

## Exercise # 9

### 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:

```
python3 -m venv myenv
```

4. Activate the virtual environment:

```
source myenv/bin/activate
```

### Step 3: Install Django

With your virtual environment active, install Django using pip:

```
pip install django
```

### Step 4: Create a Django Project

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

```
django-admin startproject geocodper_app  
cd geocoder_app
```

### Step 5: Run the Development Server

1. Start the development server:

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

2. Open your web browser and go to <http://127.0.0.1:8000/>. You should see the default Django welcome page.
3. To open the Django app on PC, edit the `'settings.py'` as:

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

## Step 6: Create a Django App

1. Create a new app within your project:

```
python manage.py startapp geocoder
```

2. Open `geocoder_app/settings.py` and add your new app to the `INSTALLED_APPS` list:

```
INSTALLED_APPS = [  
    # ...  
    'geocoder_app',  
]
```

## 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.

## 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_app/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:

```
nano geocoder_app/models.py
# 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`:
# 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 '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 which make another folder of same name as app i.e. geocoder\_app. Now create two html files named, 'base.html' and 'user\_address.html'.

For base.html

```

<!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/bootstrap.
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"></scr
ipt>
```

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

```

## For user\_addressss.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;
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;
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;
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

    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 your app's `urls.py`:

```

from django.urls import path
from geocoder_app.views import user_address_view

urlpatterns = [
    path('user_address/', 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

```

## 2. Connect and Configure PostgreSQL in Django Project

## Step 1: Create the PostgreSQL Database

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 psycpg2-binary
```

### Step 3: Configure Database Postgres Settings

Open your Django project's **settings.py** file (located inside the main project folder) 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:

```
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)  
  
    }  
}
```

```
}
```

## 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
```

## Step 5: Create Superuser and Test Connection

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

## OUTPUT:

### Enter Your Address

Zipcode:  
44000

House number:  
117

Street:  
19, g-9


City:  
Islamabad

Country:  
Pakistan

Submit

Latitude: 33.6914356, Longitude: 73.0307196

Copy





GeocoderDB/postgres@PostgreSQL 15							
No limit							
Query Query History							
1 select * from geocoder_app_useraddress;							
Data Output Messages Notifications							
	10	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