The **Complete updated CRUD project using ModelForm** with proper clean structure and best practice.

We'll build:

```
Model: Record (id, name, city)
CRUD using Class-Based Views
Using ModelForm (recommended way)
```

Here's a **step-by-step guide** to building a CRUD app in **Django using Class-Based Views (CBVs)** for a model with fields:

- `id`
- `name`
- `city`

We'll build a simple app called **records**.

□ Step 1: Install & Create Django Project

Create Environment:

□ Windows(CMD)

```
python -m venv venv
```

Activate Virtual Environment

□ Windows (CMD)

```
venv\Scripts\activate
```

1 Install Django

```
pip install django
```

2 Create Project

```
django-admin startproject myproject
cd myproject
```

3 Create App

```
python manage.py startapp records
```

4 Register App in your Project:-

In your myproject folder open `settings.py and add 'records' as shown below`:

```
INSTALLED_APPS = [
    ...
    'records',
]
```

Step 2: Create Model

---

1 models.py

`records/models.py`

```
from django.db import models

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

    def __str__(self):
        return f"{self.name} - {self.city}"
```

Notes

- `id` → Automatically created by Django.
- `__str__()` → Better display in admin/shell.

Create & Apply Migrations

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

□ 2□ forms.py (Using ModelForm)

□ records/forms.py

```
from django import forms
from .models import Record

class RecordForm(forms.ModelForm):
    class Meta:
        model = Record
        fields = ['name', 'city']
        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control'}),
            'city': forms.TextInput(attrs={'class': 'form-control'}),
        }
```

□ Why ModelForm?

- Automatically creates form fields from model.
- Handles validation.
- Saves data easily using `form.save()`.

□ 3□ views.py (All CRUD with ModelForm)

□ records/views.py

```
from django.urls import reverse_lazy
from django.views.generic import (
    ListView,
    DetailView,
    CreateView,
    UpdateView,
    DeleteView
)
```

```
from .models import Record
from .forms import RecordForm
```

□ List View

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

□ Detail View

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

□ Create View

```
class RecordCreateView(CreateView):
    model = Record
    form_class = RecordForm
    template_name = 'records/record_form.html'
    success_url = reverse_lazy('record-list')
```

□ Update View

```
class RecordUpdateView(UpdateView):
    model = Record
    form_class = RecordForm
    template_name = 'records/record_form.html'
    success_url = reverse_lazy('record-list')
```

□ Delete View

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

□ 4□ urls.py (App Level)

□ `records/urls.py`

```
from django.urls import path
from .views import (
    RecordListView,
    RecordDetailView,
    RecordCreateView,
    RecordUpdateView,
    RecordDeleteView
)

urlpatterns = [
    path('', RecordListView.as_view(), name='record-list'),
    path('record/<int:pk>/', RecordDetailView.as_view(), name='record-
detail'),
    path('record/create/', RecordCreateView.as_view(), name='record-create'),
    path('record/<int:pk>/update/', RecordUpdateView.as_view(), name='record-
update'),
    path('record/<int:pk>/delete/', RecordDeleteView.as_view(), name='record-
delete'),
]
```

□ 5□ Main Project urls.py

□ `myproject/urls.py`

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

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

□ 6□ Templates

□ Folder Structure:

```
records/
    templates/
        records/
            record_list.html
            record_detail.html
            record_form.html
            record_confirm_delete.html
```

□ record_list.html

```
<h1>All Records</h1>

<a href="{% url 'record-create' %}">Add New Record</a>

<hr>

<ul>
    {% for record in records %}
        <li>
            {{ record.name }} - {{ record.city }}
            <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>
        </li>
    {% empty %}
        <li>No records available.</li>
    {% endfor %}
</ul>
```

□ record_detail.html

```
<h2>Record Details</h2>

<p><strong>Name:</strong> {{ record.name }}</p>
<p><strong>City:</strong> {{ record.city }}</p>

<a href="{% url 'record-update' record.pk %}">Edit</a> |
<a href="{% url 'record-list' %}">Back to List</a>
```

□ record_form.html

```
<h2>Record Form</h2>

<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Save</button>
</form>

<br>
<a href="{% url 'record-list' %}">Cancel</a>
```

□ record_confirm_delete.html

```
<h2>Delete Record</h2>

<p>Are you sure you want to delete "{{ object.name }}"?</p>

<form method="post">
    {% csrf_token %}
    <button type="submit">Confirm Delete</button>
</form>

<br>
<a href="{% url 'record-list' %}">Cancel</a>
```

□ 7□ Run Migrations

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

□ How This Works Internally

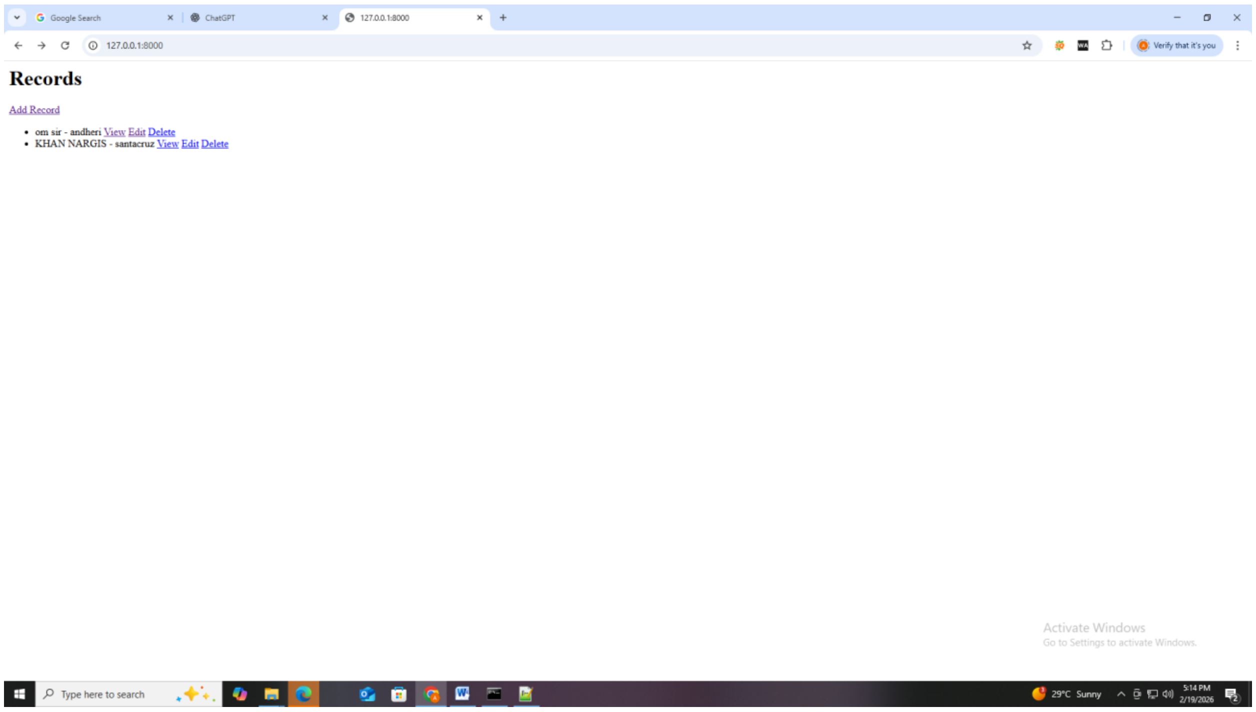
View	What Happens
CreateView	GET → Show empty form, POST → Save new record
UpdateView	GET → Show filled form, POST → Update record
DeleteView	GET → Show confirm page, POST → Delete
ListView	Fetch all records
DetailView	Fetch single record

ModelForm automatically:

- Validates fields
- Saves model instance
- Handles errors

□ Final Flow

```
/                    → List all
```

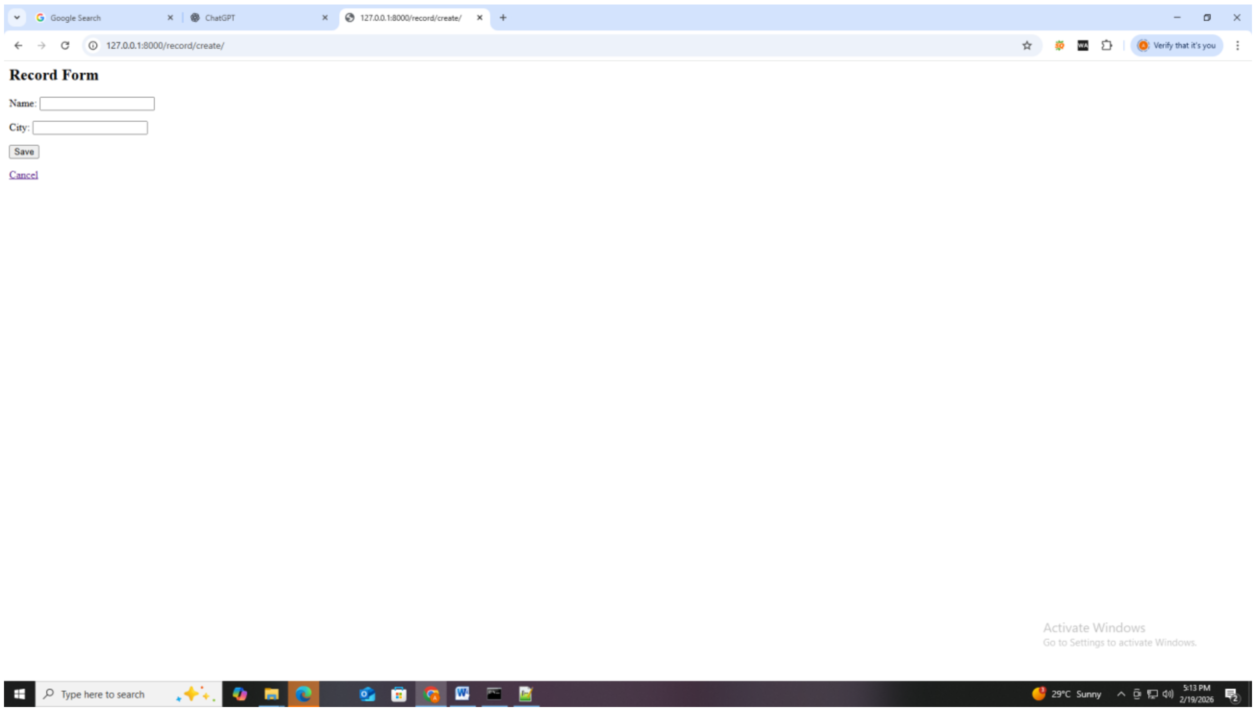


Records

Add Record

- om sir - andheri View Edit Delete
- KHAN NARGIS - santacruz View Edit Delete

```
/record/create/  → Add new
```



Record Form

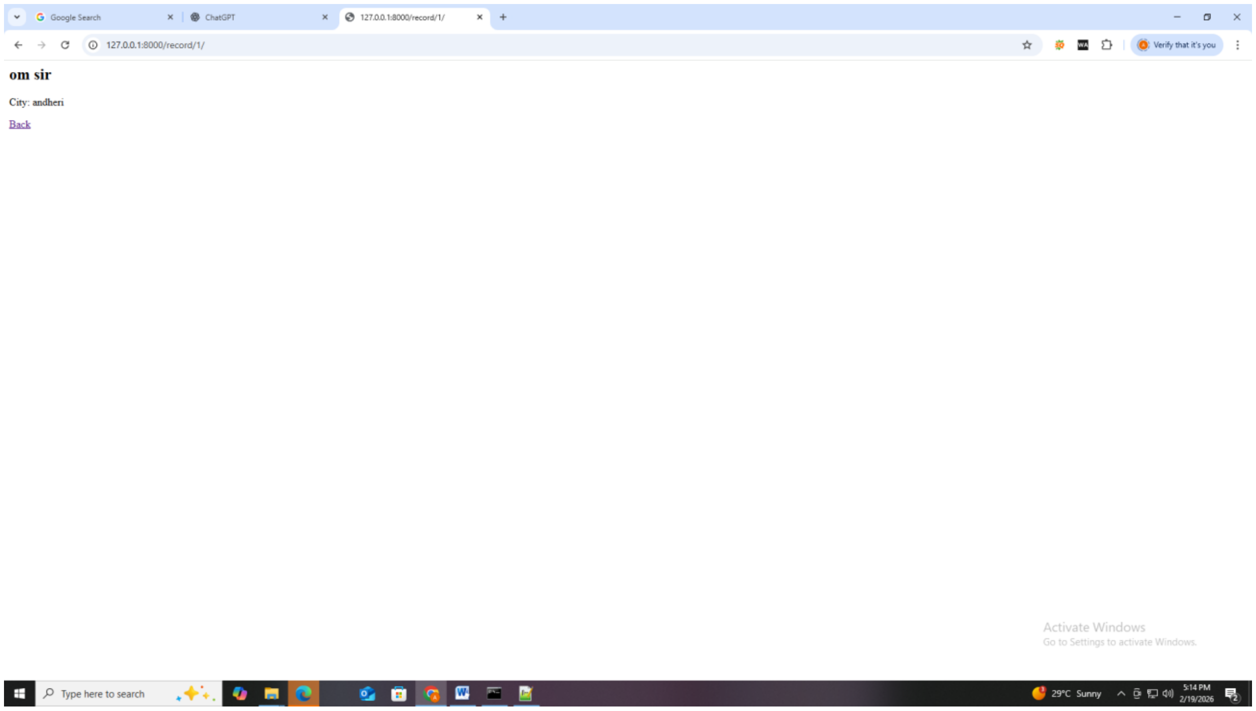
Name:

City:

Save

Cancel

```
/record/1/        → View record
```

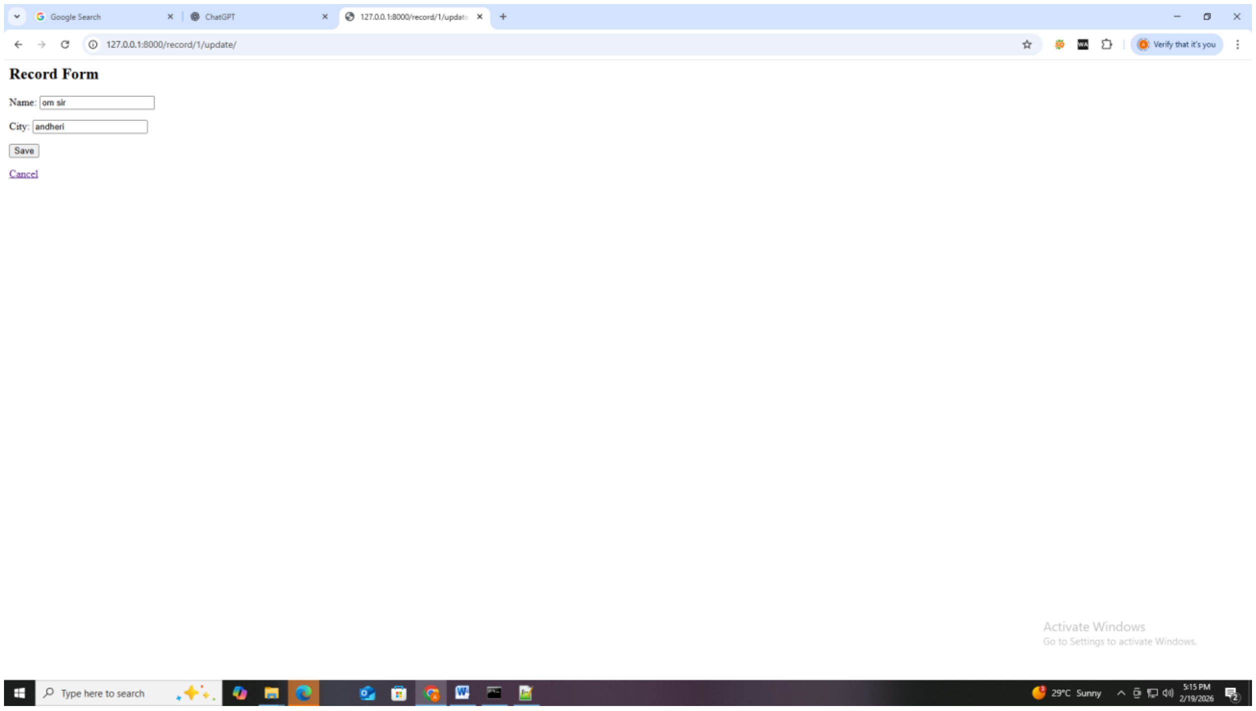


om sir

City: andheri

Back

```
/record/1/update/ → Edit
```



Record Form

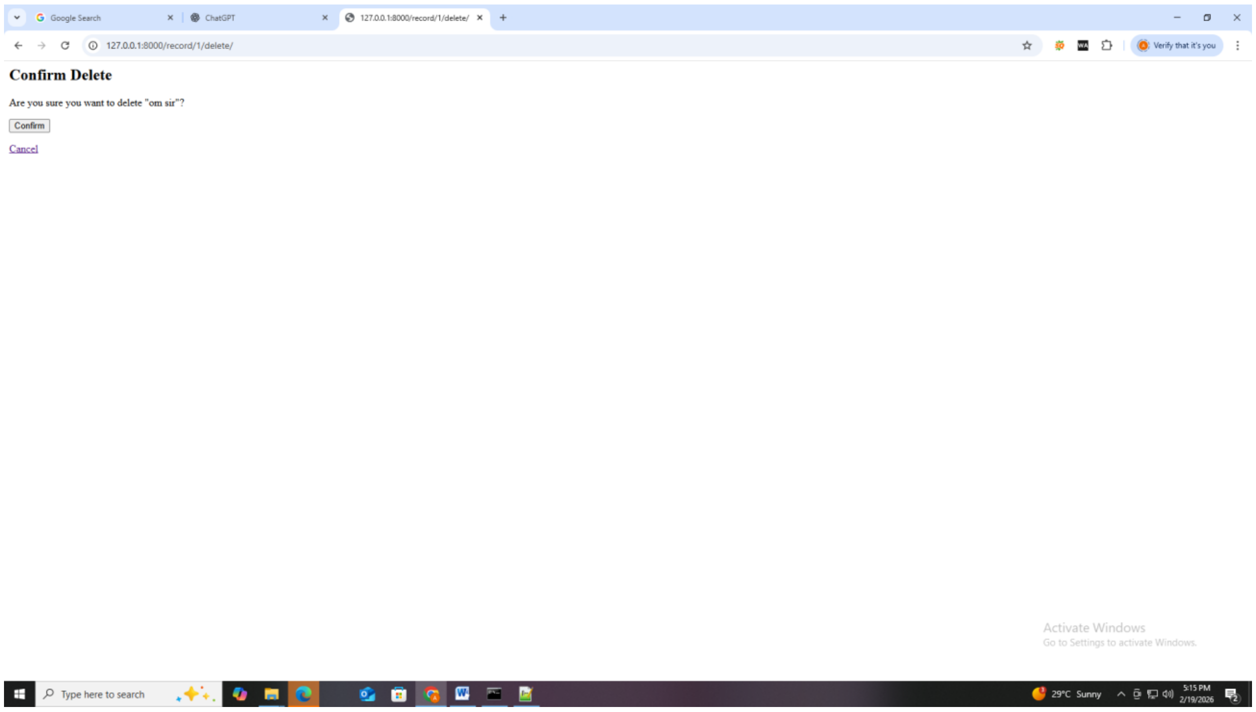
Name: om sir

City: andheri

Save

Cancel

/record/1/delete/ → Delete



Confirm Delete

Are you sure you want to delete "om sir"?

Confirm

Cancel