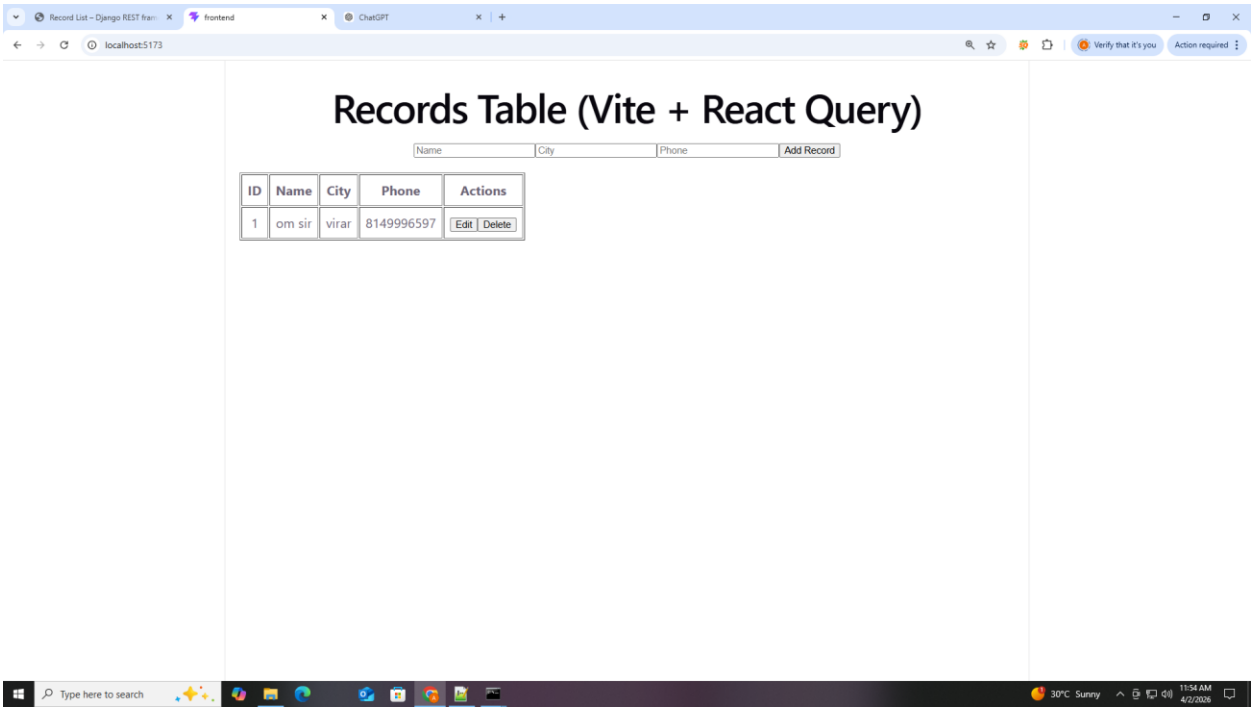
Let’s make a full Django + React Query example using a table with fields: id, name, city, phone. I’ll show backend + frontend with CRUD support.



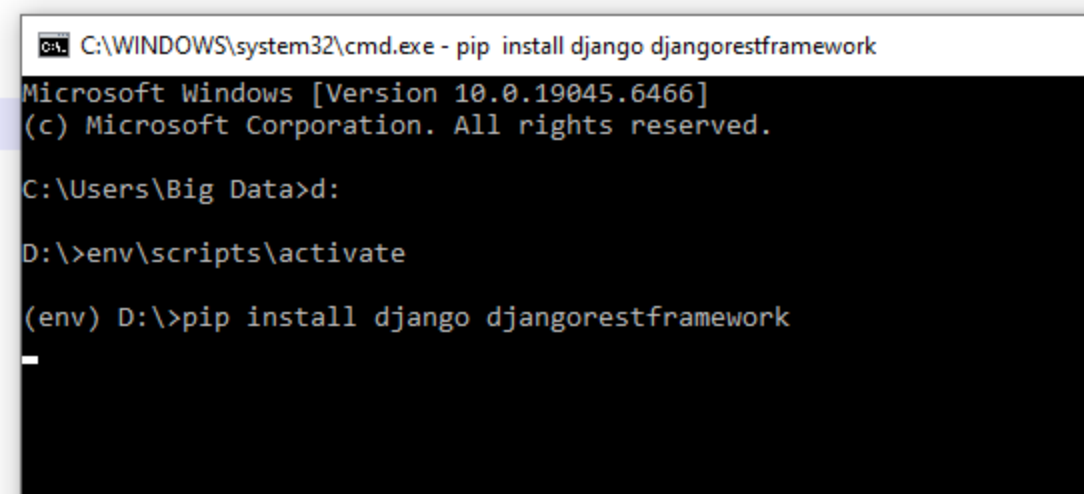

Let’s make a full Django + React Query example using a table with fields: id, name, city, phone. I’ll show backend + frontend with CRUD support.

## 1 Django Backend

Install packages (if not done already):

pip install django djangorestframework

pip install django-cors-headers




Create project & app:

django-admin startproject restapi

cd restapi

python manage.py startapp api

```
(env) D:\recap>django-admin startproject restapi

(env) D:\recap>cd restapi

(env) D:\recap\restapi>python manage.py startapp api

(env) D:\recap\restapi>
```

Add apps in restapi/settings.py:

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'api',
```

```
    'corsheaders',
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',

            ]



# settings.py
CORS_ALLOWED_ORIGINS = [
    "http://localhost:5173",
]
```

Create model (api/models.py):-

```
from django.db import models

class Record(models.Model):
    name = models.CharField(max_length=100)
    city = models.CharField(max_length=100)
    phone = models.CharField(max_length=20)

    def __str__(self):
        return self.name
```

Create serializer (api/serializers.py):

```
from rest_framework import serializers
from .models import Record

class RecordSerializer(serializers.ModelSerializer):
    class Meta:
        model = Record
        fields = ['id', 'name', 'city', 'phone']
```

Create viewset (api/views.py):

```
from rest_framework import viewsets
from .models import Record
from .serializers import RecordSerializer

class RecordViewSet(viewsets.ModelViewSet):
    queryset = Record.objects.all()
    serializer_class = RecordSerializer
```

Create router (api/urls.py):

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import RecordViewSet

router = DefaultRouter()
router.register(r'records', RecordViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

Include API URLs in restapi/urls.py:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('api.urls')),
]
```

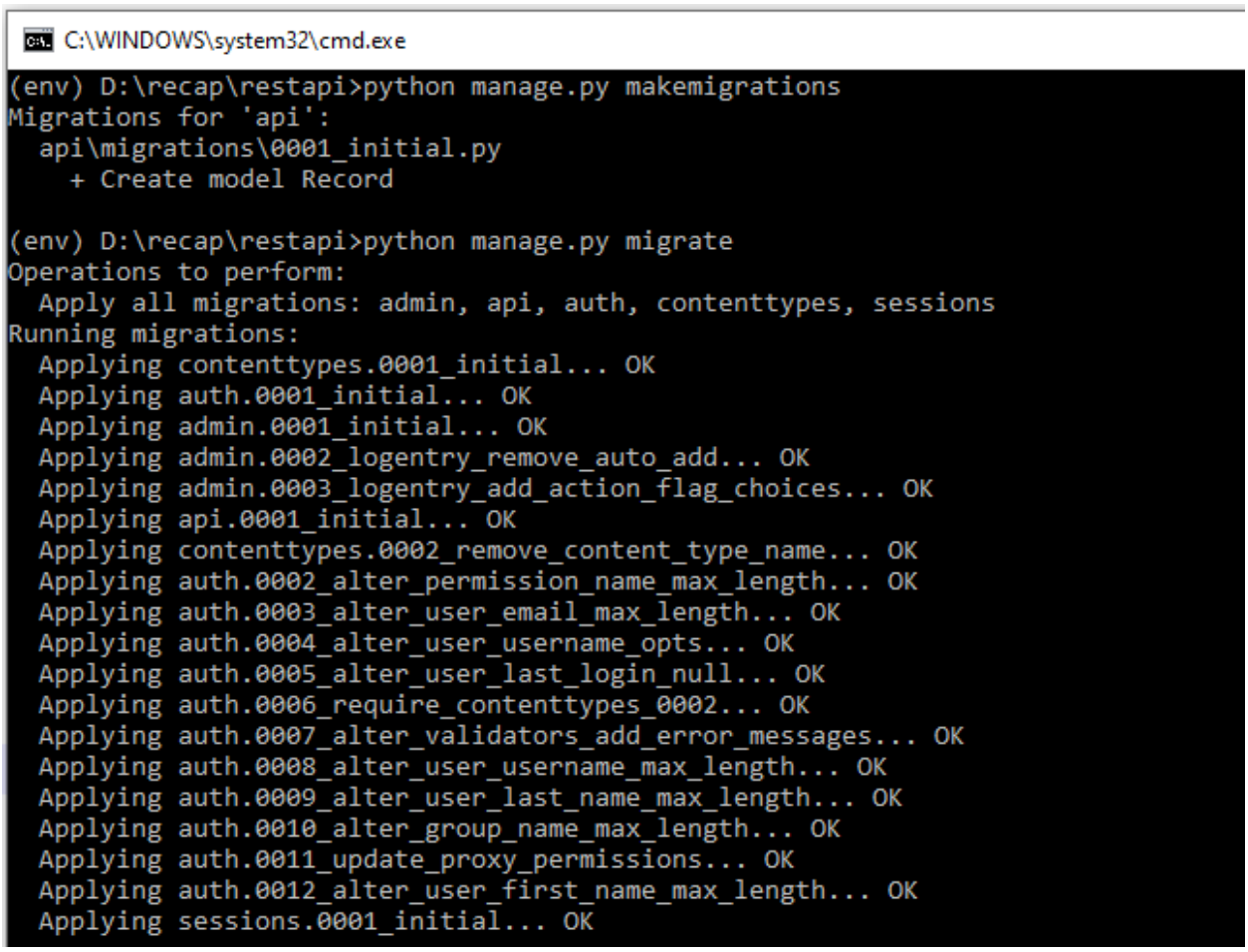
Migrate and create sample data:

python manage.py makemigrations

```
(env) D:\recap\restapi>python manage.py makemigrations
Migrations for 'api':
  api\migrations\0001_initial.py
    + Create model Record

(env) D:\recap\restapi>
```

python manage.py migrate



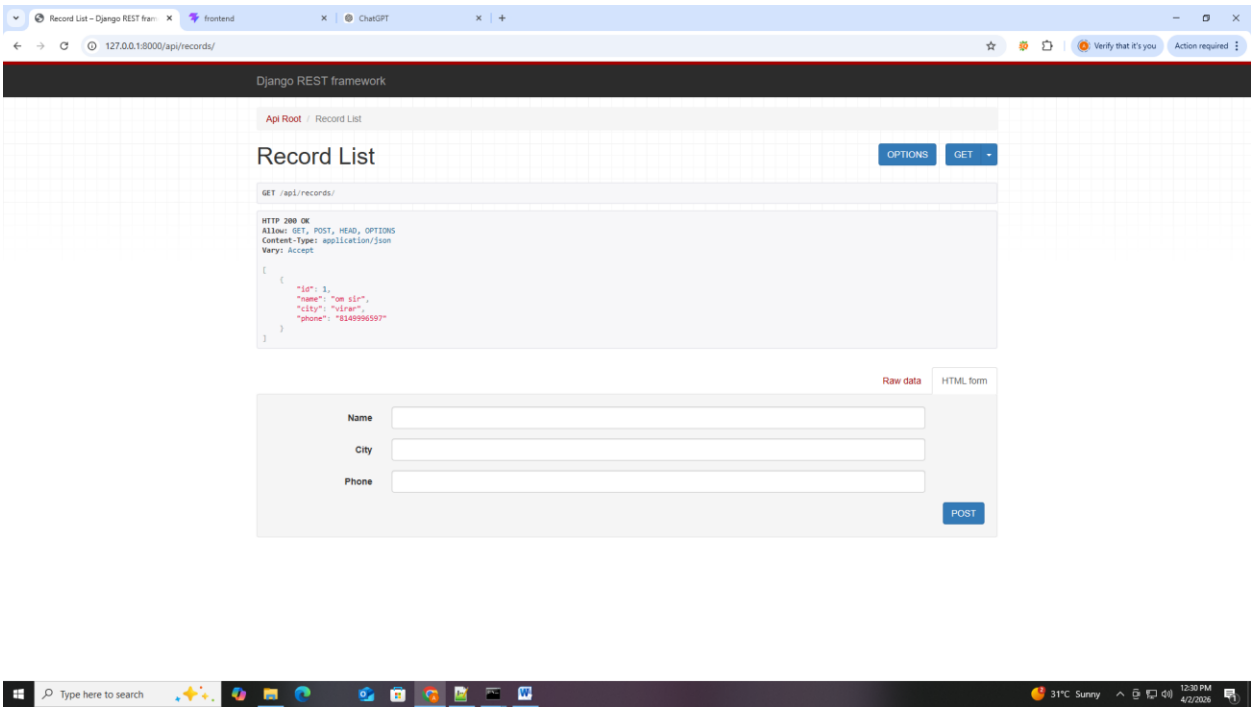

python manage.py createsuperuser

python manage.py runserver

Your API endpoint:

GET http://127.0.0.1:8000/api/records/

POST http://127.0.0.1:8000/api/records/


Record List – Django REST fram
frontend
ChatGPT
127.0.0.1:8000/api/records/
Verify that it's you
Action required
Django REST framework
Api Root / Record List
Record List
OPTIONS
GET
GET /api/records/
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 1,
"name": "om sir",
"city": "virar",
"phone": "8149996597"
}
]
Raw data
HTML form
Name
City
Phone
POST


Type here to search
31°C Sunny
12:30 PM
4/2/2026

## 2 React Frontend with React Query + Axios

## 1 Django Backend

This stays exactly the same as before. Your API endpoints:

GET /api/records/ → fetch all records

POST /api/records/ → create record

PUT /api/records/:id/ → update record

DELETE /api/records/:id/ → delete record

Make sure Django is running at: http://127.0.0.1:8000/

## 2 React Frontend with Vite

Create Vite React app:

npm create vite@latest frontend -- --template react

```
D:\recap>npm create vite@latest frontend -- --template react

> npx
> create-vite frontend --template react

|
o  Install with npm and start now?
|  Yes
|
o  Scaffolding project in D:\recap\frontend...
|
o  Installing dependencies with npm...

added 151 packages, and audited 152 packages in 30s
```

cd frontend

npm install

npm install axios @tanstack/react-query

```
D:\recap>cd frontend

D:\recap\frontend>npm install axios @tanstack/react-query

added 25 packages, and audited 177 packages in 5s

44 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

D:\recap\frontend>
```

Axios instance (src/api/axios.js)

import axios from "axios";

```
const axiosInstance = axios.create({
  baseURL: "http://127.0.0.1:8000/api/",
});

export default axiosInstance;
```

React Query Provider (src/main.jsx)

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App.jsx";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";
import "./index.css";
```

```
const queryClient = new QueryClient();


ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <QueryClientProvider client={queryClient}>
      <App />
    </QueryClientProvider>
  </React.StrictMode>
);
```

CRUD App (src/App.jsx)

App.jsx file code:-

```
import React, { useState } from "react";
import { useQuery, useMutation, useQueryClient } from "@tanstack/react-query";
import axiosInstance from "./api/axios";


const fetchRecords = async () => {
  const { data } = await axiosInstance.get("records/");
  return data;
};


const addRecord = async (record) => {
  const { data } = await axiosInstance.post("records/", record);
  return data;
};


const updateRecord = async ({ id, record }) => {
  const { data } = await axiosInstance.put(`records/${id}/`, record);
  return data;
};


const deleteRecord = async (id) => {
  const { data } = await axiosInstance.delete(`records/${id}/`);
  return data;
```

```
};

function App() {
  const queryClient = useQueryClient();
  const [form, setForm] = useState({ name: "", city: "", phone: "" });
  const [editId, setEditId] = useState(null);

  // ✅ FIXED (v5 format)
  const { data = [], isLoading, error } = useQuery({
    queryKey: ["records"],
    queryFn: fetchRecords,
  });

  // ✅ FIXED (v5 format)
  const addMutation = useMutation({
    mutationFn: addRecord,
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ["records"] });
    },
  });

  // ✅ FIXED (v5 format)
  const updateMutation = useMutation({
    mutationFn: updateRecord,
    onSuccess: () => {
```

```
    queryClient.invalidateQueries({ queryKey: ["records"] });
    setEditId(null);
    setForm({ name: "", city: "", phone: "" });
  },
});

// ✅ FIXED (v5 format)
const deleteMutation = useMutation({
  mutationFn: deleteRecord,
  onSuccess: () => {
    queryClient.invalidateQueries({ queryKey: ["records"] });
  },
});

const handleSubmit = (e) => {
  e.preventDefault();
  if (editId) {
    updateMutation.mutate({ id: editId, record: form });
  } else {
    addMutation.mutate(form);
  }
  setForm({ name: "", city: "", phone: "" });
};

const handleEdit = (record) => {
```

```
  setEditId(record.id);
  setForm({ name: record.name, city: record.city, phone: record.phone });
};

if (isLoading) return <p>Loading...</p>;
if (error) return <p>Error fetching records!</p>;

return (
  <div style={{ padding: "20px" }}>
    <h1>Records Table (Vite + React Query)</h1>

    <form onSubmit={handleSubmit} style={{ marginBottom: "20px" }}>
      <input
        placeholder="Name"
        value={form.name}
        onChange={(e) => setForm({ ...form, name: e.target.value })}
        required
      />
      <input
        placeholder="City"
        value={form.city}
        onChange={(e) => setForm({ ...form, city: e.target.value })}
        required
      />
      <input
```

```
    placeholder="Phone"
    value={form.phone}
    onChange={(e) => setForm({ ...form, phone: e.target.value })}
    required
   />
   <button type="submit">{editId ? "Update" : "Add"} Record</button>
  </form>

  <table border="1" cellPadding="8">
   <thead>
    <tr>
     <th>ID</th>
     <th>Name</th>
     <th>City</th>
     <th>Phone</th>
     <th>Actions</th>
    </tr>
   </thead>
   <tbody>
    {data.map((record) => (
     <tr key={record.id}>
      <td>{record.id}</td>
      <td>{record.name}</td>
      <td>{record.city}</td>
      <td>{record.phone}</td>
```

```
            <td>
              <button onClick={() => handleEdit(record)}>Edit</button>
              <button onClick={() => deleteMutation.mutate(record.id)}>
                Delete
              </button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
 );
}

export default App;
```

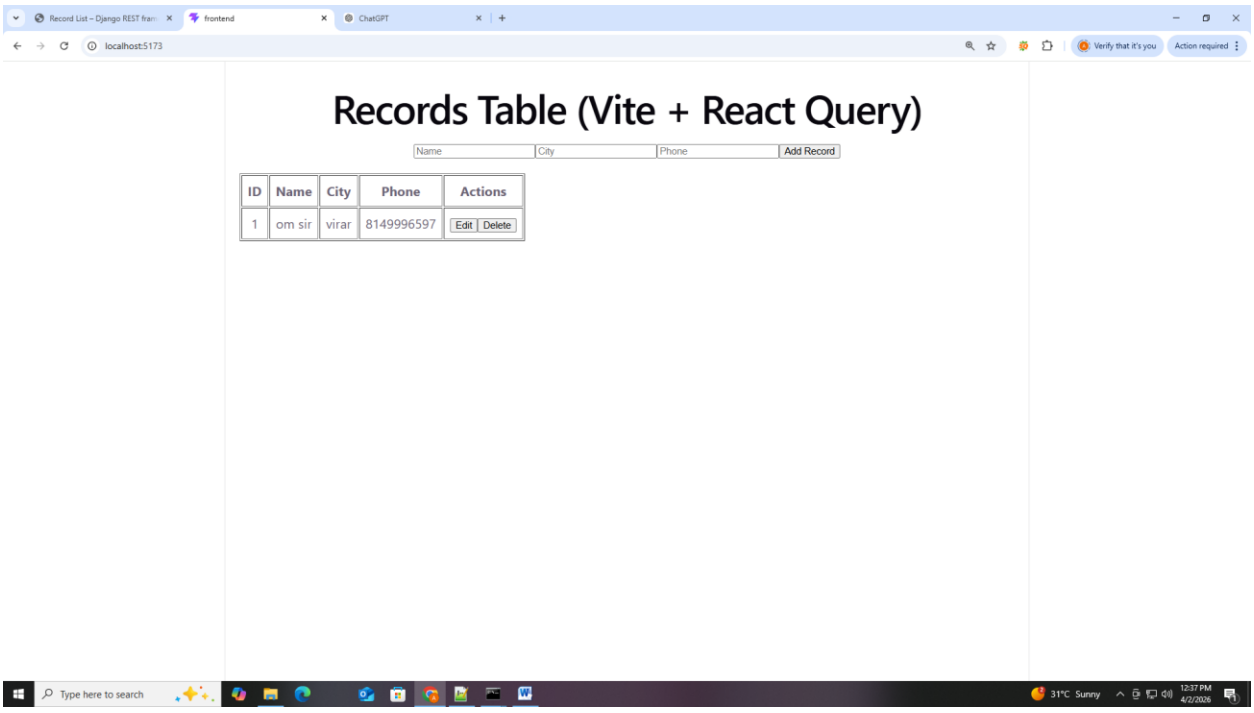
finally  run it:-

npm run dev

and visit:-

http://localhost:5173/

you will see




## ✅ Features implemented

Fetch all records (GET)

Add record (POST)

Edit record (PUT)

Delete record (DELETE)

React Query handles caching, refetch, loading, and error states

Vite + React provides fast HMR (hot reload)