Django Crud Code Explanation in Marathi:-

records/models.py file code:-

खाली तुमचा `models.py` कोड **संपूर्ण comment वगळून clean आणि aligned स्वरूपात** दिला आहे

```
from django.db import models

class Record(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    phone = models.CharField(max_length=15)
    city = models.CharField(max_length=100)

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

खाली दिलेल्या Django `models.py` कोडमधील **प्रत्येक ओळीचे सविस्तर मराठीत स्पष्टीकरण** देत आहे

---

1) Import Statement

```
from django.db import models
```

- Django च्या `db` (database) module मधून `models` import करतो.
- `models` मध्ये database table तयार करण्यासाठी लागणारे सर्व field types (CharField, EmailField इ.) असतात.
- Model तयार करण्यासाठी ही ओळ आवश्यक आहे.

---

2) Model Class तयार करणे

```
class Record(models.Model):
```

- `Record` नावाचा एक Model (Database Table) तयार केला आहे.
- `models.Model` हा Django चा base class आहे.
- प्रत्येक model class म्हणजे database मधील एक table असते.

- `Record` हे table चे नाव होईल (database मध्ये साधारणपणे `appname_record` असे नाव तयार होते).

---

3) Fields (टेबलमधील कॉलम्स)

Name Field

```
name = models.CharField(max_length=100)
```

- `name` नावाचा column तयार होतो.
- `CharField` म्हणजे short text साठवण्यासाठी field.
- `max_length=100` म्हणजे जास्तीत जास्त 100 अक्षरे ठेवता येतील.
- हा field default ने required असतो (रिकामा ठेवू शकत नाही).

Email Field

```
email = models.EmailField()
```

- `email` नावाचा column तयार होतो.
- `EmailField` email address साठवतो.
- Django आपोआप email format validate करतो (उदा. abc@gmail.com योग्य आहे का ते तपासतो).

Phone Field

```
phone = models.CharField(max_length=15)
```

- `phone` नावाचा column तयार होतो.
- Phone number text स्वरूपात साठवला जातो.
- `max_length=15` म्हणजे जास्तीत जास्त 15 अक्षरे ठेवता येतील.
- Phone number साठी `CharField` वापरतो कारण त्यात `+`, `-`, space असू शकतात.

City Field

```
city = models.CharField(max_length=100)
```

- `city` नावाचा column तयार होतो.
- City नाव text स्वरूपात साठवतो.
- जास्तीत जास्त 100 अक्षरे ठेवता येतात.

---

4) str Method

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

- `__str__()` हा Python चा special method आहे.
- Object print केल्यावर काय दिसेल ते ठरवतो.
- जर आपण `Record` object admin panel किंवा shell मध्ये पाहिला, तर `Record object (1)` ऐवजी त्या व्यक्तीचे `name` दिसेल.
- त्यामुळे readability वाढते.

हा Model प्रत्यक्षात काय करतो?

हा model database मध्ये खालीलप्रमाणे table तयार करतो:

id name email phone city

`id` field Django आपोआप तयार करतो (Primary Key म्हणून).

STEP 5 — Run Migrations (Database मध्ये बदल लागू करणे)

Model तयार केल्यानंतर database मध्ये table तयार करण्यासाठी migrations करणे आवश्यक आहे.

---

1) Make Migrations

```
python manage.py makemigrations
```

- Django model मधील बदल ओळखतो.
- Migration file तयार करतो.
- ही file database मध्ये कोणते बदल करायचे ते सांगते.
- अजून database मध्ये table तयार होत नाही, फक्त plan तयार होतो.

□ 2) Migrate

```
python manage.py migrate
```

- Migration file वापरून database मध्ये प्रत्यक्ष table तयार करतो.
- म्हणजेच `Record` नावाचा table तयार होतो.

□ Simple Flow

1 □ Model तयार करा
2 □ `makemigrations` चालवा
3 □ `migrate` चालवा
4 □ Database मध्ये table तयार

Records/forms.py file code:-

खाली तुमचा `forms.py` कोड **योग्य alignment (indentation)** सह आणि **प्रत्येक ओळीचे सविस्तर मराठीत स्पष्टीकरण** दिले आहे □

□ Proper Aligned Code (`forms.py`)

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


class RecordForm(forms.ModelForm):

    class Meta:
```

```
        model = Record
        fields = ['name', 'email', 'phone', 'city']
```

□ प्रत्येक ओळीचे सविस्तर स्पष्टीकरण (मराठीत)

□ 1) Import Forms Module

```
from django import forms
```

- Django मधील `forms` module import करतो.
- Form तयार करण्यासाठी आवश्यक classes येथे असतात.
- User कडून data घेण्यासाठी Django Forms वापरले जातात.
- Validation (data योग्य आहे का तपासणे) आपोआप होते.

□ 2) Import Model

```
from .models import Record
```

- आपल्या app मधील `models.py` फाईलमधून `Record` model import करतो.
- . म्हणजे current folder (आपला app).
- हा model आपण form सोबत जोडणार आहोत.

□ 3) Form Class तयार करणे

```
class RecordForm(forms.ModelForm):
```

- `RecordForm` नावाचा form तयार केला आहे.
- तो `forms.ModelForm` पासून inherit केला आहे.

ModelForm म्हणजे काय?

- `ModelForm` हा Django चा special form आहे.
- तो थेट Model शी जोडलेला असतो.
- Model मधील fields वापरून आपोआप form तयार करतो.

- त्यामुळे वेगळे fields manually लिहावे लागत नाहीत.

□ 4) Meta Class

```
class Meta:
```

- `Meta` ही inner class आहे.
- Django ला form बद्दल अतिरिक्त माहिती देते.
- ModelForm साठी `Meta` class आवश्यक असते.

□ 5) Model Specify करणे

```
model = Record
```

- हा form कोणत्या model वर आधारित आहे ते सांगतो.
- येथे `Record` model वापरला आहे.
- म्हणजे form मधील data `Record` table मध्ये save होईल.

□ 6) Fields Specify करणे

```
fields = ['name', 'email', 'phone', 'city']
```

- Form मध्ये कोणते fields दाखवायचे ते येथे लिहिले आहे.
- हे fields `Record` model मधून घेतले जातात.
- याच क्रमाने form मध्ये fields दिसतील.
- जर एखादा field इथे लिहिला नाही तर तो form मध्ये दिसणार नाही.

□ हा Form प्रत्यक्षात काय करतो?

□ `Record` model मधील fields वापरून form तयार करतो
□ User कडून data घेतो
□ Data validate करतो
□ `form.save()` केल्यावर database मध्ये save करतो

□ Simple Flow

1 □ Model तयार
2 □ ModelForm तयार
3 □ View मध्ये `RecordForm()` वापरला
4 □ Template मध्ये `{{ form.as_p }}` वापरला
5 □ Data database मध्ये save झाला

records/views.py file:-

Imports (आवश्यक मॉड्यूल्स इम्पोर्ट करणे)

```
from django.shortcuts import render, redirect, get_object_or_404
from .models import Record
from .forms import RecordForm
```

□ CREATE + READ

□ 1) Record List (सर्व रेकॉर्ड दाखवणे)

```
def record_list(request):
    records = Record.objects.all()
    return render(request, 'records/record_list.html', {'records': records})
```

□ 2) Record Create (नवीन रेकॉर्ड तयार करणे)

```
def record_create(request):
    if request.method == 'POST':
        form = RecordForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('record_list')
    else:
        form = RecordForm()
```

```
    return render(request, 'records/record_form.html', {'form': form})
```

□ UPDATE (रेकॉर्ड अपडेट करणे)

```
def record_update(request, pk):
    record = get_object_or_404(Record, pk=pk)

    if request.method == 'POST':
        form = RecordForm(request.POST, instance=record)
        if form.is_valid():
            form.save()
            return redirect('record_list')
    else:
        form = RecordForm(instance=record)

    return render(request, 'records/record_form.html', {'form': form})
```

□ DELETE (रेकॉर्ड delete करणे)

```
def record_delete(request, pk):
    record = get_object_or_404(Record, pk=pk)

    if request.method == 'POST':
        record.delete()
        return redirect('record_list')

    return render(request, 'records/record_confirm_delete.html', {'record':
record})
```

खाली दिलेल्या Django कोडमधील **प्रत्येक ओळीचे सविस्तर मराठीत स्पष्टीकरण** देत आहे

□ Imports (आवश्यक मॉड्यूल्स इम्पोर्ट करणे)

```
from django.shortcuts import render, redirect, get_object_or_404
```

- `render` → HTML template आणि data एकत्र करून browser ला response पाठवण्यासाठी वापरले जाते.
- `redirect` → एका URL वरून दुसऱ्या URL वर पाठवण्यासाठी वापरले जाते.
- `get_object_or_404` → Database मधून object शोधतो; object न मिळाल्यास 404 error दाखवतो.

```
from .models import Record
```

- आपल्या app मधील `models.py` फाईलमधून `Record` model import करतो.
- `.` म्हणजे current folder (सध्याचा app).

```
from .forms import RecordForm
```

- `forms.py` मधून `RecordForm` import करतो.
- हा form Create आणि Update साठी वापरला जातो.

□ CREATE + READ

□ 1) Record List (सर्व रेकॉर्ड दाखवणे)

```
def record_list(request):
```

- `record_list` नावाचा function तयार केला आहे.
- `request` म्हणजे user कडून आलेली HTTP request.

```
records = Record.objects.all()
```

- `Record` model मधील सर्व records database मधून घेतो.
- `objects.all()` → सर्व data fetch करतो.

```
return render(request, 'records/record_list.html', {'records': records})
```

- `record_list.html` template ला data पाठवतो.
- `{ 'records': records }` → template मध्ये `records` नावाने data उपलब्ध होतो.

□ 2) Record Create (नवीन रेकॉर्ड तयार करणे)

```
def record_create(request):
```

- नवीन record तयार करण्यासाठी function.

```
if request.method == 'POST':
```

- जर form submit झाला असेल (POST request आली असेल) तर खालील code execute होतो.

```
form = RecordForm(request.POST)
```

- User ने भरलेला form data घेऊन `RecordForm` object तयार करतो.

```
if form.is_valid():
```

- Form मधील data योग्य आहे का ते तपासतो.

```
form.save()
```

- Data database मध्ये save करतो.

```
return redirect('record_list')
```

- Save झाल्यानंतर `record_list` page वर redirect करतो.

```
else:
    form = RecordForm()
```

- जर GET request असेल (page प्रथमच उघडला असेल) तर रिकामा form दाखवतो.

```
return render(request, 'records/record_form.html', {'form': form})
```

- `record_form.html` template ला form पाठवतो.

□ UPDATE (रेकॉर्ड अपडेट करणे)

```
def record_update(request, pk):
```

- `pk` म्हणजे primary key (record ची unique id).
- विशिष्ट record update करण्यासाठी function.

```
record = get_object_or_404(Record, pk=pk)
```

- `pk` वापरून record शोधतो.
- Record न मिळाल्यास 404 error दाखवतो.

```
if request.method == 'POST':
```

- Form submit झाला आहे का ते तपासतो.

```
form = RecordForm(request.POST, instance=record)
```

- Existing record मध्ये नवीन data भरतो.
- `instance=record` → update साठी आवश्यक.

```
if form.is_valid():
```

- Data योग्य आहे का तपासतो.

```
form.save()
```

- Updated data database मध्ये save करतो.

```
return redirect('record_list')
```

- Update नंतर list page वर redirect करतो.

```
else:
    form = RecordForm(instance=record)
```

- GET request असल्यास जुना data form मध्ये भरून दाखवतो.

```
return render(request, 'records/record_form.html', {'form': form})
```

- Update form template ला data पाठवतो.

□ DELETE (रेकॉर्ड delete करणे)

```
def record_delete(request, pk):
```

- विशिष्ट record delete करण्यासाठी function.

```
record = get_object_or_404(Record, pk=pk)
```

- Delete करायचा record शोधतो.
- न मिळाल्यास 404 error.

```
if request.method == 'POST':
```

- Delete confirmation form submit झाला आहे का ते तपासतो.

```
record.delete()
```

- Record database मधून delete करतो.

```
return redirect('record_list')
```

- Delete झाल्यानंतर list page वर redirect करतो.

```
return render(request, 'records/record_confirm_delete.html', {'record':
record})
```

- Delete करण्यापूर्वी confirmation page दाखवतो.
- `record_confirm_delete.html` template मध्ये record ची माहिती दाखवतो.

records/urls.py file :-

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


urlpatterns = [
path('', views.record_list, name='record_list'),
path('create/', views.record_create, name='record_create'),
path('update/<int:pk>/', views.record_update, name='record_update'),
path('delete/<int:pk>/', views.record_delete, name='record_delete'),
]
```

खाली दिलेल्या Django `urls.py` कोडमधील **प्रत्येक ओळीचे सविस्तर मराठीत स्पष्टीकरण** देत आहे

□ Imports (आवश्यक मॉड्यूल्स इम्पोर्ट करणे)

```
from django.urls import path
```

- Django मधील `urls` module मधून `path` function import करतो.
- `path()` वापरून आपण URL pattern define करतो.
- म्हणजे कोणता URL उघडल्यावर कोणता function चालेल ते ठरवतो.

```
from . import views
```

- सध्याच्या app मधील `views.py` फाईल import करतो.
- `.` म्हणजे current folder (आपला app).
- त्यामुळे आपण `views.record_list`, `views.record_create` असे functions वापरू शकतो.

□ URL Patterns

```
urlpatterns = [
```

- `urlpatterns` ही एक list आहे.
- Django या list मधील URL patterns तपासतो.

- User browser मध्ये URL टाकतो तेव्हा Django या list मधून match शोधतो.

□ 1) Record List URL

```
path('', views.record_list, name='record_list'),
```

- `''` → रिकामा path म्हणजे Home URL (उदा. `http://127.0.0.1:8000/`)
- `views.record_list` → हा function call होईल.
- `name='record_list'` → या URL ला एक नाव दिले आहे.
    - हे नाव आपण `redirect()` किंवा `{% url %}` template tag मध्ये वापरतो.

□ म्हणजे Home page उघडल्यावर `record_list` view चालेल.

□ 2) Record Create URL

```
path('create/', views.record_create, name='record_create'),
```

- `'create/'` → URL असेल `http://127.0.0.1:8000/create/`
- `views.record_create` → नवीन record तयार करण्यासाठी view function
- `name='record_create'` → या URL चे नाव

□ User `/create/` वर गेला की record create page उघडेल.

□ 3) Record Update URL

```
path('update/<int:pk>/', views.record_update, name='record_update'),
```

- `'update/<int:pk>/'` → Dynamic URL आहे.
- `<int:pk>` → integer type parameter आहे.
- `pk` म्हणजे primary key (record ची unique id).

उदा:

```
http://127.0.0.1:8000/update/5/
```

येथे `pk = 5`

- `views.record_update` → हा function त्या specific record ला update करेल.
- `name='record_update'` → URL चे नाव

 Django `pk` value view function मध्ये parameter म्हणून पाठवतो.

---

4) Record Delete URL

```
path('delete/<int:pk>/', views.record_delete, name='record_delete'),
```

- `'delete/<int:pk>/'` → Dynamic URL
- `<int:pk>` → integer id घेते
- `views.record_delete` → specific record delete करतो
- `name='record_delete'` → URL चे नाव

उदा:

```
http://127.0.0.1:8000/delete/3/
```

येथे `pk = 3`

```
]
```

- `urlpatterns` list येथे संपते.

संपूर्ण कोड काय करतो?

हा `urls.py` फाईल Django मध्ये **URL Routing System** तयार करतो.

URL	कोणता Function चालेल	काम
`/`	record_list	सर्व records दाखवणे
`/create/`	record_create	नवीन record तयार करणे
`/update/1/`	record_update	record update करणे
`/delete/1/`	record_delete	record delete करणे

records/templates/records:-

HTML templates **योग्य indentation (alignment) आणि clean formatting** सह दिले आहेत

---

1) `record_list.html`

```
<!DOCTYPE html>
<html>
<head>
    <title>Record List</title>
</head>
<body>

    <h2>Record List</h2>

    <a href="{% url 'record_create' %}">Add New Record</a>

    <br><br>

    <table border="1">
        <tr>
            <th>Name</th>
            <th>Email</th>
            <th>Phone</th>
            <th>City</th>
            <th>Actions</th>
        </tr>

        {% for record in records %}
        <tr>
            <td>{{ record.name }}</td>
            <td>{{ record.email }}</td>
            <td>{{ record.phone }}</td>
            <td>{{ record.city }}</td>
            <td>
                <a href="{% url 'record_update' record.id %}">Edit</a>
                <a href="{% url 'record_delete' record.id %}">Delete</a>
            </td>
        </tr>
        {% endfor %}

    </table>

</body>
</html>
```

---

2) record_form.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Record Form</title>
</head>
<body>

    <h2>Record Form</h2>

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

    <br>

    <a href="{% url 'record_list' %}">Back to List</a>

</body>
</html>
```

---

3) record_confirm_delete.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Delete Record</title>
</head>
<body>

    <h2>Are you sure you want to delete "{{ record.name }}"?</h2>

    <form method="POST">
        {% csrf_token %}
        <button type="submit">Yes, Delete</button>
    </form>

    <br>

    <a href="{% url 'record_list' %}">Cancel</a>

</body>
</html>
```

खाली दिलेल्या तीनही HTML templates मधील **प्रत्येक ओळीचे सविस्तर मराठीत स्पष्टीकरण** देत आहे

□ 1) `record_list.html`

□ Path: `records/templates/records/record_list.html`

```
<!DOCTYPE html>
```

- हा HTML5 document आहे हे browser ला सांगतो.

```
<html>
```

- HTML page ची सुरुवात.

```
<head>
    <title>Record List</title>
</head>
```

- `<head>` मध्ये page ची माहिती असते.
- `<title>` browser च्या tab मध्ये दिसणारे नाव.

```
<body>
```

- Page चा मुख्य content येथे लिहिला जातो.

```
<h2>Record List</h2>
```

- Page वर heading दाखवते.

```
<a href="{% url 'record_create' %}">Add New Record</a>
```

- "Add New Record" नावाची link तयार करते.
- `{% url 'record_create' %}` → Django template tag आहे.
- `record_create` URL ला redirect करते.

```
<br><br>
```

- दोन line break (spacing साठी).

```
<table border="1">
```

- Border असलेली table तयार करते.

```
<tr>
```

- Table row सुरू.

```
<th>Name</th>
<th>Email</th>
<th>Phone</th>
<th>City</th>
<th>Actions</th>
```

- `<th>` म्हणजे table heading.
- Columns ची नावे.

```
</tr>
```

- Row बंद.

```
{% for record in records %}
```

- Django loop सुरू.
- `records` हा view मधून आलेला data आहे.
- प्रत्येक `record` वर loop चालतो.

```
<tr>
```

- प्रत्येक record साठी नवीन row.

```
<td>{{ record.name }}</td>
```

- `record` मधील `name` दाखवतो.
- `{{ }}` → Django template variable.

```
<td>{{ record.email }}</td>
<td>{{ record.phone }}</td>
<td>{{ record.city }}</td>
```

- Email, Phone, City दाखवतो.

```
<td>
<a href="{% url 'record_update' record.id %}">Edit</a>
<a href="{% url 'record_delete' record.id %}">Delete</a>
</td>
```

- Edit आणि Delete links.

- `record.id` → specific record ची id पाठवतो.
- Update/Delete dynamic URL call होतो.

```
</tr>
{% endfor %}
```

- Loop संपतो.

```
</table>
</body>
</html>
```

- Table आणि page बंद.

□ 2) `record_form.html`

□ Path: `records/templates/records/record_form.html`

```
<!DOCTYPE html>
<html>
<head>
    <title>Record Form</title>
</head>
<body>
```

- HTML structure सुरू.

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

- Page heading.

```
<form method="POST">
```

- Form सुरू.
- `POST` method वापरतो (data submit करण्यासाठी).

```
{% csrf_token %}
```

- Security साठी आवश्यक.
- Cross-Site Request Forgery protection देते.
- Django मध्ये POST form साठी mandatory आहे.

```
{{ form.as_p }}
```

- Django form object render करतो.
- `as_p` → प्रत्येक field `<p>` tag मध्ये दाखवतो.

```
<button type="submit">Save</button>
```

- Save button.

```
</form>
```

- Form बंद.

```
<br>
<a href="{% url 'record_list' %}">Back to List</a>
```

- Record list page वर परत जाण्यासाठी link.

```
</body>
</html>
```

- Page समाप्त.

---

3) `record_confirm_delete.html`

 Path: `records/templates/records/record_confirm_delete.html`

```
<!DOCTYPE html>
<html>
<head>
    <title>Delete Record</title>
</head>
<body>
```

- HTML page सुरू.

```
<h2>Are you sure you want to delete "{{ record.name }}"?</h2>
```

- Delete confirmation message.
- `record.name` → कोणता record delete होणार आहे ते दाखवतो.

```
<form method="POST">
```

- Delete confirmation साठी POST form.

```
{% csrf_token %}
```

- Security token.

```
<button type="submit">Yes, Delete</button>
```

- Delete confirm button.

```
</form>
```

- Form बंद.

```
<br>
<a href="{% url 'record_list' %}">Cancel</a>
```

- Cancel केल्यास list page वर परत.

```
</body>
</html>
```

- Page समाप्त.

□ संपूर्ण Flow (Templates कसे काम करतात)

1□ `record_list.html` → सर्व records दाखवतो
2□ `record_form.html` → Create आणि Update साठी form दाखवतो
3□ `record_confirm_delete.html` → Delete confirm करतो