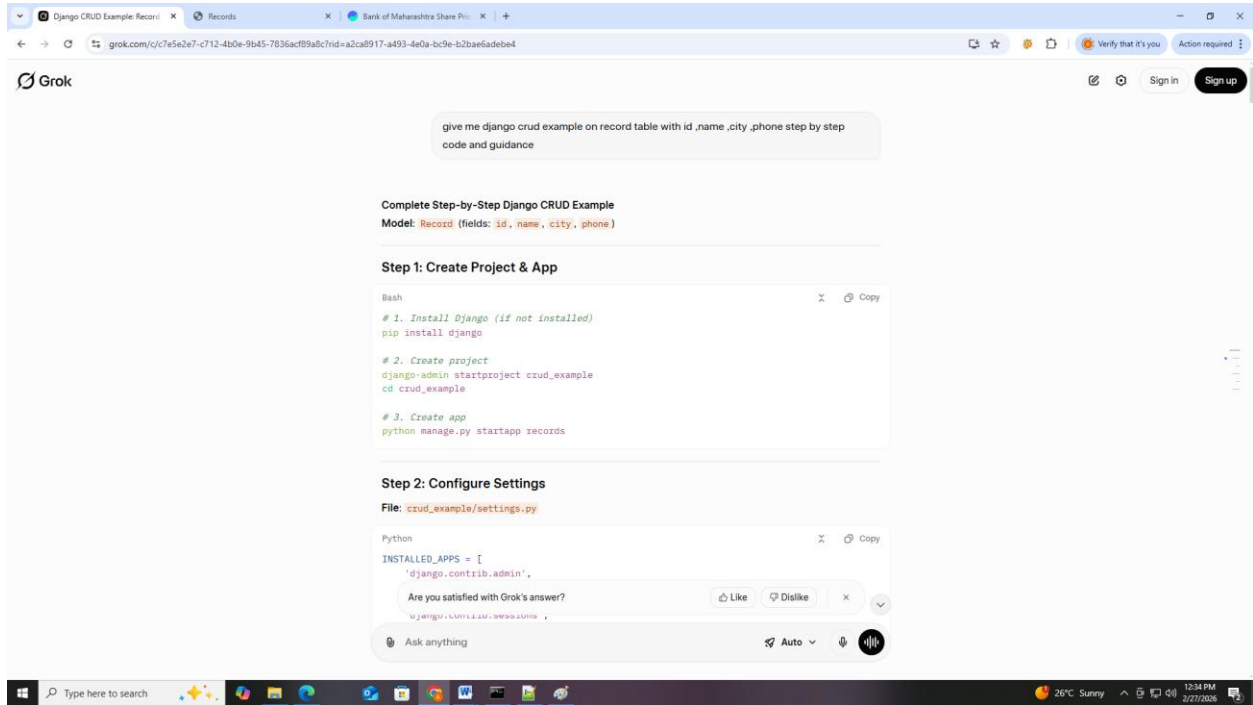


Grok.com Testing for coding Guidance:-



The screenshot shows the Grok.com interface. At the top, there's a search bar with the text "give me django crud example on record table with id ,name ,city ,phone step by step code and guidance". Below this, the response is titled "Complete Step-by-Step Django CRUD Example" with a model definition: "Model: Record (fields: id, name, city, phone)".

Step 1: Create Project & App

```
Bash
# 1. Install Django (if not installed)
pip install django

# 2. Create project
django-admin startproject crud_example
cd crud_example

# 3. Create app
python manage.py startapp records
```

Step 2: Configure Settings

File: crud_example/settings.py

```
Python
INSTALLED_APPS = [
    'django.contrib.admin',
```

Below the code, there's a feedback section: "Are you satisfied with Grok's answer?" with "Like" and "Dislike" buttons. At the bottom, there's an "Ask anything" input field and an "Auto" dropdown menu.

Complete Step-by-Step Django CRUD Example Model: Record (fields: id, name, city, phone)

Step 1: Create Project & App

Bash

```
# 1. Install Django (if not installed)
pip install django

# 2. Create project
django-admin startproject crud_example
cd crud_example

# 3. Create app
python manage.py startapp records
```

Step 2: Configure Settings

File: crud_example/settings.py

Python

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'records',          # ← Add this  
]
```

Step 3: Create the Model

File: records/models.py

Python

```
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=15)  
  
    def __str__(self):  
        return self.name  
  
    class Meta:  
        ordering = ['name']
```

Run migrations:

Bash

```
python manage.py makemigrations  
python manage.py migrate
```

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Big Data>d:

D:\>env\scripts\activate

(env) D:\>django-admin startproject crud_example

(env) D:\>cd crud_example

(env) D:\crud_example>python manage.py startapp records

(env) D:\crud_example>python manage.py makemigrations
Migrations for 'records':
  records\migrations\0001_initial.py
    + Create model Record

(env) D:\crud_example>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, records, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying records.0001_initial... OK
  Applying sessions.0001_initial... OK

(env) D:\crud_example>

```

Step 4: Create Class-Based Views (CRUD)

File: records/views.py

Python

```

from django.views.generic import ListView, DetailView, CreateView,
UpdateView, DeleteView

from django.urls import reverse_lazy

from .models import Record

```

```
class RecordListView(ListView):
    model = Record
    template_name = 'records/record_list.html'
    context_object_name = 'records'

class RecordDetailView(DetailView):
    model = Record
    template_name = 'records/record_detail.html'

class RecordCreateView(CreateView):
    model = Record
    fields = ['name', 'city', 'phone']
    template_name = 'records/record_form.html'
    success_url = reverse_lazy('record_list')

class RecordUpdateView(UpdateView):
    model = Record
    fields = ['name', 'city', 'phone']
    template_name = 'records/record_form.html'
    success_url = reverse_lazy('record_list')

class RecordDeleteView(DeleteView):
    model = Record
    template_name = 'records/record_confirm_delete.html'
    success_url = reverse_lazy('record_list')
```

Step 5: Set Up URLs

File: records/urls.py (create this file)

Python

```
from django.urls import path
from .views import *
```

```
urlpatterns = [
```

```
    path('', RecordListView.as_view(), name='record_list'),
    path('record/<int:pk>/', RecordDetailView.as_view(),
name='record_detail'),
    path('record/new/', RecordCreateView.as_view(), name='record_create'),
    path('record/<int:pk>/edit/', RecordUpdateView.as_view(),
name='record_update'),
    path('record/<int:pk>/delete/', RecordDeleteView.as_view(),
name='record_delete'),
]
```

File: crud_example/urls.py

Python

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('records.urls')), # ← All records URLs at root
]
```

Step 6: Create Templates

Create folder structure:

records/templates/records

1. records/templates/records/record_list.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Records</title>
```

```
<style>
```

```
body { font-family: Arial, sans-serif; margin: 20px; }
```

```
table {
```

```
border-collapse: collapse;
```

```
width: 100%;
```

```
max-width: 900px;
```

```
margin: 20px 0;
```

```
}
```

```
th, td {
```

```
border: 1px solid #ddd;
```

```
padding: 10px;
```

```
text-align: left;
```

```
}
```

```
th {
```

```
background-color: #f4f4f4;
```

```
font-weight: bold;
```

```
}
```

```
tr:nth-child(even) { background-color: #f9f9f9; }
```

```
a { color: #0066cc; text-decoration: none; }
```

```
a:hover { text-decoration: underline; }
```

```
.actions a { margin-right: 8px; }
```

```
.add-btn {
```

```
display: inline-block;
```

```
padding: 8px 16px;

background-color: #28a745;

color: white;

border-radius: 4px;

text-decoration: none;

margin-bottom: 15px;

}

.add-btn:hover { background-color: #218838; }

</style>

</head>

<body>

<h1>All Records</h1>

<a href="{% url 'record_create' %}" class="add-btn">+ Add New Record</a>

<table>

<thead>

<tr>

<th>#</th>    <!-- Changed from ID to visual row number -->

<th>Name</th>

<th>City</th>

<th>Phone</th>

<th>Actions</th>

</tr>
```

```
</thead>
<tbody>
  {% for record in records %}
  <tr>
    <td>{{ forloop.counter }}</td> <!-- 1, 2, 3, ... on this page -->
    <td>{{ record.name }}</td>
    <td>{{ record.city }}</td>
    <td>{{ record.phone }}</td>
    <td class="actions">
      <a href="{% url 'record_detail' record.pk %}">View</a> |
      <a href="{% url 'record_update' record.pk %}">Edit</a> |
      <a href="{% url 'record_delete' record.pk %}">Delete</a>
    </td>
  </tr>
  {% empty %}
  <tr>
    <td colspan="5" style="text-align:center; color:#777; padding:30px;">
      No records found.
    </td>
  </tr>
  {% endfor %}
</tbody>
</table>

</body>
```

```
</html>
```

2. records/templates/records/record_detail.html

```
<!DOCTYPE html>
<html>
<head><title>Record Detail</title></head>
<body>
  <h1>Record #{{ object.id }}</h1>
  <p><strong>Name:</strong> {{ object.name }}</p>
  <p><strong>City:</strong> {{ object.city }}</p>
  <p><strong>Phone:</strong> {{ object.phone }}</p>
  <hr>
  <a href="{% url 'record_list' %}">Back to List</a>
</body>
</html>
```

3. records/templates/records/record_form.html (used by Create & Update)

HTML

```
<!DOCTYPE html>
<html>
<head><title>{% if object %}Edit{% else %}New{% endif %}
Record</title></head>
<body>
  <h1>{% if object %}Edit{% else %}Create{% endif %} Record</h1>
  <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Save</button>
    <a href="{% url 'record_list' %}">Cancel</a>
  </form>
</body>
</html>
```

4. records/templates/records/record_confirm_delete.html

HTML

```
<!DOCTYPE html>
<html>
<head><title>Delete Record</title></head>
<body>
  <h1>Delete Record #{{ object.id }}</h1>
  <p>Are you sure you want to delete <strong>{{ object.name
}}</strong>?</p>
  <form method="post">
    {% csrf_token %}
    <button type="submit">Yes, Delete</button>
    <a href="{% url 'record_list' %}">Cancel</a>
  </form>
</body>
</html>
```

Step 7: (Optional) Register in Admin

File: records/admin.py

Python

```
from django.contrib import admin
from .models import Record

admin.site.register(Record)
```

Create superuser:

Bash

```
python manage.py createsuperuser
```

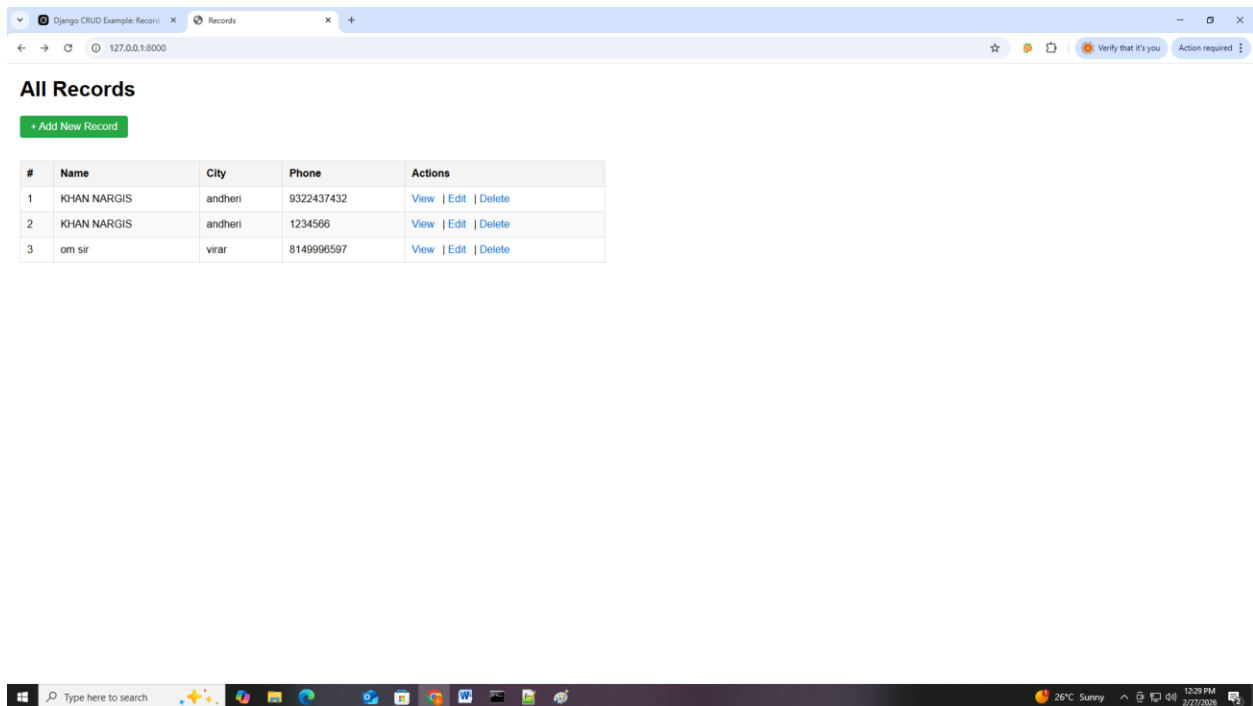
Step 8: Run the Project

Bash

```
python manage.py runserver
```

Open browser: <http://127.0.0.1:8000/>

You will see the full CRUD interface.



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000`. The page title is "All Records". Below the title is a green button labeled "+ Add New Record". A table displays the following records:

#	Name	City	Phone	Actions
1	KHAN NARGIS	andheri	9322437432	View Edit Delete
2	KHAN NARGIS	andheri	1234566	View Edit Delete
3	om sir	virar	8149996597	View Edit Delete

The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray with the date and time: 12:39 PM, 2/27/2025, 26°C Sunny.