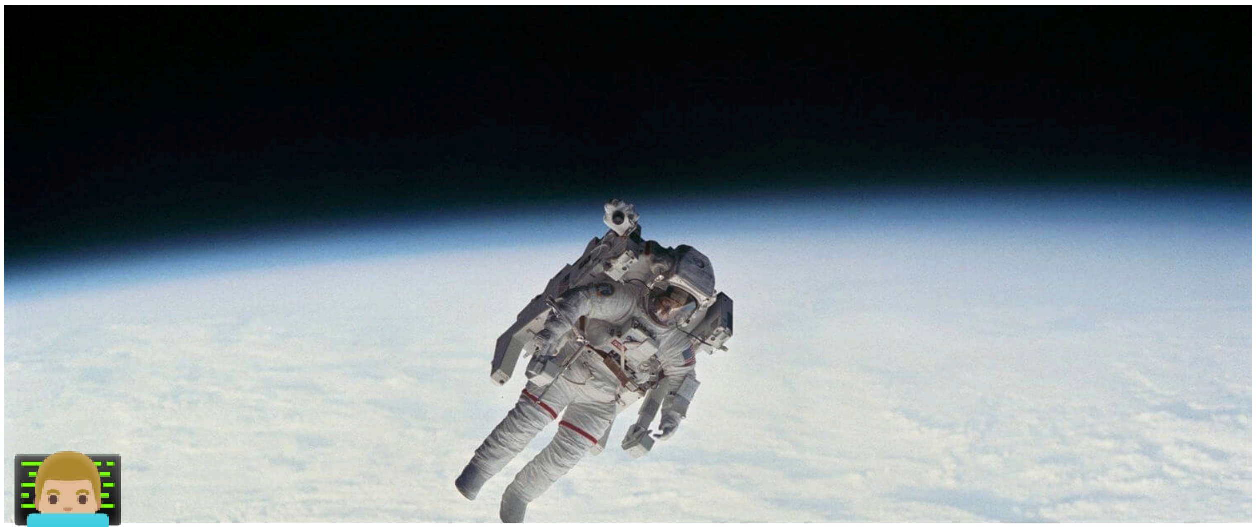


ChatGPT clone Using Django

This is the follow-along notes summary of a ChatGPT clone tutorial that I followed from youtube.

LET’S GET STARTED→

First things first, we’ll go ahead and make a virtual environment and then we’ll activate it.

Install Django in that virtual environment and make a project.

Now, we’ll start a new app and give it a name.

Add the newly created app inside INSTALLED_APPS list in the settings.py file.This is just to let Django know that we’ve created a new app that is a part of this project.

Create the templates folder in the same directory as manage.py file.

Add the provided templates from the Github repository.

In settings.py file under the templates list, the ‘DIRS’:[] is meant to be able to tell django where to look for templates folder.

Next, we create the urls.py file for the app directory.

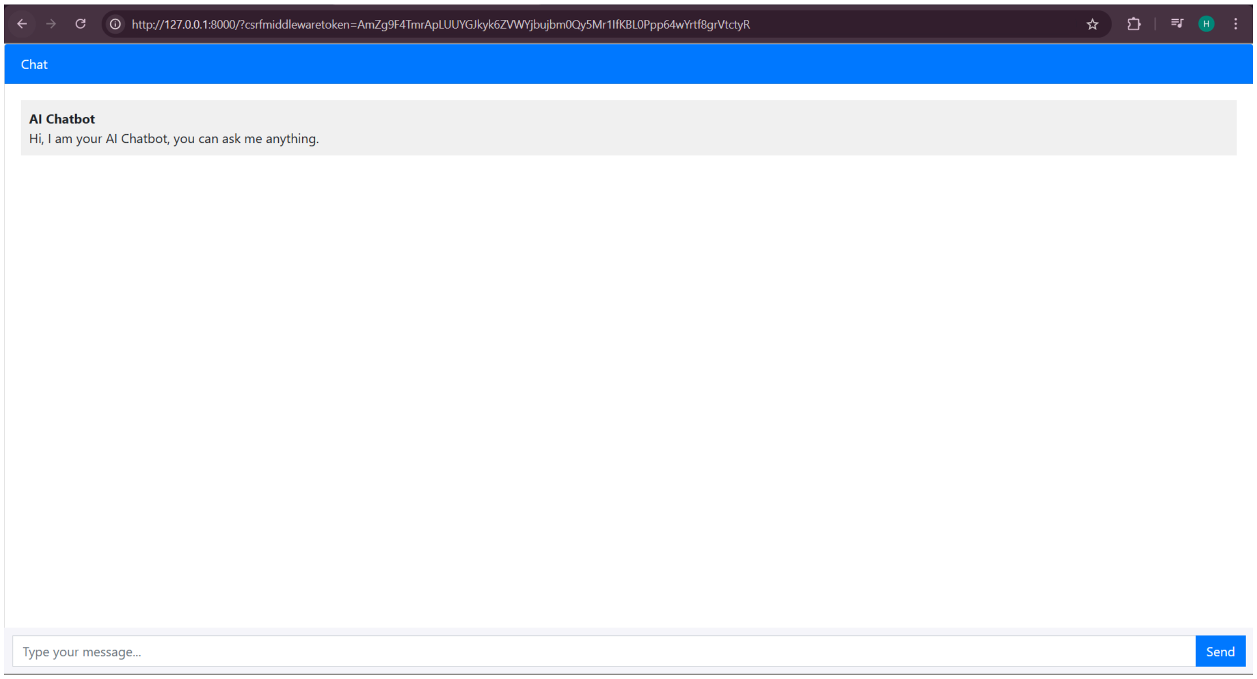
We’ll go ahead and add this to this newly made file-:

```
from django.urls import path, include
from . import views

url_patterns = [
    path('', views.chatbot, name='chatbot'),
]
```

Next, we’ll create a view in the views.py file to match the one we mentioned in urls.py.

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

# Create your views here.
def chatbot(request):
    return render(request, 'chatbot/chatbot.html')
```

Finally, we’ll include the path we made in the app level urls.py file in the project level urls.py file and run the server.Doing all this gets us the basic interface up an running-:


http://127.0.0.1:8000/?csrfmiddlewaretoken=AmZg9F4TmrApLUUYGJkyk6ZVWYjbujbm0Qy5Mr1IfKBL0Ppp64wYrtf8grVtctyR
Chat
Type your message...
Send

AI Chatbot
Hi, I am your AI Chatbot, you can ask me anything.

Then we add some code after the closing of the div tag and enclose that code within script tags. This code would allow us to display messages on the screen after the user has clicked the Send button.

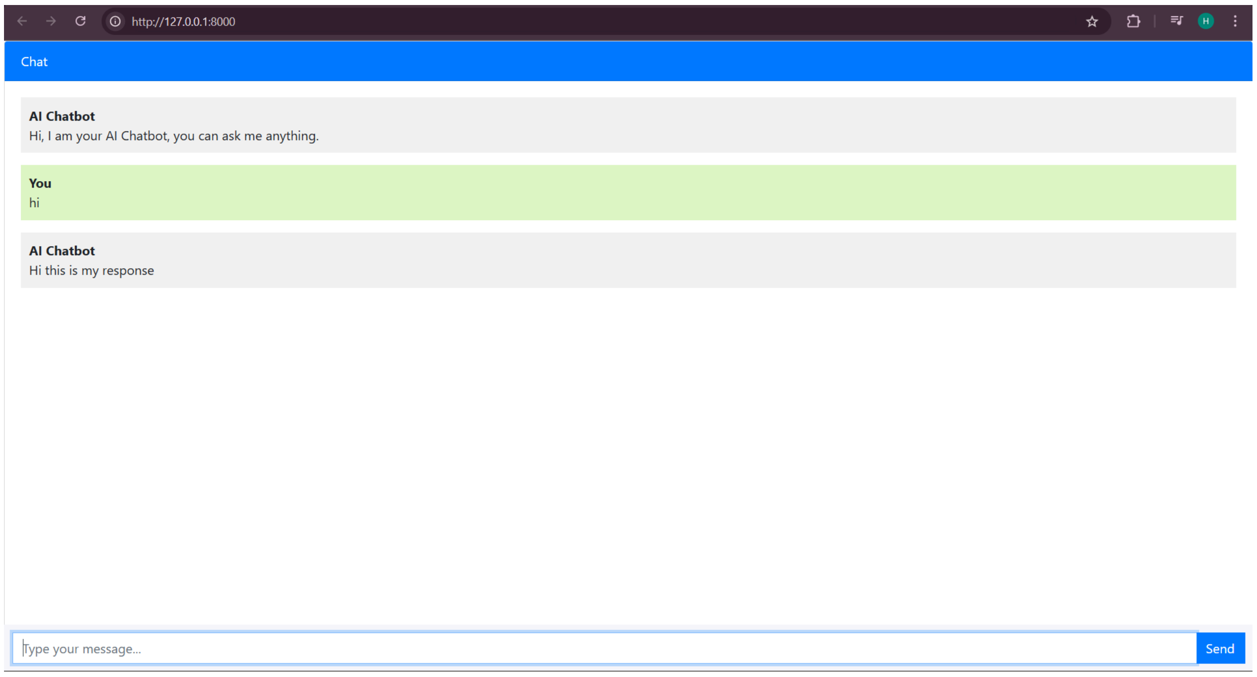
We'll elaborate the entire logic of the chatbot.html file later.

Now, we'll send the input received by the user to the backend and create a view that sends this message to OpenAI API and then gives us a response and then we'll show the response.

```
def chatbot(request):
    if request.method == 'POST':
        message = request.POST.get('message')
        # Here you would typically process the message and generate a respons
e
        response = 'Hi this is my response'
        return JsonResponse({'message': message, 'response': response})
    return render(request, 'chatbot.html')
```

So, in javascript, there's something called fetch which is used to kind of send a request to an API or your backend which in this case would be Django backend.

This is where we are right now→



AI Chatbot
Hi, I am your AI Chatbot, you can ask me anything.

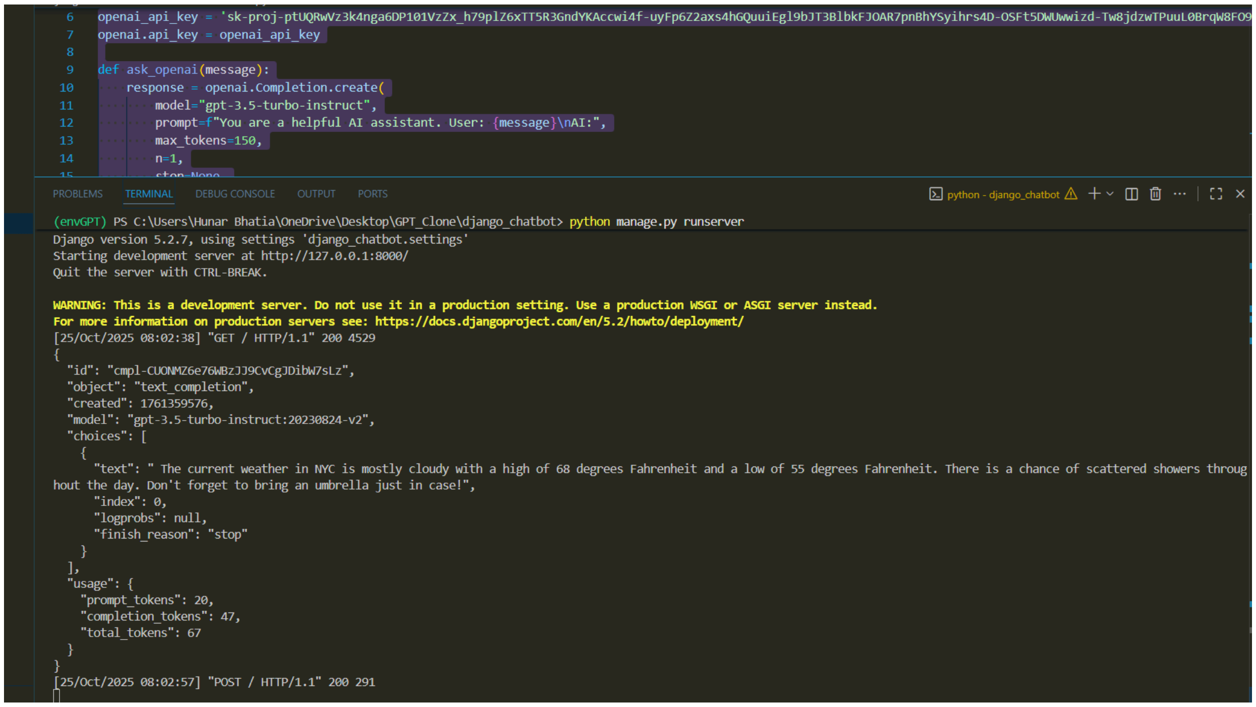
You
hi

AI Chatbot
Hi this is my response

Next we install openai library via the terminal so that we can use the openai API key later.
Now we'll create the function that should be communicating with the openai API→

```
from django.shortcuts import render
from django.http import HttpResponse
from django.http import JsonResponse
import openai

openai_api_key = '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'
openai.api_key = openai_api_key

def ask_openai(message):
    response = openai.Completion.create(
        model="gpt-3.5-turbo-instruct",
        prompt=f"You are a helpful AI assistant. User: {message}\nAI:",
        max_tokens=150,
        n=1,
        stop=None,
        temperature=0.7,
    )
    print(response)
    answer = response.choices[0].text.strip()
    return answer

# Create your views here.
def chatbot(request):
    if request.method == 'POST':
        message = request.POST.get('message')
        # Here you would typically process the message and generate a respons
e
        response = ask_openai(message)
        return JsonResponse({'message': message, 'response': response})
    return render(request, 'chatbot.html')
```

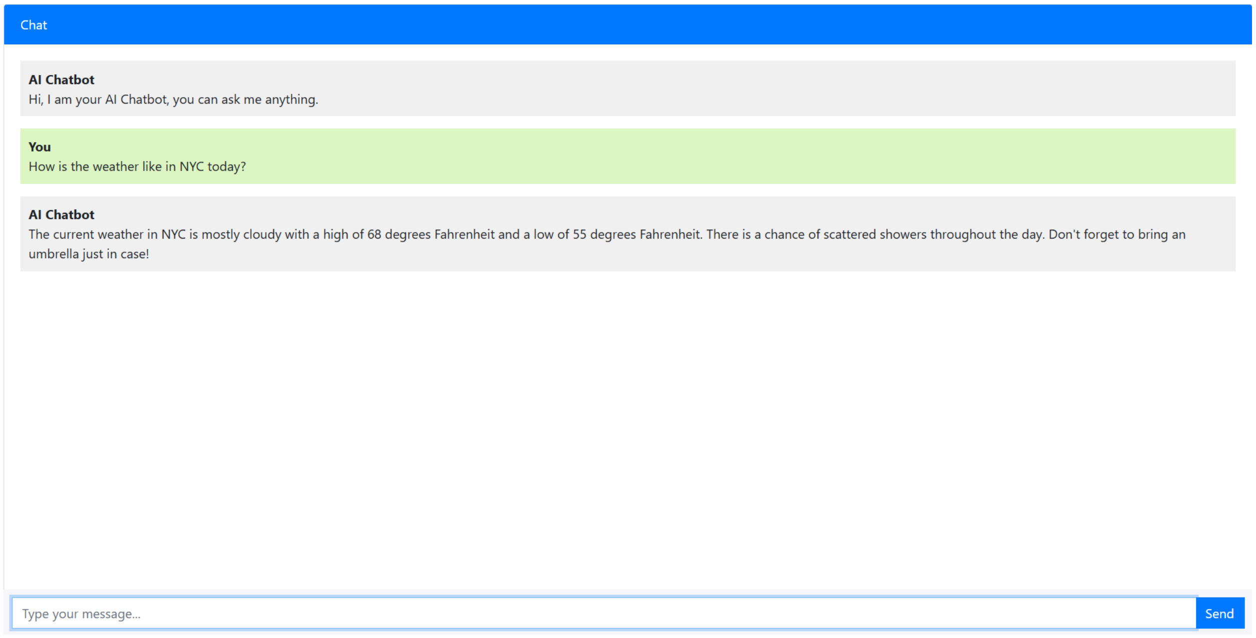
Because we wrote the print(response) line in the code, it gave us the response in the terminal as follows-:



```python
openai_api_key = 'sk-proj-ptUQRwVz3k4nga6DP101VzZx_h79plZ6xTT5R3GndYKAccwi4f-uyFp6Z2axs4hGQuuiEgl9bJT3BlbkFJOAR7pnBhYSyihrs4D-OSFt5DWUwwizd-Tw8jdzwTPuuL0BrqW8FO9
openai.api_key = openai_api_key

def ask_openai(message):
    response = openai.Completion.create(
        model="gpt-3.5-turbo-instruct",
        prompt=f"You are a helpful AI assistant. User: {message}\nAI:",
        max_tokens=150,
        n=1,
```

```
(envGPT) PS C:\Users\Hunar Bhatia\OneDrive\Desktop\GPT_Clone\django_chatbot> python manage.py runserver
Django version 5.2.7, using settings 'django_chatbot.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[25/Oct/2025 08:02:38] "GET / HTTP/1.1" 200 4529
{
  "id": "cmpl-CUONMZ6e76WBzJJ9CvCgJDibW7sLz",
  "object": "text_completion",
  "created": 1761359576,
  "model": "gpt-3.5-turbo-instruct:20230824-v2",
  "choices": [
    {
      "text": " The current weather in NYC is mostly cloudy with a high of 68 degrees Fahrenheit and a low of 55 degrees Fahrenheit. There is a chance of scattered showers throug
hout the day. Don't forget to bring an umbrella just in case!",
      "index": 0,
      "logprobs": null,
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 20,
    "completion_tokens": 47,
    "total_tokens": 67
  }
}
[25/Oct/2025 08:02:57] "POST / HTTP/1.1" 200 291
```

And this is what the browser preview looks like→



AI Chatbot
Hi, I am your AI Chatbot, you can ask me anything.

You
How is the weather like in NYC today?

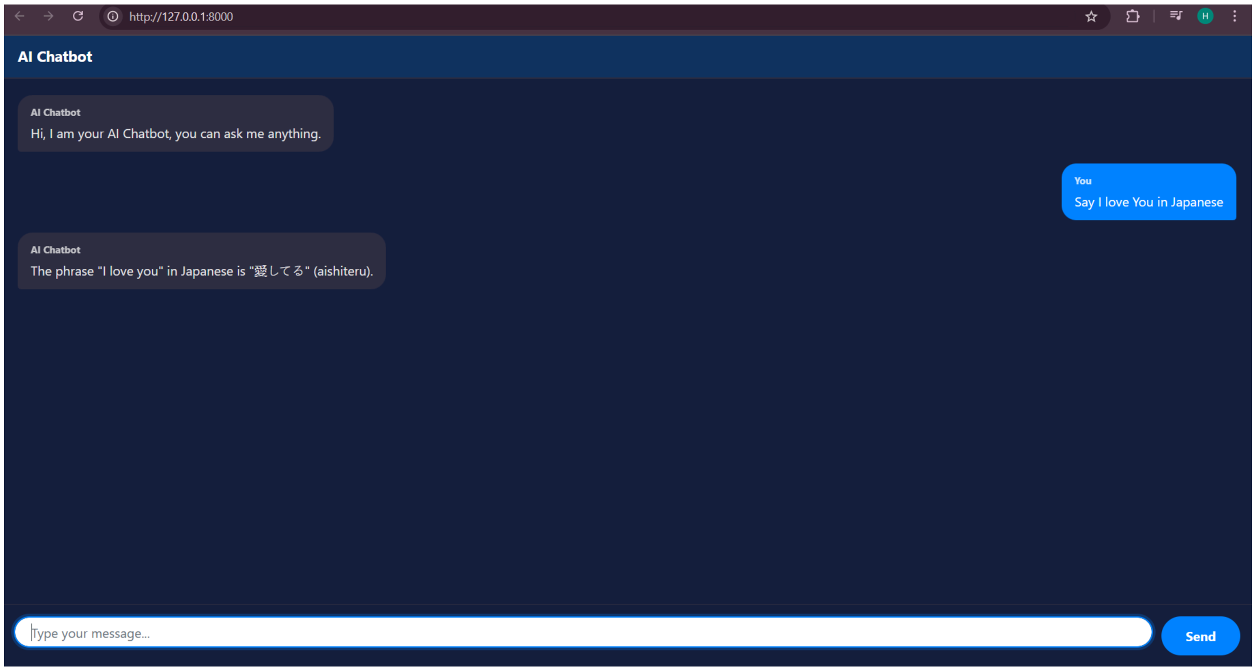
AI Chatbot
The current weather in NYC is mostly cloudy with a high of 68 degrees Fahrenheit and a low of 55 degrees Fahrenheit. There is a chance of scattered showers throughout the day. Don't forget to bring an umbrella just in case!

So here I’m thinking of making some changes of my own I’m thinking of asking the chatbot in the prompt to be brutally honest which is a problem with the actual chatgpt as it ain’t honest and always keeps buttering us.

Here’s the prompt I gave→

```
prompt=f"You are a brutally honest AI assistant. You don't sugarcoat things or give fake encouragement. You tell the truth directly, even if it's harsh or uncomfortable. Be straightforward and honest in your responses, no matter what. User: {message}\nAI:",
```

Changed the UI a bit-:



AI Chatbot
Hi, I am your AI Chatbot, you can ask me anything.

You
Say I love You in Japanese

AI Chatbot
The phrase "I love you" in Japanese is "愛してる" (aishiteru).

Next, we go ahead and add the code for the login and register HTML templates from the github repo into the codebase in two new files with the name login.html and register.html.

Then we’ll add the paths for these in the urls.py file.

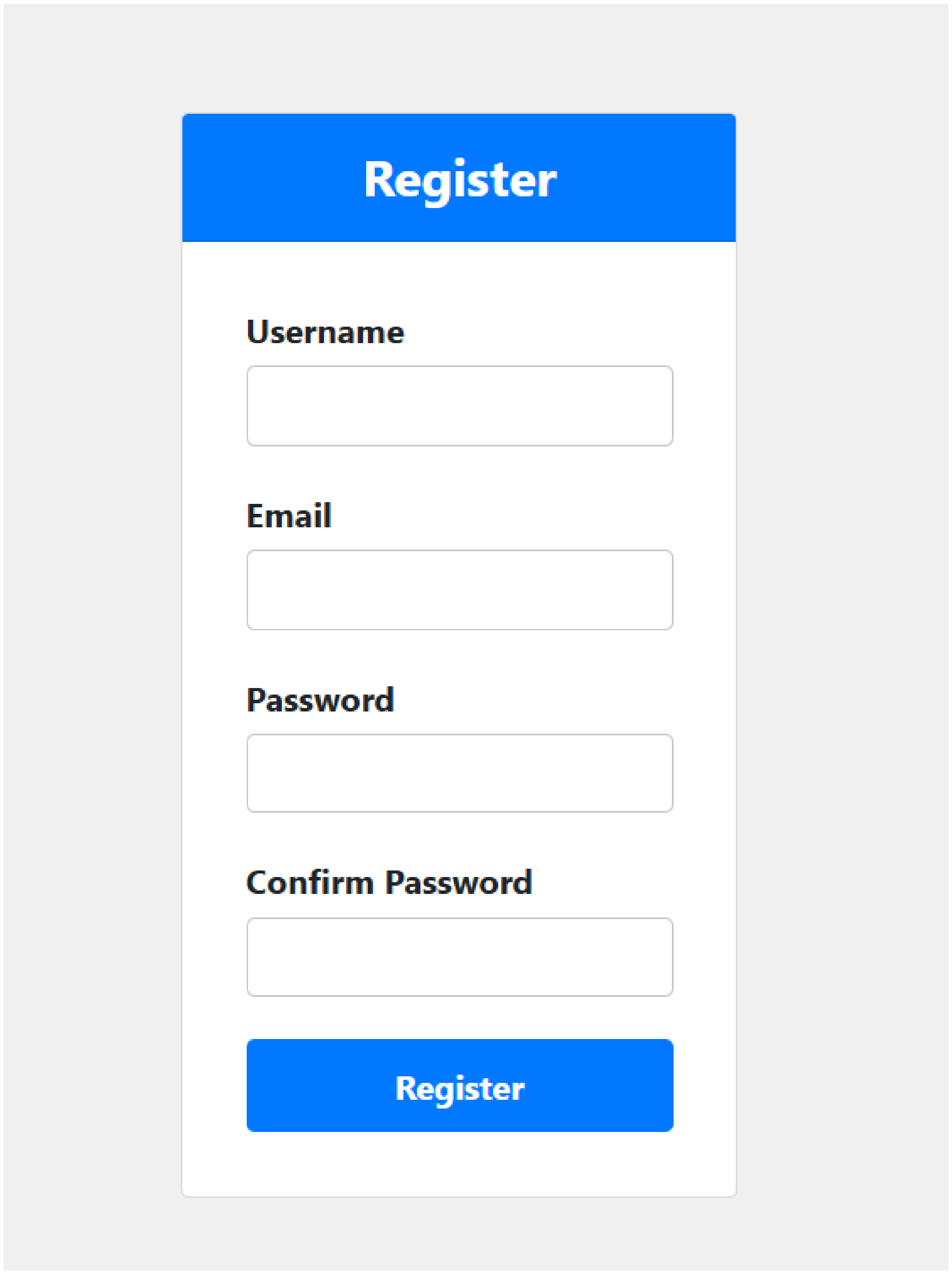
```
urlpatterns = [
    path('', views.chatbot, name='chatbot'),
    path('login/', views.login, name='login'),
    path('register/', views.register, name='register'),
    path('logout/', views.logout, name='logout'),
]
```

Now, we’ll go ahead and create views for login, logout and register.

```
def login(request):
    return render(request, 'login.html')

def register(request):
    return render(request, 'register.html')

def logout(request):
    auth.logout(request)
```

Add the registration and login logic so that your file looks like this-:

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from django.http import JsonResponse
import openai
from django.contrib import auth
from django.contrib.auth.models import User



openai_api_key = 'sk-proj-ptUQRwVz3k4nga6DP101VzZx_h79plZ6xTT5R3Gnd
YKAccwi4f-uyFp6Z2axs4hGQuuiEgl9bJT3BlbkFJOAR7pnBhYSyihrs4D-OSFt5
DWUwwizd-Tw8jdzwTPuuL0BrqW8FO9uSoBKdnrRvU3KnIvhwXMA'
openai.api_key = openai_api_key

def ask_openai(message):
    response = openai.Completion.create(
        model="gpt-3.5-turbo-instruct",
        prompt=f"You are a brutally honest AI assistant. You don't sugarcoat thin
gs or give fake encouragement. You tell the truth directly, even if it's harsh or
uncomfortable. Be straightforward and honest in your responses, no matter w
hat. But you have to always provide me the answer and be helpful. User: {mes
sage}\nAI:",
        max_tokens=150,
        n=1,
        stop=None,
        temperature=0.7,
    )
    answer = response.choices[0].text.strip()
    return answer

# Create your views here.
def chatbot(request):
    if request.method == 'POST':
        message = request.POST.get('message')
        # Here you would typically process the message and generate a respons
e
        response = ask_openai(message)
        return JsonResponse({'message': message, 'response': response})
    return render(request, 'chatbot.html')

def login(request):
    return render(request, 'login.html')

def register(request):
    if request.method == 'POST':
        # Handle user registration logic here
        username = request.POST['username']
        email = request.POST['email']
        password1 = request.POST['password1']
        password2 = request.POST['password2']

        if password1 == password2:
            try:
                user = User.objects.create_user(username=username, email=email,
password=password1)
                user.save()
                auth.login(request, user)
                return redirect('chatbot')
            except:
                error_message = "Username already exists."
                return render(request, 'register.html', {'error_message': error_messa
ge})
        else:
            error_message = "Passwords do not match."
            return render(request, 'register.html', {'error_message': error_messag
e})
    return render(request, 'register.html')

def logout(request):
    auth.logout(request)
```



This is what our registration page looks like💁🏻‍♂️.

Did a little bit of editing in the HTML part and now we go on and create a model-:

```
from django.db import models
from django.contrib.auth.models import User

# Create your models here.
class Chat(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    message = models.TextField()
    response = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f'{self.user.username}: {self.message}'
```

Now, we create a superuser for admin.In the admin.py file we'll now import the model that we made earlier -:

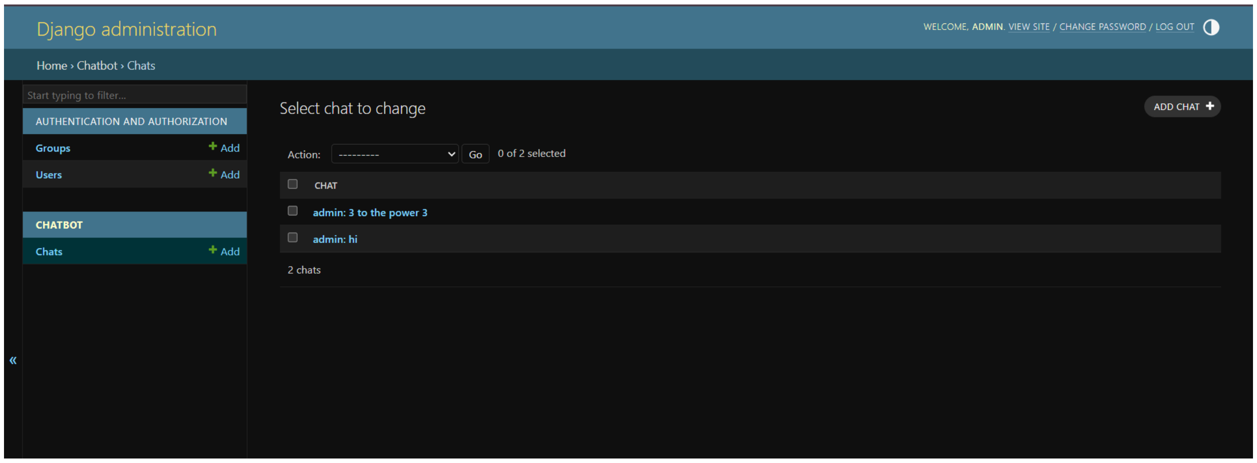
```
from django.contrib import admin
from .models import Chat
# Register your models here.
admin.site.register(Chat)
```

Now, what we need to do is we need to save all the message history in our Chat database.So, let's do that right now.Make these changes in the views.py file and it would work→

```
def chatbot(request):
    if request.method == 'POST':
        message = request.POST.get('message')
        # Here you would typically process the message and generate a respons
e
        response = ask_openai(message)

        # Save chat to database
        chat=Chat(user=request.user, message=message, response=response,
created_at=timezone.now())
```

```
        chat.save()

        return JsonResponse({'message': message, 'response': response})
    return render(request, 'chatbot.html')
```

Going to the admin page we can verify that the chat is being saved as follows-:


Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Chatbot › Chats

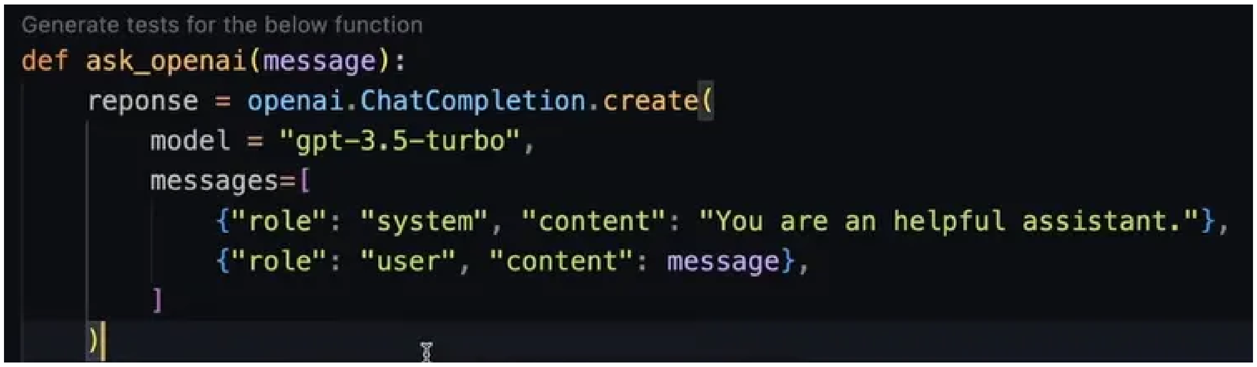
Select chat to change

Action: --------- Go 0 of 2 selected

CHAT
admin: 3 to the power 3
admin: hi

2 chats

Finally, we add openai chat completion to our project by making the following changes in the views file-:


Generate tests for the below function

```
def ask_openai(message):
    reponse = openai.ChatCompletion.create(
        model = "gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are an helpful assistant."},
            {"role": "user", "content": message},
        ]
    )
```

So chat completion is not exactly working as it has been updated in the OpenAI codebase and we are using the older version of the package and changing it would lead to collapsing of our entire codebase so…

Thanks For Reading!!!