Source Code of App: chat

admin.py

```
from django.contrib import admin

# Register your models here.
```

apps.py

```
# chat/apps.py
from django.apps import AppConfig

class ChatConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'chat'

    def ready(self):
        from myapp.models import Car
```

consumers.py

```
# consumers.py
import json
from channels.generic.websocket import AsyncWebsocketConsumer
from channels.db import database_sync_to_async
from django.contrib.auth import get_user_model
from .models import ChatRoom, Message

class ChatConsumer(AsyncWebsocketConsumer):
    async def connect(self):
        self.room_id = self.scope['url_route']['kwargs']['room_id']
        self.room_group_name = f'chat_{self.room_id}'
        self.user = self.scope['user']

        # Verify the user is a participant
        if not await self.is_room_participant():
            await self.close()
            return

        await self.channel_layer.group_add(
            self.room_group_name,
            self.channel_name
        )
        await self.accept()

    @database_sync_to_async
    def is_room_participant(self):
        try:
            room = ChatRoom.objects.get(id=self.room_id)
            return self.user in room.participants.all()
        except ChatRoom.DoesNotExist:
            return False

    @database_sync_to_async
    def save_message(self, message):
        room = ChatRoom.objects.get(id=self.room_id)
        return Message.objects.create(
            chat_room=room,
            sender=self.user,
            content=message
        )

    async def disconnect(self, close_code):
        await self.channel_layer.group_discard(
            self.room_group_name,
            self.channel_name
        )

    async def receive(self, text_data):
        data = json.loads(text_data)
        message = data['message']

        # Save message to database
        message_obj = await self.save_message(message)

        # Send message to room group
        await self.channel_layer.group_send(
            self.room_group_name,
            {
                'type': 'chat_message',
```

```
                'message': message,
                'sender_id': self.user.id,
                'sender_name': self.user.username,
                'timestamp': message_obj.timestamp.strftime('%H:%M')
            }
        )

    async def chat_message(self, event):
        # Send message to WebSocket
        await self.send(text_data=json.dumps({
            'message': event['message'],
            'sender_id': event['sender_id'],
            'sender_name': event['sender_name'],
            'timestamp': event['timestamp']
        }))
```

forms.py

```
from django import forms
from .models import Message

class MessageForm(forms.ModelForm):
    class Meta:
        model = Message
        fields = ['content']
```

models.py

```
# chat/models.py
from django.db import models
from django.conf import settings
from myapp.models import Car   # ใช้การ import โมเดลจากแอป myapp โดยตรง

class ChatRoom(models.Model):
    car = models.ForeignKey(Car, on_delete=models.CASCADE)
    participants = models.ManyToManyField(settings.AUTH_USER_MODEL, related_name='chat_rooms')
    created_at = models.DateTimeField(auto_now_add=True)
    is_active = models.BooleanField(default=True)  # Add this to mark if chat is still active
    last_message_at = models.DateTimeField(auto_now=True)  # Add this to track latest activity

    class Meta:
        ordering = ['-last_message_at']  # Sort by most recent activity

    def __str__(self):
        return f"Chat Room for {self.car.name} with {', '.join([user.username for user in self.participants.all()])}"

    def get_other_participant(self, user):
        """Helper method to get the other participant"""
        return self.participants.exclude(id=user.id).first()

class Message(models.Model):
    chat_room = models.ForeignKey(ChatRoom, related_name='messages', on_delete=models.CASCADE)
    sender = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    content = models.TextField()
    timestamp = models.DateTimeField(auto_now_add=True)
    is_read = models.BooleanField(default=False)  # Add this to track read status

    class Meta:
        ordering = ['timestamp']  # Sort messages chronologically

    def __str__(self):
        return f"{self.sender.username}: {self.content}"

    def mark_as_read(self):
        """Helper method to mark message as read"""
        self.is_read = True
        self.save()
```

routing.py

```
from django.urls import re_path
from . import consumers

websocket_urlpatterns = [
    re_path(r'ws/chat/(?P\w+)/$', consumers.ChatConsumer.as_asgi()),
]
```

tests.py

```
from django.test import TestCase

# Create your tests here.
```

urls.py

```
from django.urls import path, include
from chat import views as chat_views
from django.contrib.auth.views import LoginView, LogoutView
from . import views
from .views import my_chats

appname = "chat"

urlpatterns = [
    path("chat/", chat_views.chatPage, name="chat-page"),

    path('create-chat//', views.create_chat_room, name='create_chat_room'),
    path('chat//', views.chat_room, name='chat_room'),
    path('my-chats/', my_chats, name='my_chats'),

]
```

views.py

```
from django.shortcuts import render, redirect
from django.shortcuts import render, get_object_or_404
from .models import ChatRoom, Message
from .forms import MessageForm
from .models import ChatRoom, Car
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseForbidden
from django.core.exceptions import PermissionDenied


def chatPage(request):
    # ดึงห้องแชททั้งหมด
    chat_rooms = ChatRoom.objects.all()
    return render(request, "chat/chatPage.html", {"chat_rooms": chat_rooms})


@login_required
def create_chat_room(request, car_id):
    car = get_object_or_404(Car, id=car_id)

    # Check if chat room already exists
    chat_room = ChatRoom.objects.filter(
        car=car,
        participants=request.user
    ).first()

    if not chat_room:
        # Create new chat room and add both user and car owner
        chat_room = ChatRoom.objects.create(car=car)
        chat_room.participants.add(request.user, car.owner)

    return redirect('chat_room', room_id=chat_room.id)

# views.py
@login_required
def chat_room(request, room_id):
    room = get_object_or_404(ChatRoom, id=room_id)

    # Check if user is authorized (either owner or participant)
    if request.user != room.car.owner and request.user not in room.participants.all():
        return HttpResponseForbidden("Not authorized")

    # Get the other participant
    other_participant = room.get_other_participant(request.user)

    context = {
        'room': room,
        'messages': room.messages.all().order_by('timestamp'),
        'other_participant': other_participant,
        'is_owner': request.user == room.car.owner
    }

    return render(request, 'chat_room.html', context)


@login_required
def my_chats(request):
    # ดึงห้องแชททั้งหมดที่ผู้ใช้เป็น participant (ทั้งเจ้าของรถและผู้สนใจ)
    chats = ChatRoom.objects.filter(participants=request.user).distinct()
```

```
    return render(request, 'owner_chats.html', {
        'chats': chats
    })
```

__init__.py

0001_initial.py

```
# Generated by Django 5.1.1 on 2025-02-10 08:31

import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
        ("myapp", "0020_notification_borrower"),
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
    ]

    operations = [
        migrations.CreateModel(
            name="ChatRoom",
            fields=[
                (
                    "id",
                    models.BigAutoField(
                        auto_created=True,
                        primary_key=True,
                        serialize=False,
                        verbose_name="ID",
                    ),
                ),
                ("created_at", models.DateTimeField(auto_now_add=True)),
                (
                    "car",
                    models.ForeignKey(
                        on_delete=django.db.models.deletion.CASCADE, to="myapp.car"
                    ),
                ),
                (
                    "participants",
                    models.ManyToManyField(
                        related_name="chat_rooms", to=settings.AUTH_USER_MODEL
                    ),
                ),
            ],
        ),
        migrations.CreateModel(
            name="Message",
            fields=[
                (
                    "id",
                    models.BigAutoField(
                        auto_created=True,
                        primary_key=True,
                        serialize=False,
                        verbose_name="ID",
                    ),
                ),
                ("content", models.TextField()),
                ("timestamp", models.DateTimeField(auto_now_add=True)),
                (
                    "chat_room",
                    models.ForeignKey(
                        on_delete=django.db.models.deletion.CASCADE,
                        related_name="messages",
                        to="chat.chatroom",
                    ),
                ),
                (
                    "sender",
                    models.ForeignKey(
                        on_delete=django.db.models.deletion.CASCADE,
                        to=settings.AUTH_USER_MODEL,
                    ),
                ),
            ],
        ),
```

```
    ]
```

0002_alter_chatroom_options_alter_message_options_and_more.py

```
# Generated by Django 5.1.1 on 2025-02-10 18:53

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ("chat", "0001_initial"),
    ]

    operations = [
        migrations.AlterModelOptions(
            name="chatroom",
            options={"ordering": ["-last_message_at"]},
        ),
        migrations.AlterModelOptions(
            name="message",
            options={"ordering": ["timestamp"]},
        ),
        migrations.AddField(
            model_name="chatroom",
            name="is_active",
            field=models.BooleanField(default=True),
        ),
        migrations.AddField(
            model_name="chatroom",
            name="last_message_at",
            field=models.DateTimeField(auto_now=True),
        ),
        migrations.AddField(
            model_name="message",
            name="is_read",
            field=models.BooleanField(default=False),
        ),
    ]
```

__init__.py

chatPage.html

เลือกห้องแชท

```
                {% for room in chat_rooms %}
```

- {{ room.car.name }}

```
                {% empty %}
```

- ไม่มีห้องแชท

```
                {% endfor %}
```

chat_room.html

```
{% extends 'base.html' %}

{% block content %}
```

{{ room.car.name }}

```
            {% if request.user == room.car.owner %}
                แชทกับ: {{ other_participant.username }} (ผู้สนใจ)
            {% else %}
                แชทกับ: {{ room.car.owner.username }} (เจ้าของรถ)
            {% endif %}
```

```
        {% for message in messages %}

                        {{ message.sender.username }}

    {{ message.content }}

                    {{ message.timestamp|date:"H:i" }}

        {% endfor %}
```

พิมพ์ข้อความ...

ส่ง

```
{% endblock %}
```

owner_chats.html

```
{% extends 'base.html' %}

{% block content %}
```

💬 การสนทนาของคุณ

```
    {% if chats %}

        {% for chat in chats %}
```

{{ chat.car.name }}

```
                {% for participant in chat.participants.all %}
                    {% if participant != request.user %}
                        แชทกับ: {{ participant.username }}
                    {% endif %}
                {% endfor %}

            {% with last_message=chat.messages.last %}
                {% if last_message %}
```

💬 ข้อความล่าสุด: {{ last_message.content|truncatechars:50 }}

```
                        {{ last_message.timestamp|timesince }} ที่แล้ว
                {% endif %}
            {% endwith %}
```

[👁 ดูการสนทนา]

```
        {% endfor %}

    {% else %}
```

● ยังไม่มีการสนทนา

```
    {% endif %}

{% endblock %}
```

room.html

Chat Room for {{ room.car.name }}

{% for message in messages %}

{{ message.sender.username }}:

{{ message.content }}

{{ message.timestamp }}

{% endfor %}

{% csrf_token %}
{{ form.as_p }}
ส่งข้อความ

กลับไปหน้าห้องแชท