**Experiment 12: Install Django and Setup a Virtual Environment**

**Course Outcomes (COs):**

* CO1: Apply knowledge of installation and environment setup to isolate dependencies for web development.

**Program Outcomes (POs):**

* PO1: Engineering knowledge
* PO5: Modern tool usage

**Program Specific Outcomes (PSOs):**

* PSO1: Apply software development fundamentals

**Objective:** Learn how to install Django in a virtual environment to isolate project dependencies.

**Steps:**

1. **Install Python (if not already installed):**
   * Ensure Python 3.x is installed by running:

python --version

* + Download from <https://python.org> if not installed.

1. **Create a project folder:**
2. mkdir my\_django\_project

cd my\_django\_project

1. **Create a virtual environment:**

python -m venv venv

* + This creates an isolated environment named venv.

1. **Activate the virtual environment:**
   * On Windows:

venv\Scripts\activate

* + On macOS/Linux:

source venv/bin/activate

* + You should see (venv) at the beginning of your terminal prompt.

1. **Install Django:**

pip install django

1. **Verify installation:**

django-admin --version

1. **Start a new Django project:**

django-admin startproject mysite .

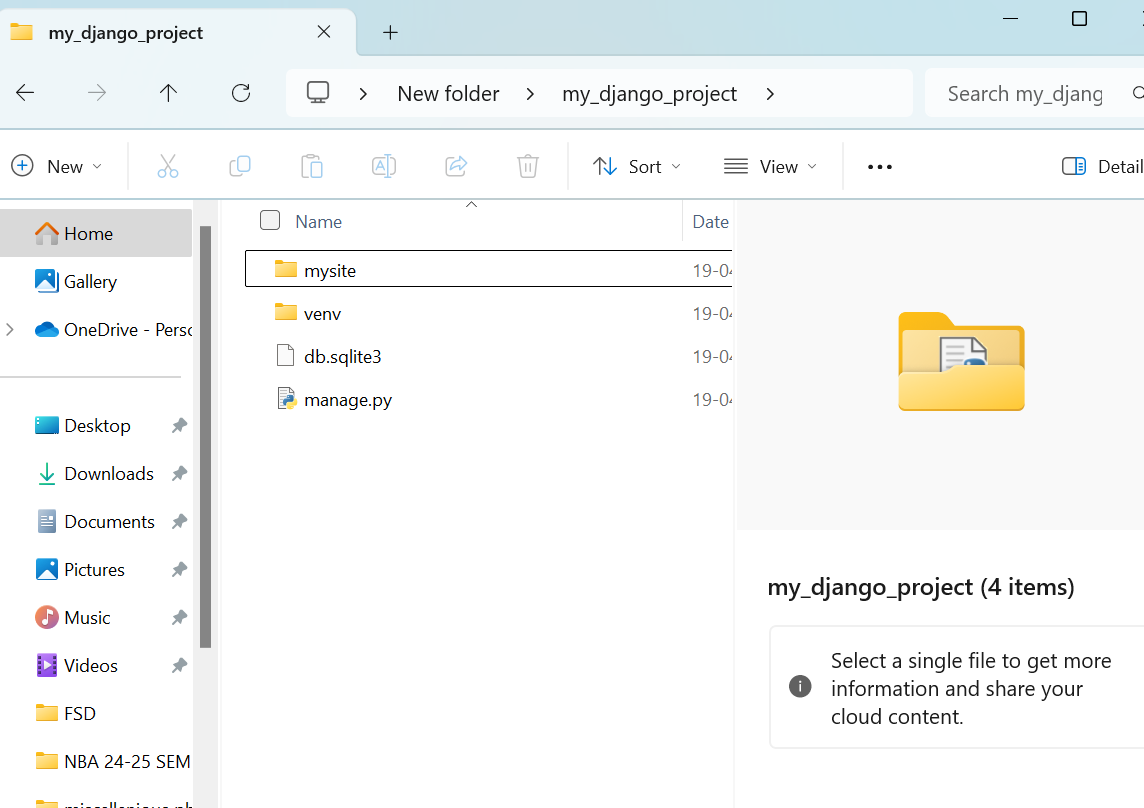
1. **Run the Django server:**

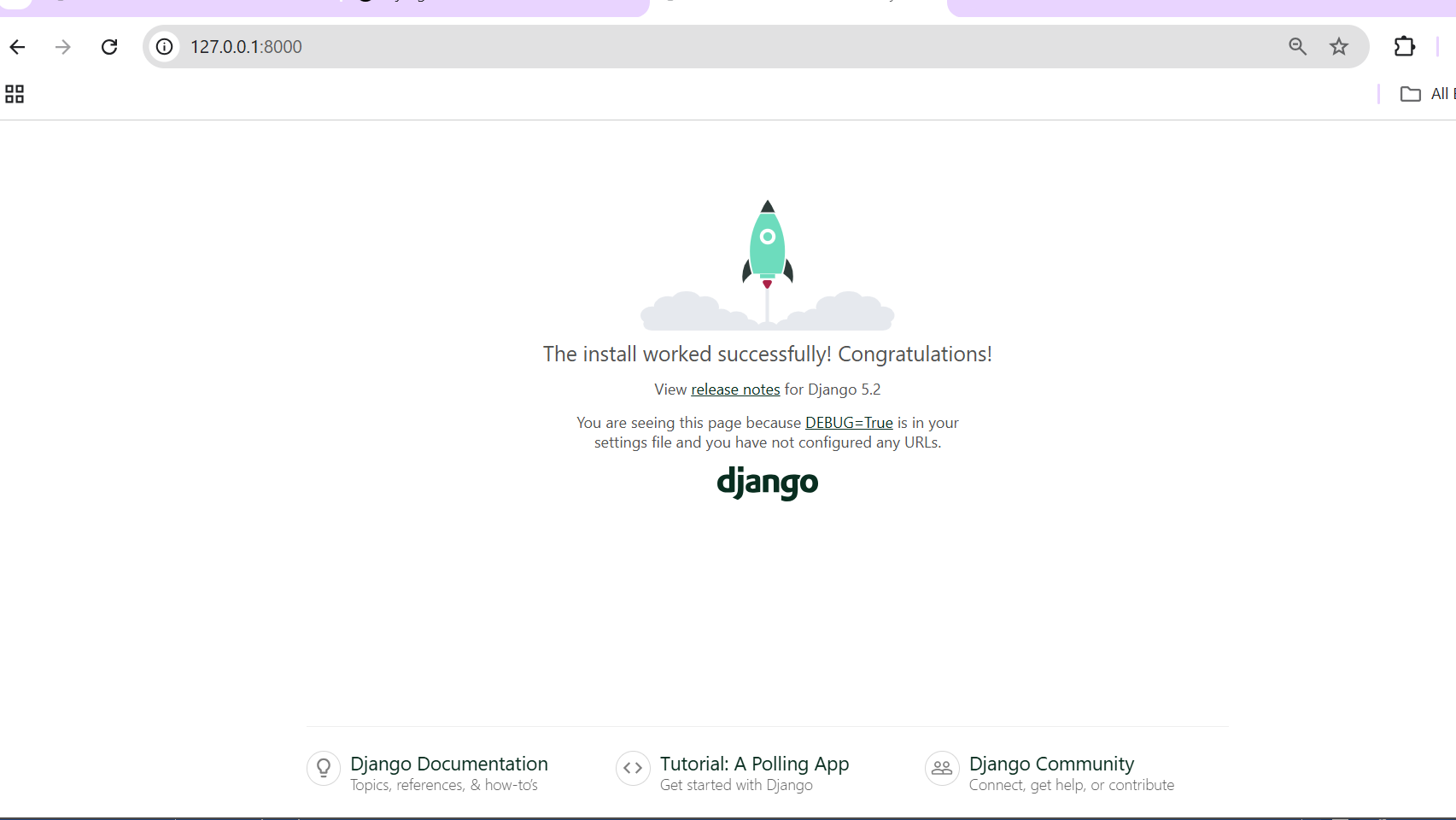
python manage.py runserver

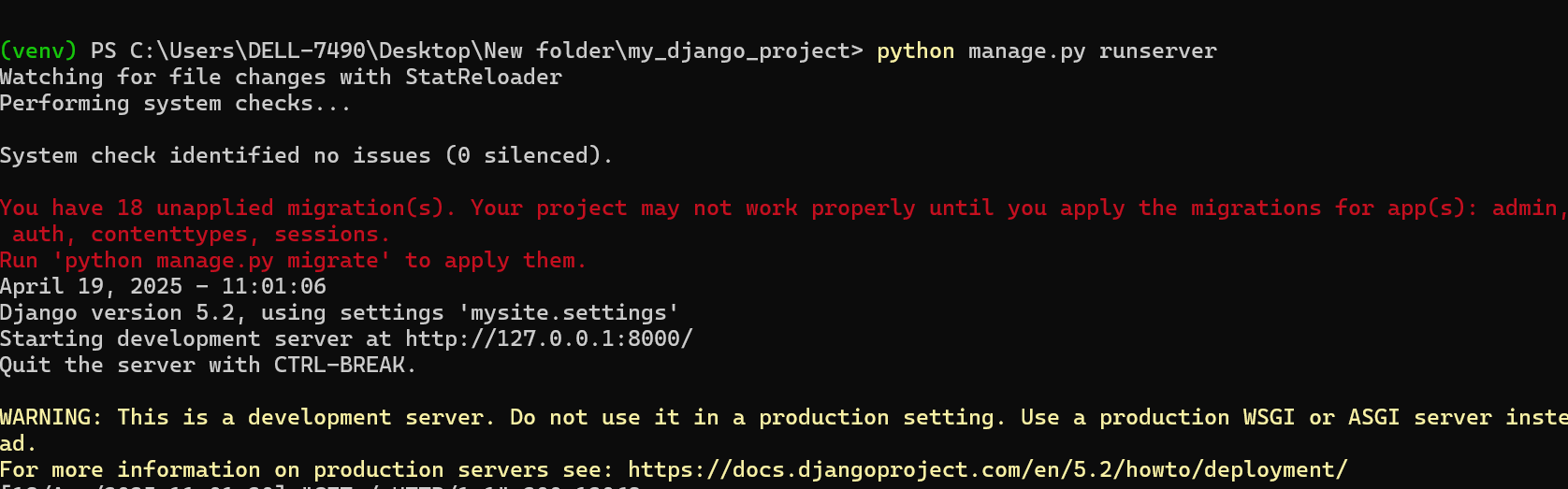
* + This starts the development server at http://127.0.0.1:8000/

**Output:**

* A Django project structure with folders: mysite, manage.py, etc.
* Running server message confirming successful setup.







## ✅ WEEK 13 – Django App with Navigation Menu (Home, Dashboard, Contact Us)

### 🎯 ****Goal****

Build a Django web application that displays:

* A **navigation bar** using Bootstrap
* Three pages: **Home**, **Dashboard**, and **Contact Us**

## 🔧 Step-by-Step Instructions

### ✅ STEP 1: Check Python Version

📍 In Command Prompt (CMD) or Terminal:

bash

python --version

Make sure it shows Python **3.x**

### ✅ STEP 2: Install Virtual Environment

bash

py -m pip install virtualenv

### What is virtualenv?

virtualenv is a **predefined tool** that allows you to create isolated Python environments. This helps in managing dependencies for different projects without causing conflicts between them. It creates a separate directory with its own Python binary and can install packages independently of the global Python environment.

**💡 What’s happening?**

* py: Runs Python.
* -m pip install virtualenv: Uses Python's package installer (pip) to **install a tool** called virtualenv.

👉 virtualenv lets you **create an isolated space** to work on your project, so it doesn't interfere with other projects or your system Python.

### ✅ STEP 3: Create Virtual Environment

bash

py -m venv second

📁 This will create a folder second/ with your isolated environment.

**💡 What’s happening?**

* venv is a built-in module in Python that helps create **virtual environments**.
* second: The name of your virtual environment folder.

📁 This creates a folder named second/, which contains:

* A **copy of Python**
* Its own pip installer
* Scripts to activate the environment

🎯 So you now have a **clean room** just for this project, without messing up other Python projects on your system.

### ✅ STEP 4: Activate Virtual Environment

bash

.\second\Scripts\activate

🔄 If it works, you'll see your terminal change to:

scss

(second) C:\Users\YourName>

**💡 What’s happening?**

* You're **entering** the virtual environment.
* It changes your terminal so that any Python or pip command will run **inside the virtual space** (second) and not in the global/system Python.

🎯 Once activated:

* You'll see something like this in the terminal:  
  (second) C:\Users\YourName\ProjectFolder>
* Now you can safely install Django and other packages just for this project.

### ✅ STEP 5: Install Django

bash

pip install django

### ✅ STEP 6: Create a Django Project

bash

django-admin startproject program13

This creates:

markdown

program13/

├── manage.py

└── program13/

├── \_\_init\_\_.py

├── settings.py

├── urls.py

├── asgi.py

└── wsgi.py

### ✅ STEP 7: Go Into the Project Folder

bash

cd program13

Now you're in the same folder as manage.py

### ✅ STEP 8: Create a templates Folder

📁 Inside this program13/ folder (where manage.py is), create a folder named:

nginx

templates

Inside templates, create these 3 files:

* home.html
* dashboard.html
* contactUs.html

### ✅ STEP 9: Add HTML Code

📄 In each file, paste this:

**home.html**

html

<!DOCTYPE html>

<html>

<head>

<title>Home</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<h1 class="text-center">Welcome to CMRIT</h1>

<nav class="navbar navbar-expand-lg bg-body-tertiary">

<div class="container-fluid">

<a class="navbar-brand" href="#">CMRIT</a>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

<li><a class="nav-link" href="{% url 'home' %}">Home</a></li>

<li><a class="nav-link" href="{% url 'dashboard' %}">Dashboard</a></li>

<li><a class="nav-link" href="{% url 'contactUs' %}">Contact Us</a></li>

</ul>

</div>

</div>

</nav>

</body>

</html>

**dashboard.html**

html

<!DOCTYPE html>

<html>

<head><title>Dashboard</title></head>

<body>

<h1>Welcome to Dashboard</h1>

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

</body>

</html>

**contactUs.html**

html

<!DOCTYPE html>

<html>

<head><title>Contact Us</title></head>

<body>

<h1>CMR Institute of Technology, Medchal, Hyderabad</h1>

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

</body>

</html>

### ✅ STEP 10: Edit settings.py to Link Templates

📄 Open: program13/program13/settings.py

1. At the top, add:

python

import os

1. Find the TEMPLATES section and change:

python

'DIRS': [],

➡️ To:

python

'DIRS': [BASE\_DIR / 'program13' / 'templates'],

### ✅ STEP 11: Create index.py for Views

📄 Inside the **inner** program13/ folder (the one with urls.py), create a file called index.py  
Paste this:

python

from django.shortcuts import render

def home(request):

return render(request, 'home.html')

def dashboard(request):

return render(request, 'dashboard.html')

def contactUs(request):

return render(request, 'contactUs.html')

### ✅ STEP 12: Update urls.py

📄 Open: program13/program13/urls.py  
Replace everything with this:

python

from django.contrib import admin

from django.urls import path

from . import index

urlpatterns = [

path('', index.home, name='home'),

path('dashboard/', index.dashboard, name='dashboard'),

path('contactUs/', index.contactUs, name='contactUs'),

]

### ✅ STEP 13: Run the Server

📍 Make sure you're still in the folder where manage.py is (outer program13):

bash

py manage.py runserver

You’ll see:

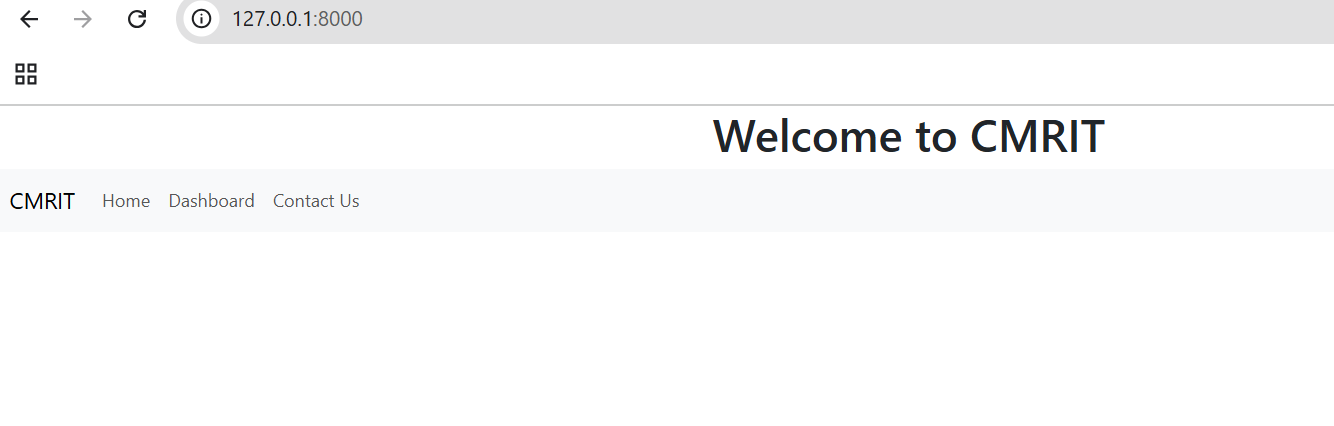
nginx

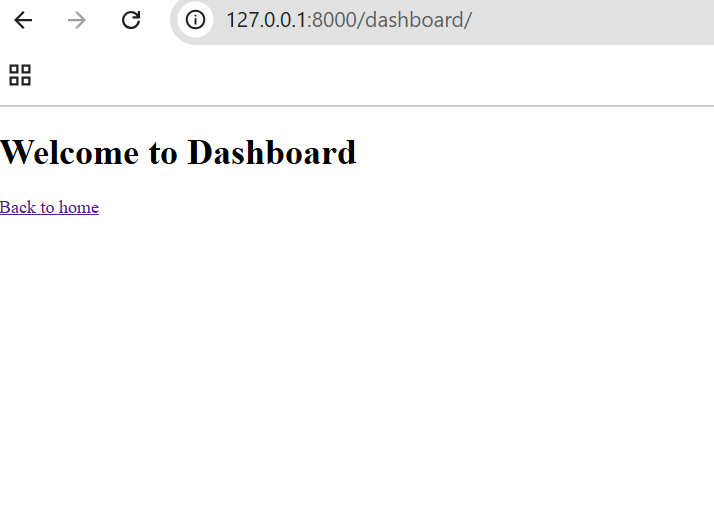
Starting development server at http://127.0.0.1:8000/

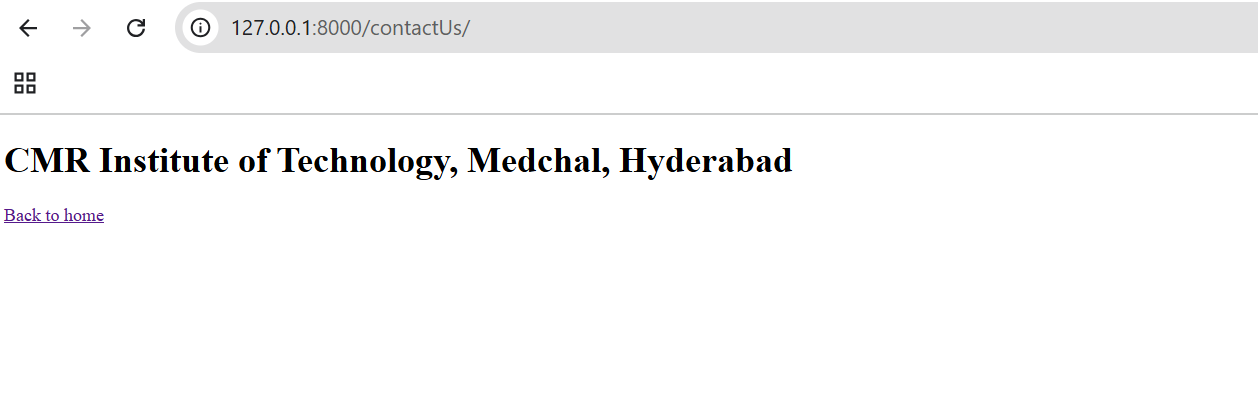
### ✅ STEP 14: Open in Browser

Visit these in your browser:

| **Page** | **URL** |
| --- | --- |
| Home | <http://127.0.0.1:8000/> |
| Dashboard | <http://127.0.0.1:8000/dashboard/> |
| Contact Us | <http://127.0.0.1:8000/contactUs/> |







## ✅ **WEEK 14 LAB ACTIVITY: Django CRUD Operations (Create, Read, Update, Delete)**

We'll build a simple **Student Management App** where you can:

* Add a new student
* View all students
* Edit a student's record
* Delete a student

### ✅ ****STEP 1: Create a Django Project****

**📍 Where**: Inside your activated virtual environment in Command Prompt or VS Code Terminal  
**📌 Command**:

bash

django-admin startproject crud\_project

cd crud\_project

**📌 Why**: This sets up the main project folder with settings, URLs, etc.

**📤 Output**:

* Folder crud\_project/ created
* Files: manage.py, and a subfolder also named crud\_project with settings.py, urls.py, etc.

### ✅ ****STEP 2: Start a Django App****

**📍 Where**: Inside the crud\_project directory  
**📌 Command**:

bash

python manage.py startapp studentapp

**📌 Why**: This creates an app called studentapp which will contain the models, views, templates, etc.

**📤 Output**:

* New folder studentapp/ created with models.py, views.py, urls.py (you will create this manually), etc.

### ✅ ****STEP 3: Add**** studentapp ****to Installed Apps****

**📍 Where**: In crud\_project/settings.py, under INSTALLED\_APPS

**🛠 Change this**:

python

INSTALLED\_APPS = [

...

'studentapp',

]

**📌 Why**: To tell Django to include this app when running the project.

### ✅ ****STEP 4: Create a Model****

**📍 Where**: In studentapp/models.py

**📌 Code**:

python

from django.db import models

class Student(models.Model):

name = models.CharField(max\_length=100)

email = models.EmailField()

address = models.TextField()

def \_\_str\_\_(self):

return self.name

**📌 Why**: This defines the structure of your database table.

### ✅ ****STEP 5: Register Model in Admin****

**📍 Where**: In studentapp/admin.py

**📌 Code**:

python

from django.contrib import admin

from .models import Student

admin.site.register(Student)

**📌 Why**: So you can manage student data from Django Admin.

### ✅ ****STEP 6: Apply Migrations****

**📍 Where**: In terminal  
**📌 Commands**:

bash

python manage.py makemigrations

python manage.py migrate

**📌 Why**: This creates and updates your database schema.

### ✅ ****STEP 7: Create Superuser****

**📍 Where**: In terminal  
**📌 Command**:

bash

python manage.py createsuperuser

Follow the prompts (username, password, etc.)

**📌 Why**: To access Django's admin panel to manage data.

### ✅ ****STEP 8: Set Up URLs****

#### In crud\_project/urls.py, include the app URLs:

python

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('studentapp.urls')),

]

#### Then create a new file: studentapp/urls.py

python

from django.urls import path

from . import views

urlpatterns = [

path('', views.student\_list, name='student\_list'),

path('add/', views.add\_student, name='add\_student'),

path('edit/<int:id>/', views.edit\_student, name='edit\_student'),

path('delete/<int:id>/', views.delete\_student, name='delete\_student'),

]

**📌 Why**: This routes user requests to correct views.

### ✅ ****STEP 9: Create Views****

**📍 Where**: In studentapp/views.py

python

from django.shortcuts import render, redirect

from .models import Student

def student\_list(request):

students = Student.objects.all()

return render(request, 'student\_list.html', {'students': students})

def add\_student(request):

if request.method == 'POST':

name = request.POST['name']

email = request.POST['email']

address = request.POST['address']

Student.objects.create(name=name, email=email, address=address)

return redirect('/')

return render(request, 'add\_student.html')

def edit\_student(request, id):

student = Student.objects.get(id=id)

if request.method == 'POST':

student.name = request.POST['name']

student.email = request.POST['email']

student.address = request.POST['address']

student.save()

return redirect('/')

return render(request, 'edit\_student.html', {'student': student})

def delete\_student(request, id):

student = Student.objects.get(id=id)

student.delete()

return redirect('/')

### ✅ ****STEP 10: Create Templates****

**📍 Where**: Create a templates folder inside studentapp/  
Make sure to configure it in settings.py:

python

import os

TEMPLATES = [

{

...

'DIRS': [os.path.join(BASE\_DIR, 'studentapp', 'templates')],

...

},

]

Create these HTML files inside studentapp/templates/:

#### 🔹 student\_list.html

html

<h2>Student List</h2>

<a href="{% url 'add\_student' %}">Add Student</a>

<ul>

{% for student in students %}

<li>

{{ student.name }} - {{ student.email }}

<a href="{% url 'edit\_student' student.id %}">Edit</a>

<a href="{% url 'delete\_student' student.id %}">Delete</a>

</li>

{% endfor %}

</ul>

#### 🔹 add\_student.html

html

<h2>Add Student</h2>

<form method="post">

{% csrf\_token %}

<input type="text" name="name" placeholder="Name"><br>

<input type="email" name="email" placeholder="Email"><br>

<textarea name="address" placeholder="Address"></textarea><br>

<button type="submit">Add</button>

</form>

#### 🔹 edit\_student.html

html

<h2>Edit Student</h2>

<form method="post">

{% csrf\_token %}

<input type="text" name="name" value="{{ student.name }}"><br>

<input type="email" name="email" value="{{ student.email }}"><br>

<textarea name="address">{{ student.address }}</textarea><br>

<button type="submit">Update</button>

</form>

### ✅ ****STEP 11: Run the Server****

**📍 Where**: Terminal  
**📌 Command**:

bash

python manage.py runserver

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

**📤 Output**: A working CRUD app for managing students!