models.py

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

class Bus(models.Model):
    name = models.CharField(max_length=100)
    driver = models.OneToOneField(User, on_delete=models.CASCADE)
    is_tracking = models.BooleanField(default=False)

    def __str__(self):
        return self.name

class BusLocation(models.Model):
    bus = models.ForeignKey(Bus, on_delete=models.CASCADE)
    latitude = models.FloatField()
    longitude = models.FloatField()
    timestamp = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"{self.bus.name} at ({self.latitude}, {self.longitude})"
```

views.py

```python
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response
from .models import Bus, BusLocation

@api_view(['POST'])
@permission_classes([IsAuthenticated])
def toggle_tracking(request):
    try:
        bus = Bus.objects.get(driver=request.user)
        bus.is_tracking = not bus.is_tracking
        bus.save()
        return Response({'tracking': bus.is_tracking})
    except Bus.DoesNotExist:
        return Response({'error': 'Bus not found for this driver'}, status=404)

@api_view(['POST'])
@permission_classes([IsAuthenticated])
def update_location(request):
    try:
        bus = Bus.objects.get(driver=request.user)
        if not bus.is_tracking:
            return Response({'error': 'Tracking is OFF'}, status=400)

        latitude = request.data.get('latitude')
        longitude = request.data.get('longitude')

        if latitude is None or longitude is None:
            return Response({'error': 'Latitude and Longitude required'}, status=400)

        BusLocation.objects.create(bus=bus, latitude=latitude, longitude=longitude)
        return Response({'status': 'Location updated'})
    except Bus.DoesNotExist:
        return Response({'error': 'Bus not found for this driver'}, status=404)
```

urls.py

```python
from django.urls import path
from .views import toggle_tracking, update_location
```

```
urlpatterns = [
    path('api/toggle-tracking/', toggle_tracking),
    path('api/update-location/', update_location),
]
```

consumers.py

```
import json
from channels.generic.websocket import AsyncWebsocketConsumer

class BusTrackingConsumer(AsyncWebsocketConsumer):
    async def connect(self):
        self.bus_id = self.scope['url_route']['kwargs']['bus_id']
        self.group_name = f'bus_{self.bus_id}'
        await self.channel_layer.group_add(self.group_name, self.channel_name)
        await self.accept()

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

    async def receive(self, text_data):
        data = json.loads(text_data)
        latitude = data['latitude']
        longitude = data['longitude']
        await self.channel_layer.group_send(
            self.group_name,
            {
                'type': 'send_location',
                'latitude': latitude,
                'longitude': longitude,
            }
        )

    async def send_location(self, event):
        await self.send(text_data=json.dumps({
            'latitude': event['latitude'],
            'longitude': event['longitude'],
        }))
```

routing.py

```
from channels.routing import ProtocolTypeRouter, URLRouter
from channels.auth import AuthMiddlewareStack
from django.urls import re_path
from your_app.consumers import BusTrackingConsumer

application = ProtocolTypeRouter({
    "websocket": AuthMiddlewareStack(
        URLRouter([
            re_path(r'ws/track/(?P<bus_id>\d+)/$', BusTrackingConsumer.as_asgi()),
        ])
    ),
})
```

React - BusTracking.js

```
import React, { useEffect, useState } from 'react';
import { MapContainer, TileLayer, Marker, Popup } from 'react-leaflet';
import 'leaflet/dist/leaflet.css';

const BusTracking = ({ busId }) => {
  const [location, setLocation] = useState(null);

  useEffect(() => {
    const socket = new WebSocket(`ws://localhost:8000/ws/track/${busId}/`);

    socket.onmessage = (event) => {
      const data = JSON.parse(event.data);
      setLocation({ lat: data.latitude, lng: data.longitude });
    };

    return () => socket.close();
  }, [busId]);

  return (
    <div>
      <h2 className="text-xl font-bold mb-2">Live Bus Location</h2>
      {location ? (
        <MapContainer center={location} zoom={16} style={{ height: '500px', width: '100%' }}>
          <TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />
          <Marker position={location}>
            <Popup>Bus is here!</Popup>
          </Marker>
        </MapContainer>
      ) : (
        <p>Waiting for location update...</p>
      )}
    </div>
  );
};

export default BusTracking;
```