

Here is a clear step-by-step guide to creating a Django project from scratch

1. Install Python

- First, make sure Python is installed: `python --version`
- If not installed, download it from the official website: [Python Software Foundation](#)

2. Create a Virtual Environment (Recommended)

A virtual environment keeps project dependencies isolated.

Create it:

- `python -m venv venv`
- Activate it:
- On Windows:
- `venv\Scripts\activate`

3. Install Django

- `pip install django`

4. Create a New Django Project

- `django-admin startproject myproject`
- Replace myproject with your desired project name.
- Move into the project folder:
- `cd myproject`

5. Run the Development Server

- `python manage.py runserver`
- Open your browser and go to:
- `http://127.0.0.1:8000/`
- You should see the Django welcome page

6. Create a Django App

- Inside your project directory:
- `python manage.py startapp myapp`
- Example:
`python manage.py startapp users`

7. Register the App

- Open:
- `myproject/settings.py`
- Add your app inside `INSTALLED_APPS`:
- `INSTALLED_APPS = [`
- `...`
- `'myapp',`
- `]`

8. Run Migrations

- **python manage.py makemigrations**
- **python manage.py migrate**

9. Create Superuser (Admin)

- **python manage.py createsuperuser**
- Then visit:
- **http://127.0.0.1:8000/admin/**

10. Python in web development (frontend/backend)

- **Frontend = HTML templates + CSS.** Example: crud_app/templates/crud_app/student_form.html.
- **Backend = Python views + models.** Views handle requests and return templates in crud_app/views.py.
- **Frontend <>Backend connection**
 - The template form submits to a URL route, which calls a view that saves data.
 - Example: **student_create** receives **POST** data, validates with **StudentForm**, and saves. See crud_app/views.py.
- **URL mapping(Connect frontend to backend)**

```
urlpatterns = [
    path("", views.student_list, name="student_list"),
    path("students/new/", views.student_create, name="student_create"),
    path("students/<int:pk>/edit/", views.student_update, name="student_update"),
    path("students/<int:pk>/delete/", views.student_delete, name="student_delete"),
]
```

11. Frontend and backend design

- **HTML form design**
 - The student registration form is built from **StudentForm** and rendered here: crud_app/templates/crud_app/student_form.html.
- **Database design**
 - The table schema is the **Student** model in crud_app/models.py.
 - Each model field becomes a database column.

12. Model Database Design

```
class Student(models.Model):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    email = models.EmailField(unique=True)
    registration_number = models.CharField(max_length=30, unique=True)
    course = models.CharField(max_length=120)
    created_at = models.DateTimeField(auto_now_add=True)
```

13. Insert, retrieve, update, delete

- **Insert (Create)**

Student Registration

Track entries, edit details, and keep the roster tidy.

Register Student

Enter accurate student details for the registry.

First name
Aime

Last name
Patrick

Email
aimepatrick@gmail.com

Registration number
25RP12345

Course
Python Fundamentals

Back to List

- o **View: crud_app/views.py**

```
def student_create(request):
    if request.method == "POST":
        form = StudentForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect("student_list")
    else:
        form = StudentForm()

    return render(
        request,
        "crud_app/student_form.html",
        {"form": form, "title": "Register Student"},
    )
```

- o **Form: crud_app/forms.py**

```
from django import forms
from .models import Student
```

```
class StudentForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = [
```

```

        "first_name",
        "last_name",
        "email",
        "registration_number",
        "course",
    ]
widgets = {
    "first_name": forms.TextInput(attrs={"placeholder": "First name"}),
    "last_name": forms.TextInput(attrs={"placeholder": "Last name"}),
    "email": forms.EmailInput(attrs={"placeholder": "email@example.com"}),
    "registration_number": forms.TextInput(
        attrs={"placeholder": "e.g. REG-2026-001"}
    ),
    "course": forms.TextInput(attrs={"placeholder": "Course name"}),
}

```

- **Retrieve (Read)**

The screenshot shows a web application titled "Student Registration". The main heading is "Student Registration" with a subtitle "Track entries, edit details, and keep the roster tidy.". Below this is a section titled "Registered Students" with a sub-subtitle "Total: 1". A table lists one student: "REG NO." (25RP12345), "NAME" (Aime Patrick), "EMAIL" (aimepatrick@gmail.com), and "COURSE" (Python Fundamentals). To the right of the table are two buttons: "Edit" and "Delete". At the top right of the "Registered Students" section is a button labeled "New Registration".

- **View** + **template:** crud_app/views.py,

crud_app/templates/crud_app/student_list.html

```

def student_list(request):
    students = Student.objects.order_by("-created_at")
    return render(request, "crud_app/student_list.html", {"students": students})

```

- **Update**

The screenshot shows a web-based application titled "Student Registration". The main title is "Student Registration" with a subtitle "Track entries, edit details, and keep the roster tidy.". Below this is a sub-section titled "Update Student" with the instruction "Enter accurate student details for the registry." A "Back to List" button is located in the top right corner of this section.

The form contains the following fields:

- First name:** Aime
- Last name:** Patrick
- Email:** aimepatrick@gmail.com
- Registration number:** 25RP00001
- Course:** Python Fundamentals

At the bottom of the form are two buttons: "Save" (highlighted in orange) and "Cancel".

- **View: crud_app/views.py**

```
def student_update(request, pk):
    student = get_object_or_404(Student, pk=pk)
    if request.method == "POST":
        form = StudentForm(request.POST, instance=student)
        if form.is_valid():
            form.save()
            return redirect("student_list")
    else:
        form = StudentForm(instance=student)

    return render(
        request,
        "crud_app/student_form.html",
        {"form": form, "title": "Update Student"},
    )
```

- **Delete**

Student Registration

Track entries, edit details, and keep the roster tidy.

Delete Student

This action cannot be undone.

[Back to List](#)

Are you sure you want to remove Aime Patrick?

[Confirm Delete](#)

[Cancel](#)

- View + confirmation page: crud_app/views.py, crud_app/templates/crud_app/student_confirm_delete.html

```
def student_delete(request, pk):
    student = get_object_or_404(Student, pk=pk)
    if request.method == "POST":
        student.delete()
        return redirect("student_list")

    return render(
        request,
        "crud_app/student_confirm_delete.html",
        {"student": student},
    )
```