CRUD operations in Django

1. First, make sure you have Django installed. You can install it via pip:

```bash
pip install django
```

2. Create a new Django project:

```bash
django-admin startproject library_project
```

3. Create a new app within your project:

```bash
cd library_project
python manage.py startapp books
```

4. Define your model in `books/models.py`:

```python
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.CharField(max_length=100)
    publication_date = models.DateField()
```

5. Create and apply migrations:

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

6. Register your model in the admin panel:

```python
# In books/admin.py

from django.contrib import admin
from .models import Book

admin.site.register(Book)
```

7. Create views for CRUD operations in `books/views.py`:

```python
from django.shortcuts import render, redirect, get_object_or_404
from .models import Book
from .forms import BookForm

def book_list(request):
    books = Book.objects.all()
    return render(request, 'books/book_list.html', {'books': books})

def book_detail(request, pk):
    book = get_object_or_404(Book, pk=pk)
    return render(request, 'books/book_detail.html', {'book': book})


def book_create(request):
    if request.method == 'POST':
        form = BookForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('book_list')
    else:
        form = BookForm()
    return render(request, 'books/book_form.html', {'form': form})


def book_update(request, pk):
    book = get_object_or_404(Book, pk=pk)
    if request.method == 'POST':
        form = BookForm(request.POST, instance=book)
        if form.is_valid():
            form.save()
            return redirect('book_list')
    else:
        form = BookForm(instance=book)
    return render(request, 'books/book_form.html', {'form': form})


def book_delete(request, pk):
    book = get_object_or_404(Book, pk=pk)
    if request.method == 'POST':
        book.delete()
        return redirect('book_list')
    return render(request, 'books/book_confirm_delete.html', {'book': book})
```

8. Create forms for creating and updating books in `books/forms.py`:

```python
from django import forms
from .models import Book

class BookForm(forms.ModelForm):
    class Meta:
        model = Book
        fields = ['title', 'author', 'publication_date']
```

9. Create templates for rendering the views:

- `books/templates/books/book_list.html`
- `books/templates/books/book_detail.html`
- `books/templates/books/book_form.html`
- `books/templates/books/book_confirm_delete.html`

Ensure you have appropriate HTML templates for each view.

10. Define URLs for your views in `books/urls.py`:

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

urlpatterns = [
    path('', views.book_list, name='book_list'),
    path('detail/<int:pk>/', views.book_detail, name='book_detail'),
    path('create/', views.book_create, name='book_create'),
    path('update/<int:pk>/', views.book_update, name='book_update'),
    path('delete/<int:pk>/', views.book_delete, name='book_delete'),
]
```

11. Include your app's URLs in the project's URL configuration (`library_project/urls.py`):

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

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

Now you should have a basic Django application set up for CRUD operations on a "Book" model. You can access the CRUD operations via URLs like `/books/`, `/books/detail/<pk>/`, `/books/create/`, `/books/update/<pk>/`, and `/books/delete/<pk>/`.