraddress;
```

```
16      NEW.geom = ST_SetSRID(ST_MakePo
17      RETURN NEW;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 90 msec.
```

```
25
26  -- viewing all rows in geocoder_useraddress table
27  select * from geocoder_app_useraddress;
```

Data Output | Messages | Notifications

	city character varying (100)	country character varying (100)	latitude numeric (20,6)	longitude numeric (20,6)	geom geometry
1	islamabad	Pakistan	33.691436	73.030720	0101000020E610000052D50451F741524077A38FF980D84040

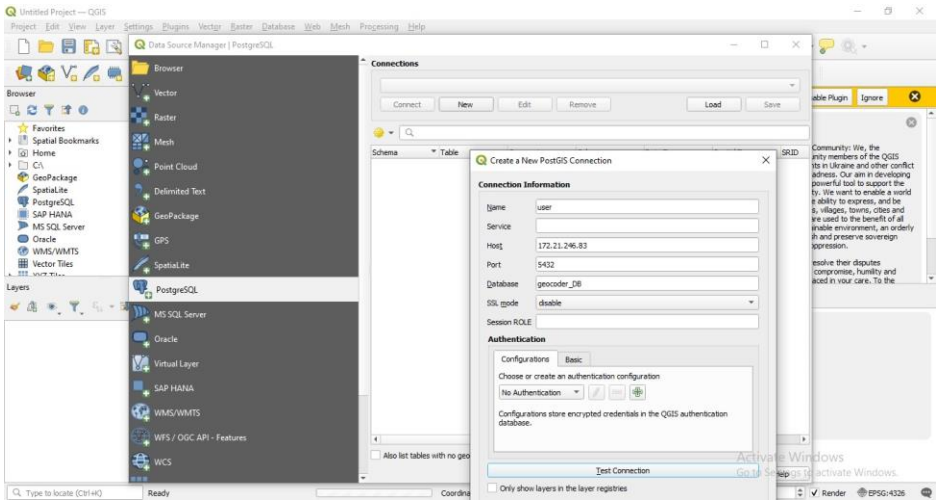
Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

3. Opening Geocoded Table on QGIS

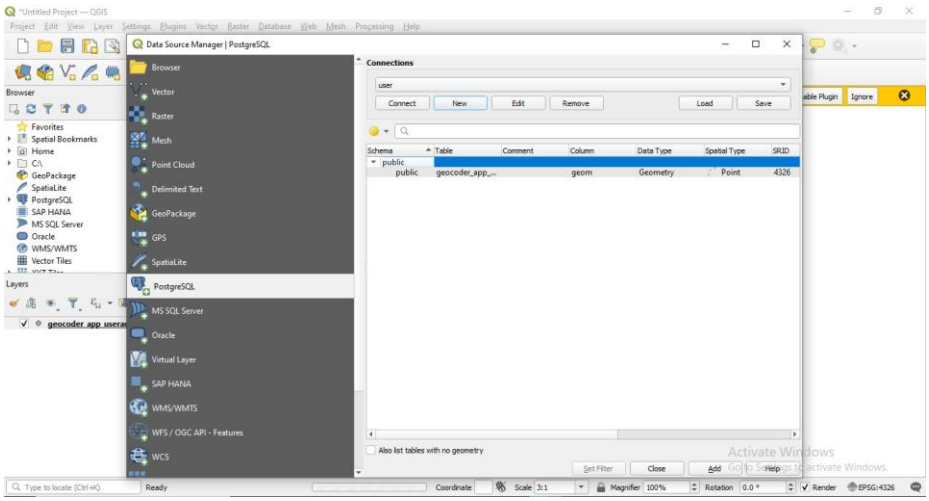
1. Open QGIS, go to Layer->Add Layer->Add PostGIS Layers.

2. Click ‘Connect’ button and add credentials.



3. Click test connection, add username and password click ‘ok’.

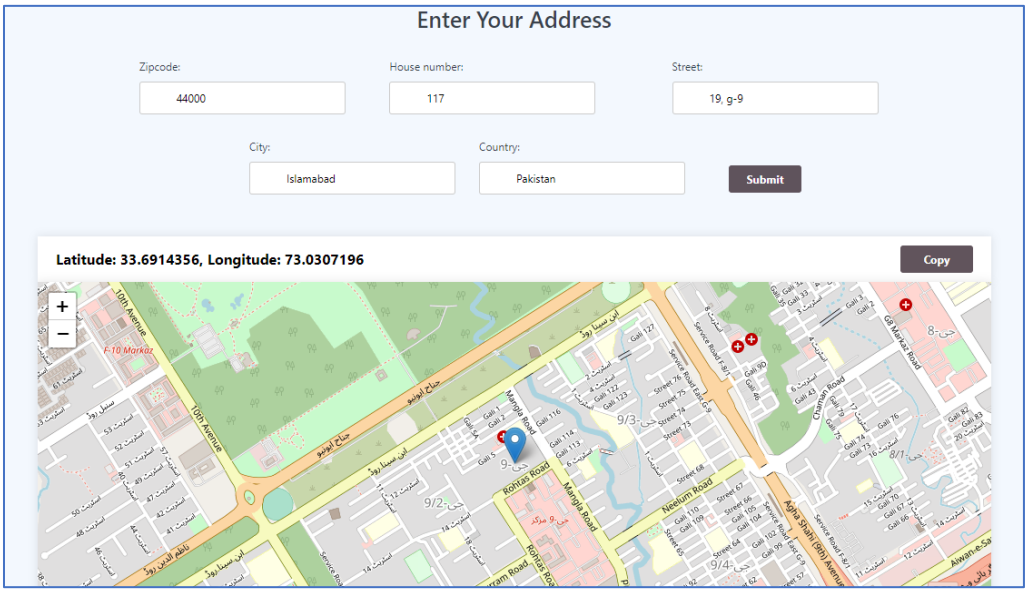
4. Go back, click ‘connect’ again, choose the table ‘geocoder_app_useraddress’ and click ‘add’.

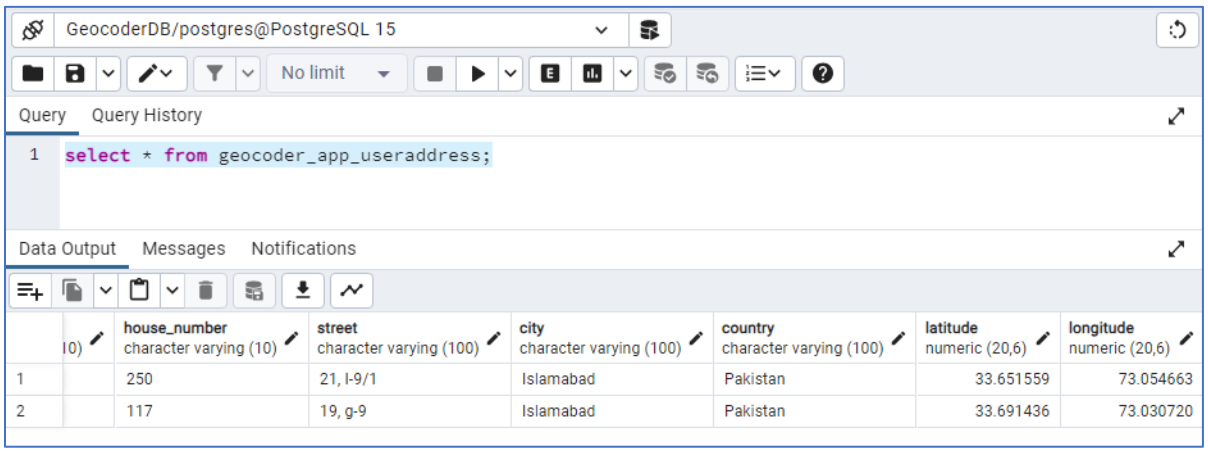


Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.

OUTPUT:





	house_number character varying (10)	street character varying (100)	city character varying (100)	country character varying (100)	latitude numeric (20,6)	longitude numeric (20,6)
1	250	21, I-9/1	Islamabad	Pakistan	33.651559	73.054663
2	117	19, g-9	Islamabad	Pakistan	33.691436	73.030720

Good Luck :)

THE END

Every time after editing code, save the changes by pressing Ctrl + S

All the text in red need to be modified according to you.