**What is a Framework?**

A framework is a particular set of rules, ideas, or beliefs which you use in order to deal with problems or to decide what to do.

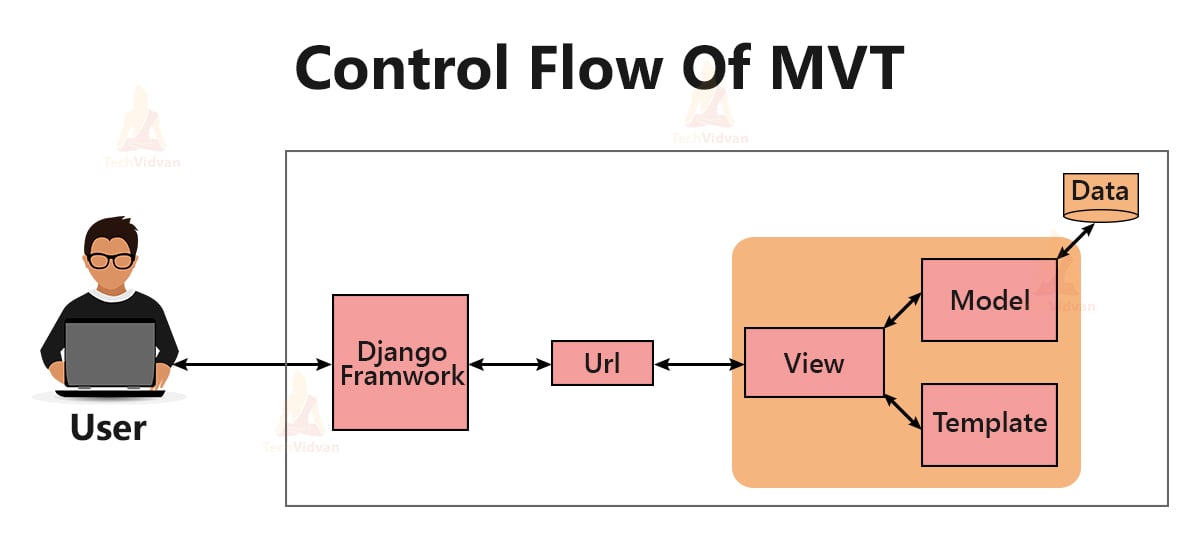
**What is Django?**

Django is a high-level Python web framework that helps you build web applications quickly and safely.

* Django is an open-source python framework. You can simply install it through **pip install django** command.
* Django follows the model-view-template architectural pattern.
* It gives you ORM (Object Relational Mapping – allows you to interact with a database {like SQLite, MySQL, or PostgreSQL} using Python objects -instead of writing raw SQL queries.), routing, templates (HTML), admin panel, forms, security features, and more out of the box.

Prerequisite – HTML, CSS, JavaScript, Database Management, Basic knowledge of Python.

**MVT architecture**



* **Model – The Data Layer:**

Defines the structure of your database tables. This is a class-based model where each model maps to a table, and each attribute maps to a column and instance of the class represents the rows of the table. Main functionality to handle data validation, database queries (creation, retrieval, update, deletion), and relationships between different data entities.

* **Templates**

Templates are typically HTML files that contain static content and also embed dynamic content taking from model also control the flow of presentation. It is known as presentation layer.

* **View**

Handles the application’s logic and functionality. It receives user requests, interacts with the Model to fetch or modify data, and prepares the data to be displayed.

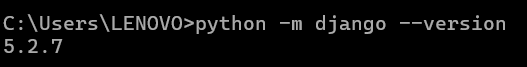
**Installation Process of Django (very simple)**

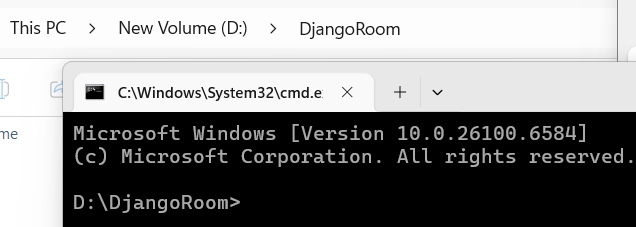
Commands (run in terminal) (Assuming everyone already has Python installed. If not, please kindly install the latest version.)

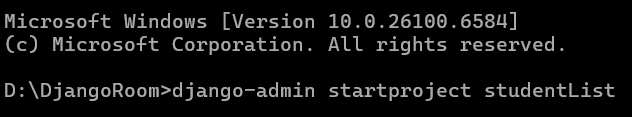
pip install django

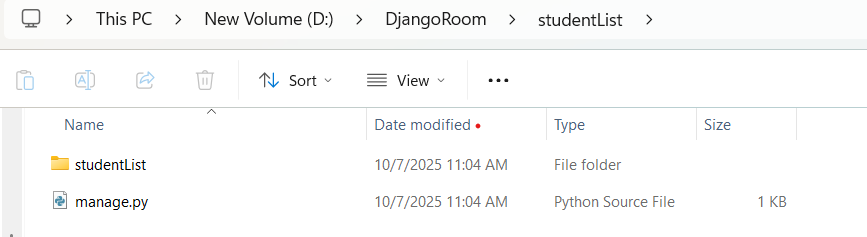
It will take some time.

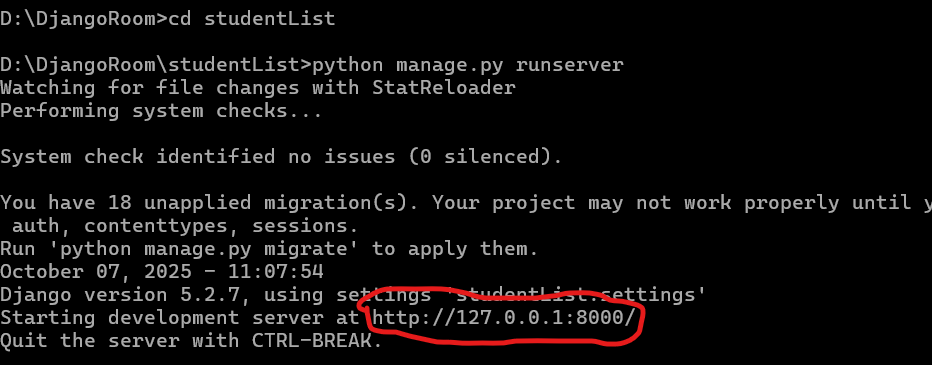
Then check version

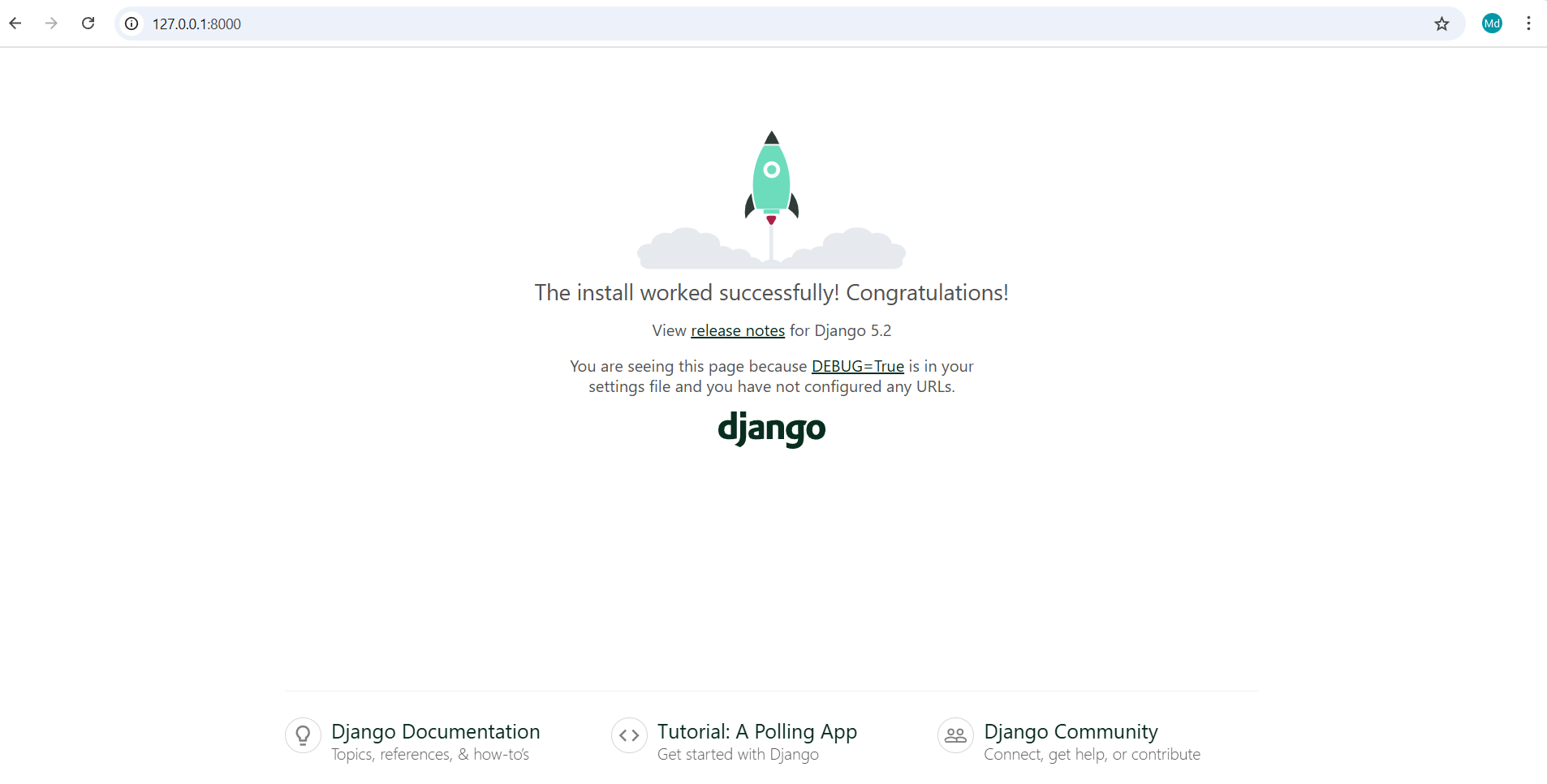


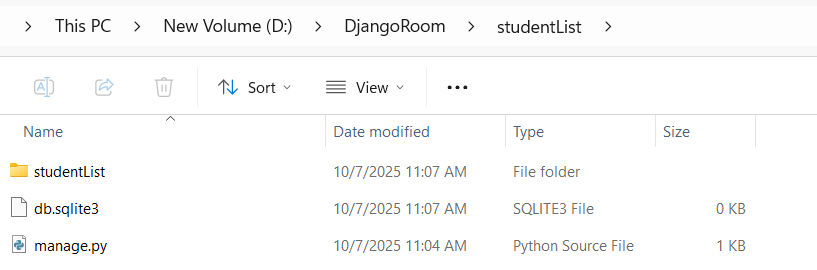
Now we will start our Django project inside a dedicated folder where we will keep all our Django projects. Open command prompt as follow: 

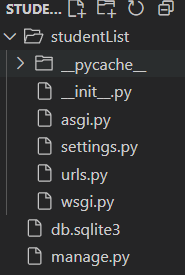


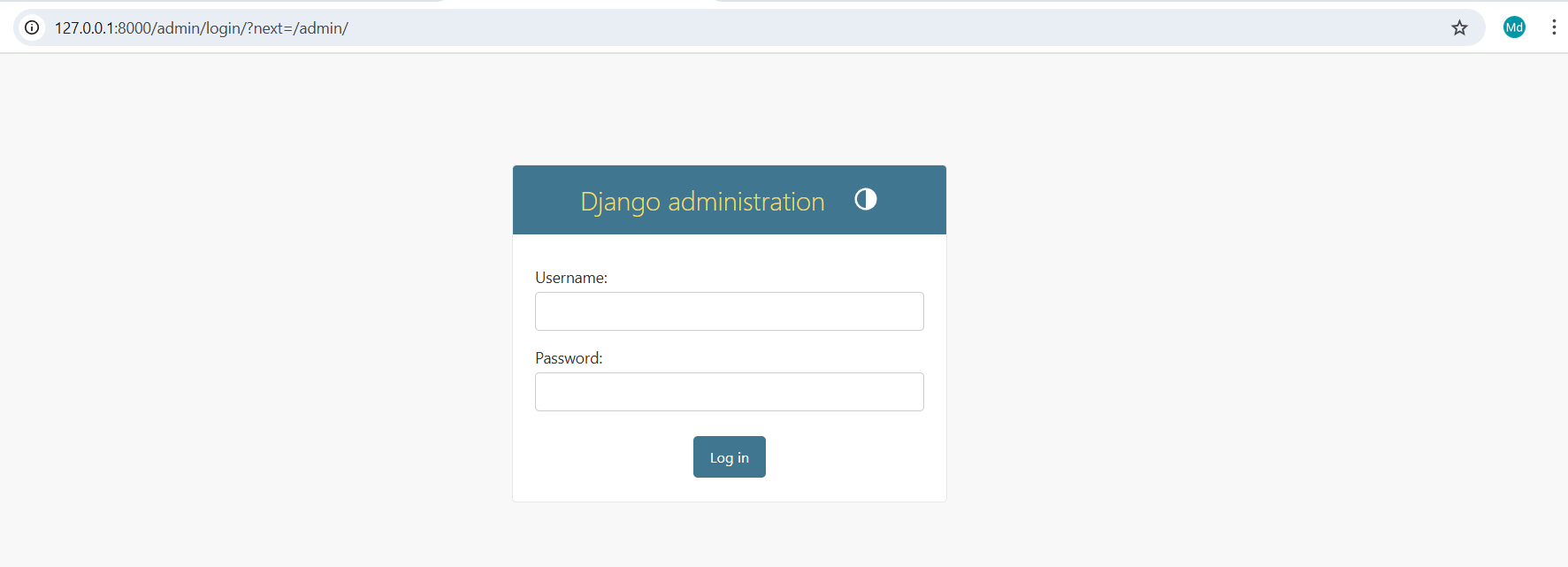


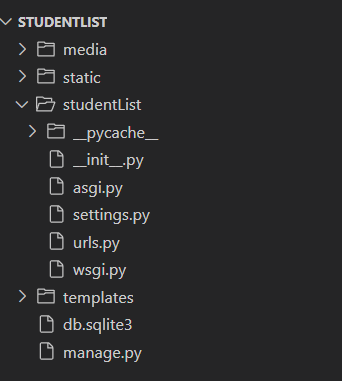












**What is manage.py?**

It is a command-line utility known as manager of your project by interacting with your Django project in various ways- like running the server, creating apps, migrating databases, managing users etc.

**What is settings.py?**

Contains all the configuration settings known as a control center of your Django project.

**What is settings.py?**

This file tells Django **how your project should work** — like a control center.

It stores:

* Secret keys.
* Database connection.
* Installed apps.
* Time zone.
* Where HTML, CSS, JS files are stored.
* **Import and Base Directory**

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

* from pathlib import Path→ helps Django find folders easily.
* BASE\_DIR → main folder of your project.  
  Example: if your project folder is mysite/, this line helps Django know where to find files like templates, static, etc.
* **Security and Debug**

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-2&qc!&#jij!(p)oxmrf=\*$40$k9@-q%&vc3m08v1!n!%pta^n#'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

* SECRET\_KEY: A secret code used by Django to protect your website. Keep it safe.
* DEBUG = True: Means Django will show detailed error messages — good for development.  
  In real websites (production), you’ll set DEBUG = False.
* ALLOWED\_HOSTS: When you make your site live, you’ll list your website name here.  
  Example: ALLOWED\_HOSTS = ['127.0.0.1', 'mycakeshop.com']
* **Installed Apps**

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

ALLOWED\_HOSTS = []

Think of this like a list of “features” Django will use.

* admin → admin dashboard
* auth → login/logout system
* sessions → store user sessions (login info)
* messages → system for success/error messages
* staticfiles → handle CSS, JS, and images
* **Middleware**

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

Middleware = “Middle helpers” between the user and your website.

They:

* Protect your site from attacks (like CSRF)
* Manage user logins
* Handle cookies and messages

You usually don’t change this part.

* **Root URL and WSGI**

ROOT\_URLCONF = 'mysite.urls'

WSGI\_APPLICATION = 'mysite.wsgi.application'

* ROOT\_URLCONF: Tells Django where your urls.py file is (it controls the links).
* WSGI\_APPLICATION: Needed for server deployment — you can ignore it for now.
* **Templates**

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

This section tells Django where to find your HTML files.

'DIRS': [] → You can tell Django where your template folder is.

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

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

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },},

]

Django will look inside each app’s templates folder automatically.

* **Database**

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

Right now it uses SQLite, a small database file — perfect for beginners. Later we’ll change it to **MySQL** using XAMPP. For now, keep this as it is.

Example MySQL (we’ll do this together later):

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'cake\_shop\_db',

'USER': 'root',

'PASSWORD': '',

'HOST': '127.0.0.1',

'PORT': '3306',

}

}

* **Password Validation**

AUTH\_PASSWORD\_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

    },

]

These rules make sure users choose strong passwords (not too short or too simple).

* **Language and Timezone**

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'Asia/Dhaka'

USE\_I18N = True

USE\_TZ = True

* **Static Files (CSS, JS, Images)**

STATIC\_URL = [**BASE\_DIR** / 'static']

Or

STATIC\_URL = 'static/'

STATICFILES\_DIRS = [BASE\_DIR / 'static']

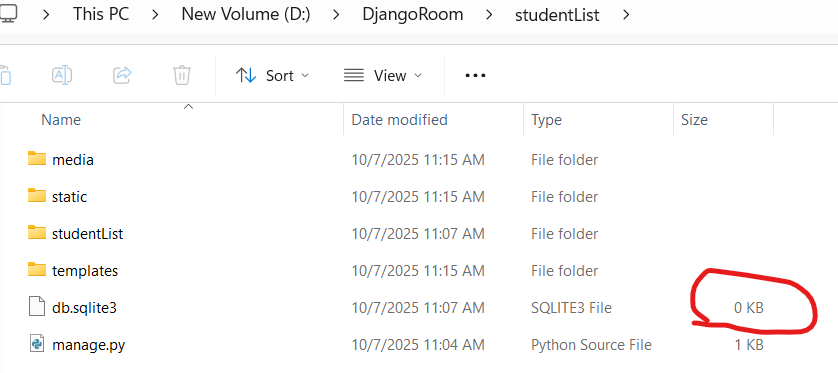
This tells Django that when you use {% static 'css/style.css' %} in HTML,  
it should look in the static folder**.**

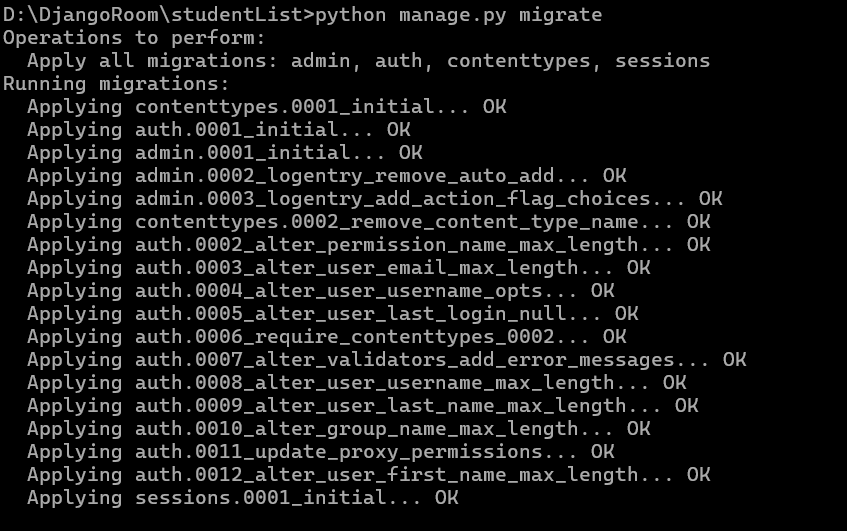
* **Default Auto Field**

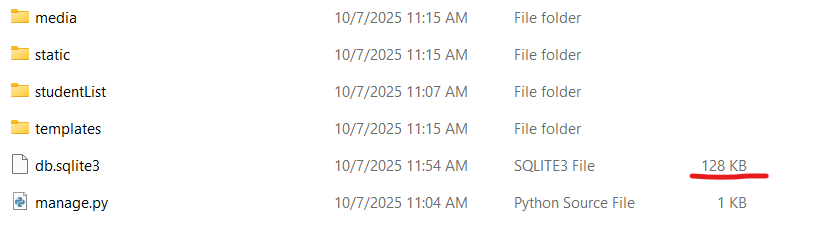
DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

This sets how Django creates ID numbers for your database tables.  
Leave it as it is.

* **How to Migrate Default Migrations**

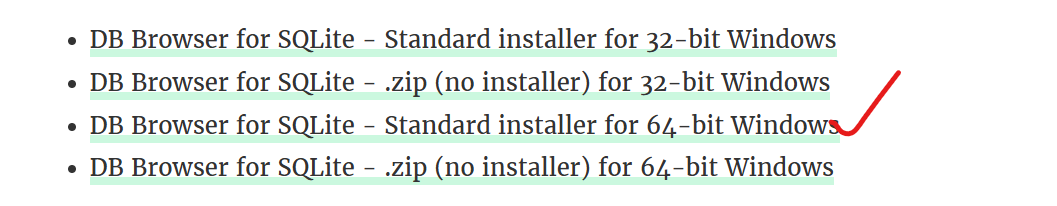
****



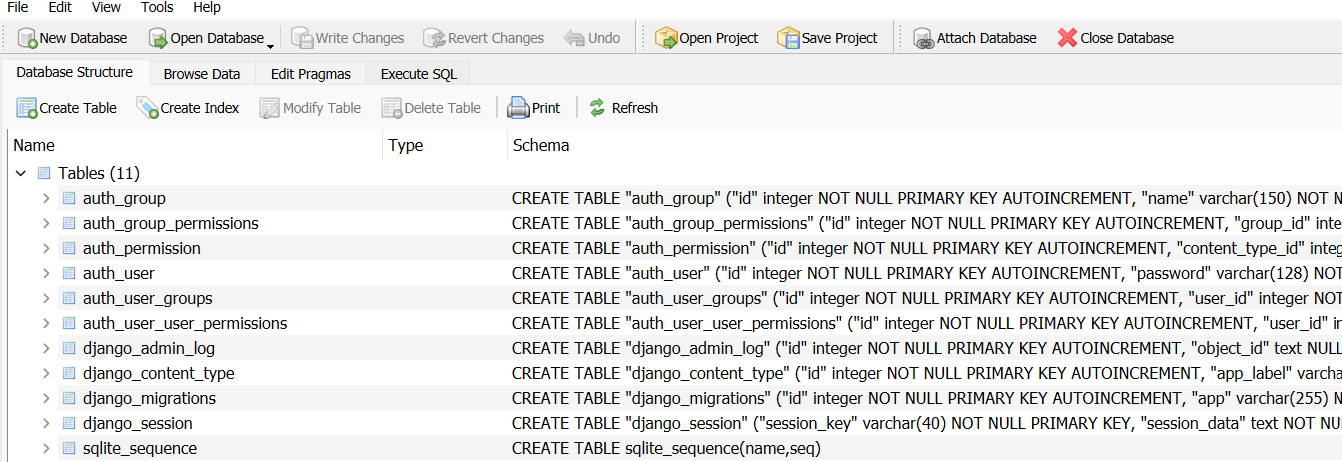


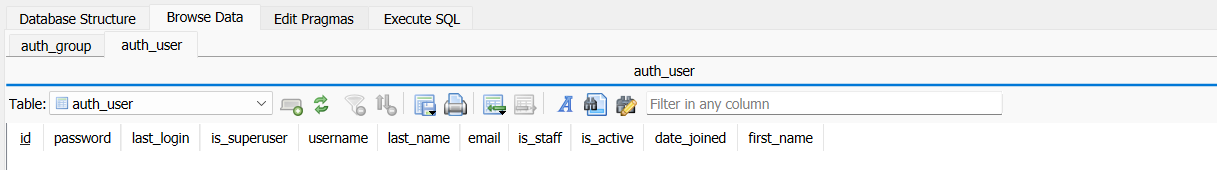
Install DB Browser for viewing SQLite3 tables. Install from following link.

<https://sqlitebrowser.org/dl/>

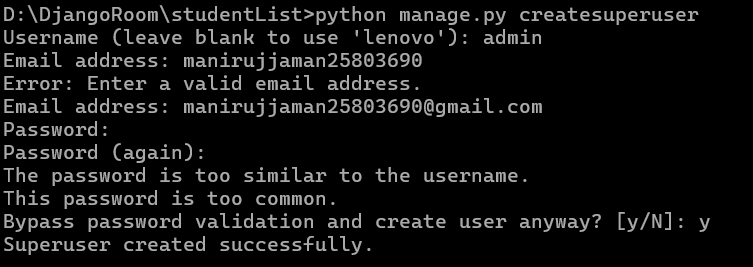


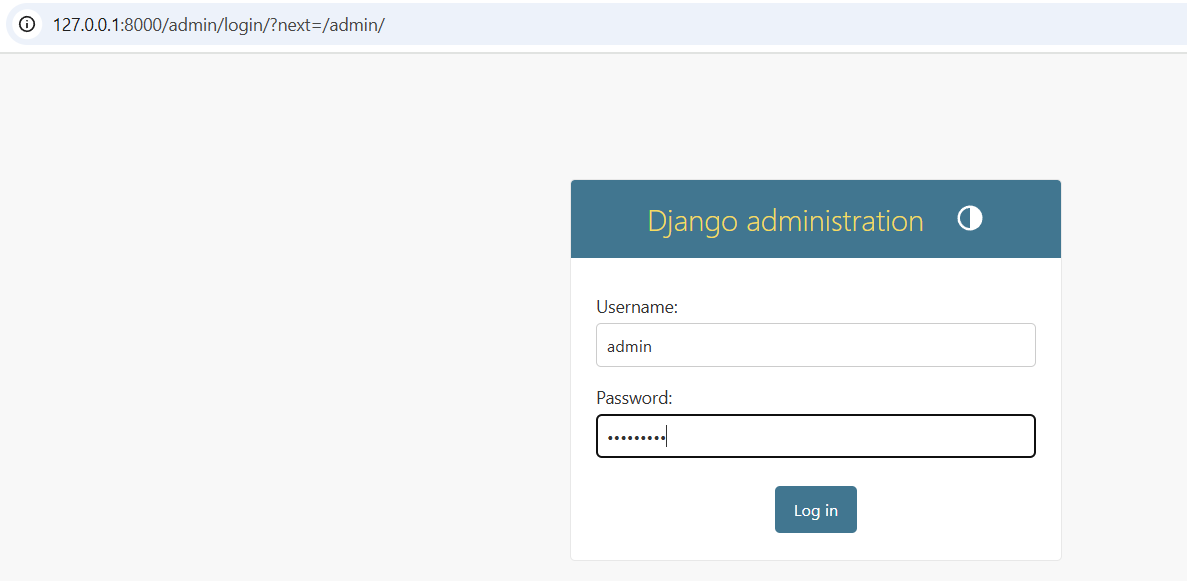
Then Open DB Browser and click on **Open Database** then select **db.sqlite3** from your project folder.

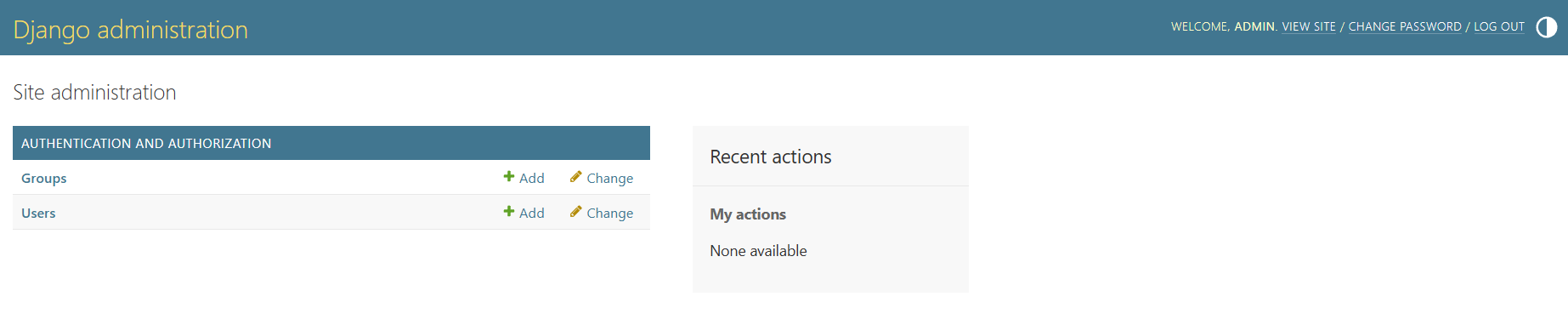




Now we will create super user to manage the admin panel.

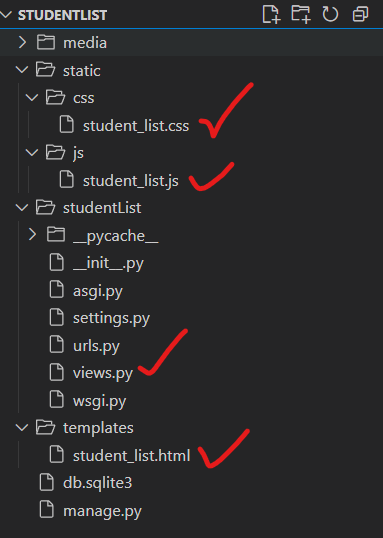






Let's dive into rendering! First, we'll render an HTML page along with its associated CSS and JavaScript static files. After that, we'll connect the page to a SQLite3 database within our Django project. For demonstration, we'll build a simple project called **Student Information**, where the page will display student details fetched directly from the database.

Set your project folder as follows:



**HTML File:** student\_list.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Student List</title>      {% load static %}      <link rel="stylesheet" href="{% static 'css/student\_list.css' %}">  </head>  <body>      <div class="container">          <h1>Student Information System</h1>          <table id="studentTable">              <thead>                  <tr>                      <th>Student ID</th>                      <th>Name</th>                      <th>Department</th>                      <th>Phone No</th>                      <th>Address</th>                  </tr>              </thead>              <tbody>                  <!-- Dummy data (will be replaced by DB data later) -->                  <tr>                      <td>STU001</td>                      <td>Rakib Hasan</td>                      <td>CSE</td>                      <td>01711111111</td>                      <td>Rajshahi</td>                  </tr>                  <tr>                      <td>STU002</td>                      <td>Nusrat Jahan</td>                      <td>EEE</td>                      <td>01822222222</td>                      <td>Dhaka</td>                  </tr>                  <tr>                      <td>STU003</td>                      <td>Sabbir Hossain</td>                      <td>BBA</td>                      <td>01933333333</td>                      <td>Rangpur</td>                  </tr>              </tbody>          </table>      </div>      <script src="{% static 'js/student\_list.js' %}"></script>  </body>  </html> |

**CSS: student\_list.css**

|  |
| --- |
| body {      font-family: Arial, sans-serif;      background-color: #f7f9fb;      margin: 0;      padding: 0;  }  .container {      width: 80%;      margin: 40px auto;      background-color: #ffffff;      padding: 20px 40px;      box-shadow: 0 0 10px rgba(0,0,0,0.1);      border-radius: 10px;  }  h1 {      text-align: center;      color: #333;      margin-bottom: 30px;  }  table {      width: 100%;      border-collapse: collapse;  }  th, td {      border: 1px solid #ddd;      text-align: center;      padding: 10px;  }  th {      background-color: #007bff;      color: white;      text-transform: uppercase;  }  tr:nth-child(even) {      background-color: #f2f2f2;  }  tr:hover {      background-color: #e8f0fe;      transition: 0.3s;  }  .selected {  background-color: #cce5ff !important;  } |

**JS: student\_list.js**

|  |
| --- |
| // Simple highlight effect on table rows  document.addEventListener("DOMContentLoaded", () => {      const rows = document.querySelectorAll("#studentTable tbody tr");      rows.forEach(row => {          row.addEventListener("click", () => {              rows.forEach(r => r.classList.remove("selected"));              row.classList.add("selected");              alert(`You selected: ${row.cells[1].innerText}`);          });      });  }); |

Before proceeding with rendering, paste the following code into your **views.py** file.

|  |
| --- |
| from django.http import HttpResponse #for displaying simple text response to my browser  def homePage(request):      return HttpResponse("<h1>This a simple Home page </h1>") |

Then set URL in **urls.py**

|  |
| --- |
| from django.contrib import admin  from django.urls import path  from . import views  urlpatterns = [      path('admin/', admin.site.urls),      path('', views.homePage, name='home'),  # Map the root URL to the homePage view  ] |

After running the server you will see following page output



Let’s render the student\_list.html page in a new URL through views.

So update your **views.py** as follows:

from django.http import HttpResponse #for displaying simple text response to my browser

from django.shortcuts import render

def homePage(request):

    return HttpResponse("<h1>This a simple Home page </h1>")

def student\_list(request):

    return render(request, 'student\_list.html')

and also update **urls.py** as follows:

from django.contrib import admin

from django.urls import path

from . import views

urlpatterns = [

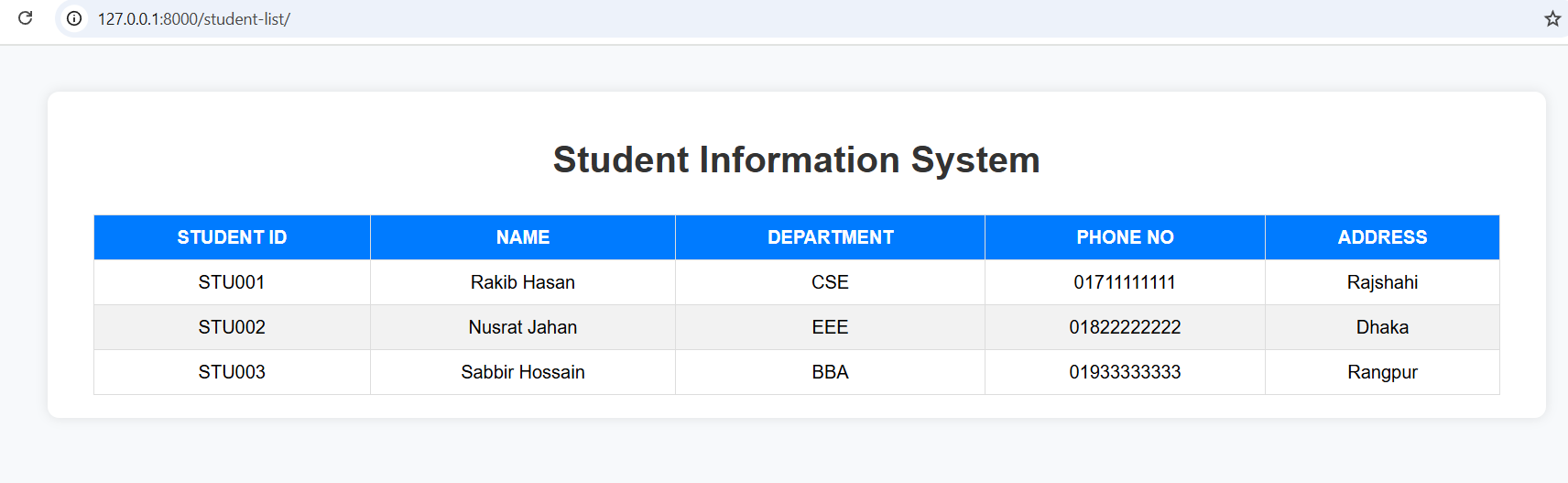
    path('admin/', admin.site.urls),

    path('', views.homePage, name='home'),  # Map the root URL to the homePage view

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

]

You will see the student List page as fellow:



Now for creating table and fetching data from database follow the following command.

**Create a Model (in studentList/models.py)**

Open models.py and paste following code

from django.db import models

class Student(models.Model):

    student\_id = models.CharField(max\_length=10, unique=True)

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

    department = models.CharField(max\_length=50)

    phone\_no = models.CharField(max\_length=15)

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

    def \_\_str\_\_(self):

        return self.name

**Step 2: Register the App**

Open studentlist/settings.py

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

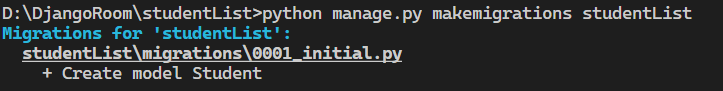
    'django.contrib.messages',

    'django.contrib.staticfiles',

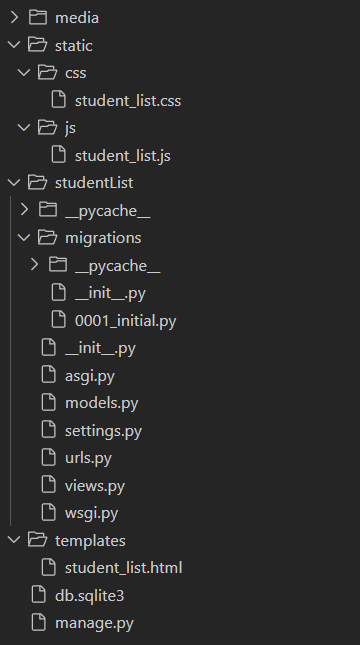
    'studentList',

]

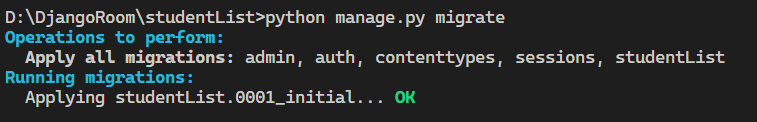
Then write given command in terminal cmd



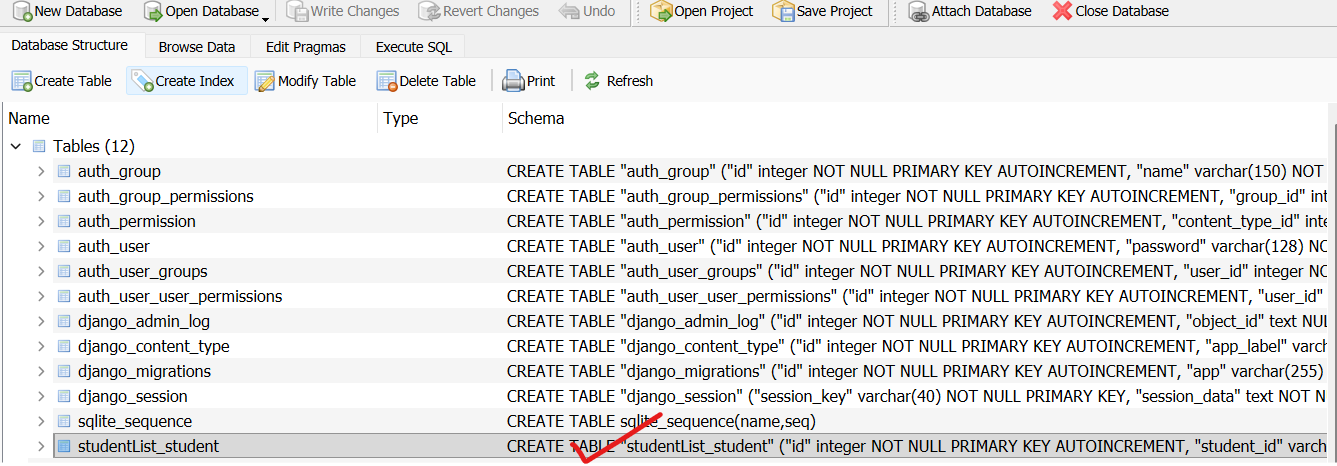
Your project folder will look like as following

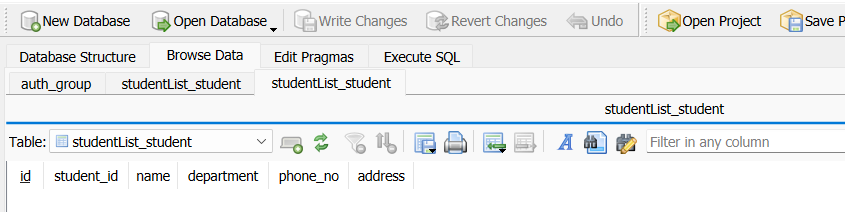


Now write following command:



studentList\_student table is created in you SQLite3 database.





Add Some Data (for testing)



Add following data:

|  |
| --- |
| from studentList.models import Student  Student.objects.create(student\_id='STU001', name='Rakib Hasan', department='CSE', phone\_no='01711111111', address='Rajshahi')  Student.objects.create(student\_id='STU002', name='Nusrat Jahan', department='EEE', phone\_no='01822222222', address='Dhaka')  Student.objects.create(student\_id='STU003', name='Sabbir Hossain', department='BBA', phone\_no='01933333333', address='Rangpur')  exit() |

|  |
| --- |
| from studentList.models import Student  Student.objects.create(student\_id=’STU004’, name=’Manirujjaman’, department=’CSE’, phone\_no = ‘01906446154’, address=’Rupgonj’) |

Now update views.py

from django.http import HttpResponse #for displaying simple text response to my browser

from django.shortcuts import render

from .models import Student

def homePage(request):

    return HttpResponse("<h1>This a simple Home page </h1>")

def student\_list(request):

    students=Student.objects.all()

    context={'students':students}

    return render(request, 'student\_list.html', context)

and also update your template **student\_list.html**

remove demo portion write following portion in <tbody> section.

  <tbody>

                  {% for student in students %}

                 <tr>

                    <td>{{ student.student\_id }}</td>

                    <td>{{ student.name }}</td>

                    <td>{{ student.department }}</td>

                    <td>{{ student.phone\_no }}</td>

                    <td>{{ student.address }}</td>

                 </tr>

                {% empty %}

                  <tr>

                     <td colspan="5">No students found.</td>

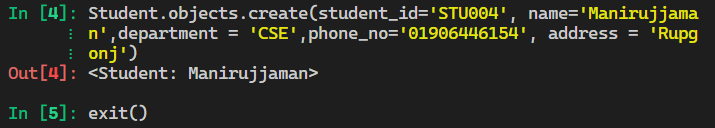
                  </tr>

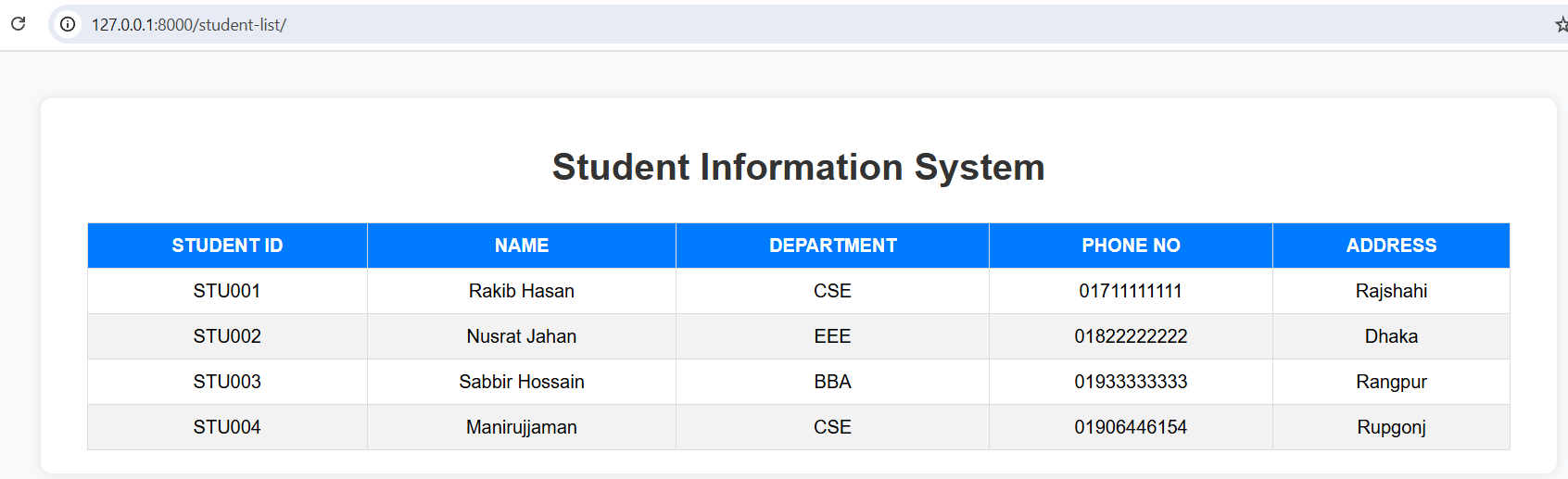
                {% endfor %}

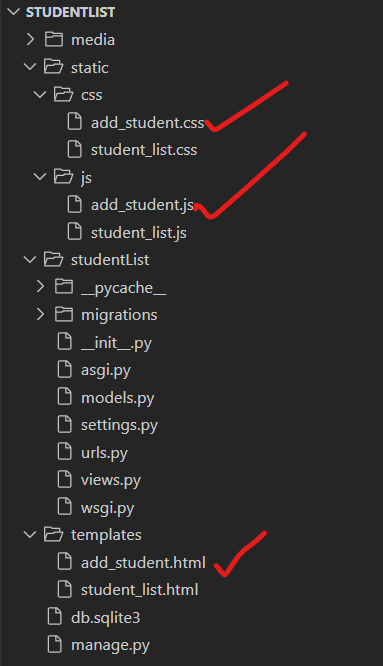
            </tbody>

Now runserver again.

Add and entry to the database.





Now we’re going to make a new template called **add\_student.html** — this one will help us insert data into the database and show it nicely on the **student\_list.html** page.   
First, let’s just render a demo HTML page (no pressure!). Then we’ll slowly connect it and pass the data to the database — step by step, like a chill coder! 

**HTML (add\_student.html)**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Add Student</title>      {% load static %}      <link rel="stylesheet" href="{% static 'css/add\_student.css' %}">  </head>  <body>      <div class="container">          <h1>Add Student Information</h1>          <form id="studentForm" method="POST">              {% csrf\_token %}              <div class="form-group">                  <label for="student\_id">Student ID</label>                  <input type="text" id="student\_id" name="student\_id" required>              </div>              <div class="form-group">                  <label for="name">Name</label>                  <input type="text" id="name" name="name" required>              </div>              <div class="form-group">                  <label for="department">Department</label>                  <input type="text" id="department" name="department" required>              </div>              <div class="form-group">                  <label for="phone\_no">Phone Number</label>                  <input type="text" id="phone\_no" name="phone\_no" required>              </div>              <div class="form-group">                  <label for="address">Address</label>                  <input type="text" id="address" name="address" required>              </div>              <button type="submit" class="btn">Add Student</button>          </form>          <p id="message"></p>          <a href="{% url 'student\_list' %}" class="back-link">← Back to Student List</a>      </div>      <script src="{% static 'js/add\_student.js' %}"></script>  </body>  </html> |

**Corresponding CSS. (add\_student.css)**

|  |
| --- |
| body {      font-family: Arial, sans-serif;      background-color: #f4f7fc;      display: flex;      justify-content: center;      align-items: center;      height: 100vh;      margin: 0;  }  .container {      background: white;      padding: 25px 35px;      border-radius: 12px;      box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);      width: 400px;  }  h1 {      text-align: center;      color: #333;      margin-bottom: 20px;  }  .form-group {      margin-bottom: 15px;  }  label {      display: block;      font-weight: bold;      margin-bottom: 5px;      color: #555;  }  input {      width: 100%;      padding: 8px 10px;      border: 1px solid #ccc;      border-radius: 6px;      font-size: 14px;  }  .btn {      background-color: #4a90e2;      color: white;      border: none;      padding: 10px 15px;      border-radius: 6px;      cursor: pointer;      width: 100%;      font-size: 16px;  }  .btn:hover {      background-color: #357abd;  }  .back-link {      display: block;      text-align: center;      margin-top: 15px;      text-decoration: none;      color: #4a90e2;  }  .back-link:hover {      text-decoration: underline;  }  #message {      text-align: center;      color: green;      margin-top: 10px;  } |

**Javascript (add\_student.js)**

|  |
| --- |
| document.getElementById("studentForm").addEventListener("submit", function(event) {      event.preventDefault();      // Get form values      const id = document.getElementById("student\_id").value;      const name = document.getElementById("name").value;      const dept = document.getElementById("department").value;      const phone = document.getElementById("phone\_no").value;      const address = document.getElementById("address").value;      // Simple validation      if (!id || !name || !dept || !phone || !address) {          alert("Please fill in all fields!");          return;      }      document.getElementById("message").textContent = "Student added successfully (frontend only for now)";      this.reset();  }); |

Again update **views.py** as fellow:

from django.http import HttpResponse #for displaying simple text response to my browser

from django.shortcuts import render

from .models import Student

def homePage(request):

    return HttpResponse("<h1>This a simple Home page </h1>")

def student\_list(request):

    students=Student.objects.all()

    context={'students':students}

    return render(request, 'student\_list.html', context)

def add\_student(request):

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

Again update **urls.py** as fellow:

from django.contrib import admin

from django.urls import path

from . import views

urlpatterns = [

    path('admin/', admin.site.urls),

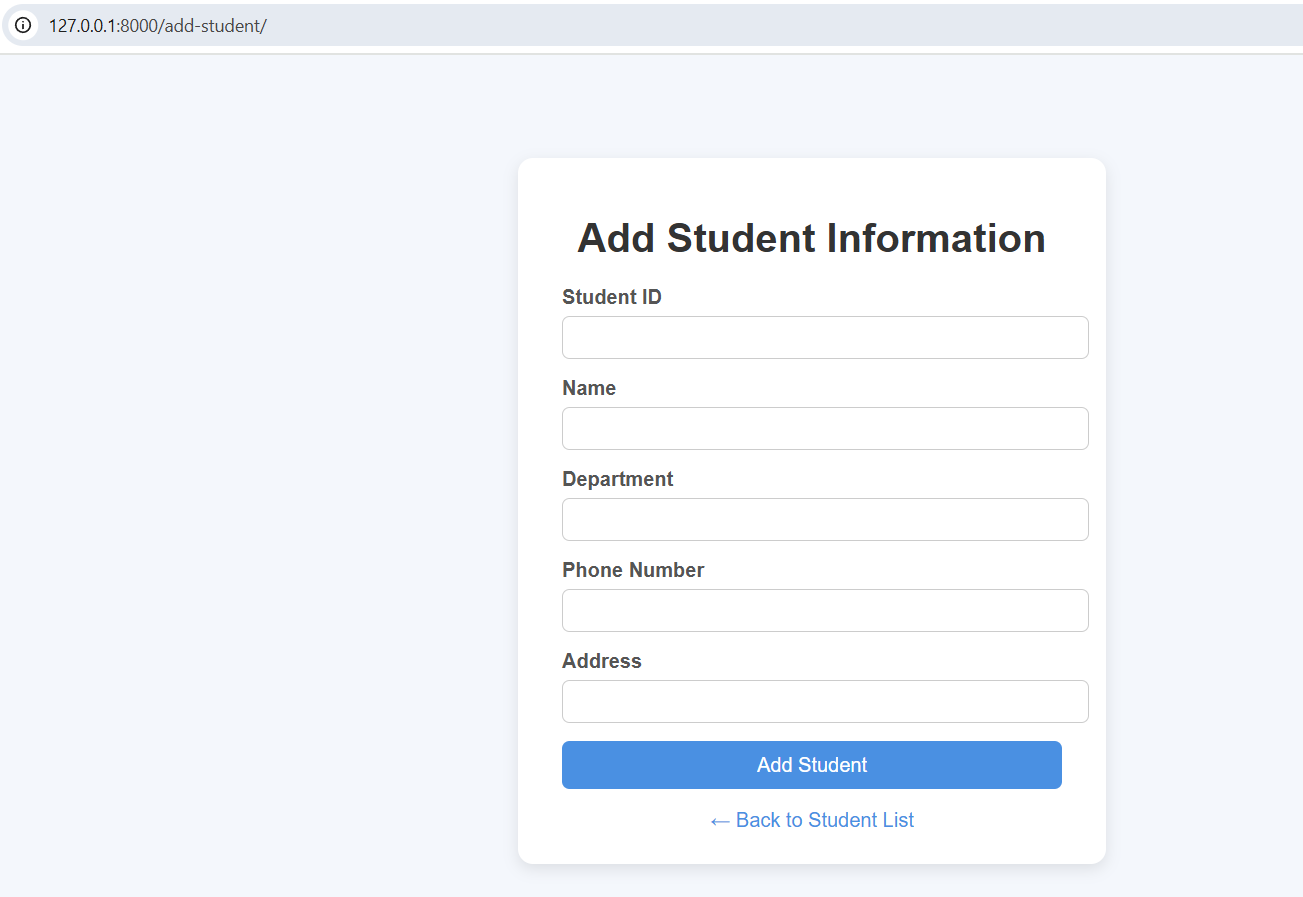
    path('', views.homePage, name='home'),  # Map the root URL to the homePage view

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

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

]

You will get output of page as like follows:



Now main task - passing data from the form to Student database.

Updated **views.py.**

from django.http import HttpResponse #for displaying simple text response to my browser

from django.shortcuts import render,redirect

from .models import Student

from django.contrib import messages

from django.db import IntegrityError

def homePage(request):

    return HttpResponse("<h1>This a simple Home page </h1>")

def student\_list(request):

    students=Student.objects.all()

    context={'students':students}

    return render(request, 'student\_list.html', context)

def add\_student(request):

    if request.method == 'POST':

        student\_id = request.POST.get('student\_id')

        name = request.POST.get('name')

        department = request.POST.get('department')

        phone\_no = request.POST.get('phone\_no')

        address = request.POST.get('address')

        try:

            Student.objects.create(

                student\_id=student\_id,

                name=name,

                department=department,

                phone\_no=phone\_no,

                address=address

            )

            #messages.success(request, '✅ Student added successfully!')

            return redirect('student\_list')

        except IntegrityError:

            messages.error(request, f'⚠️ Student ID "{student\_id}" already exists. Please use a unique ID.')

            return redirect('add\_student')

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

**Updated add\_student.html**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Add Student</title>      {% load static %}      <link rel="stylesheet" href="{% static 'css/add\_student.css' %}">  </head>  <body>      <div class="container">          <h1>Add Student Information</h1>          <!-- ✅ Form for adding student -->          <form method="POST" id="addStudentForm">              {% csrf\_token %}              <div class="form-group">                  <label for="student\_id">Student ID:</label>                  <input type="text" id="student\_id" name="student\_id" required>              </div>              <div class="form-group">                  <label for="name">Full Name:</label>                  <input type="text" id="name" name="name" required>              </div>              <div class="form-group">                  <label for="department">Department:</label>                  <input type="text" id="department" name="department" required>              </div>              <div class="form-group">                  <label for="phone\_no">Phone No:</label>                  <input type="text" id="phone\_no" name="phone\_no" required>              </div>              <div class="form-group">                  <label for="address">Address:</label>                  <input type="text" id="address" name="address" required>              </div>              <button type="submit">Add Student</button>          </form>          <a href="{% url 'student\_list' %}" class="back-link">← Back to Student List</a>      </div>      <script src="{% static 'js/add\_student.js' %}"></script>  </body>  </html> |

**Updated CSS.(add\_student.css)**

body {

    font-family: Arial, sans-serif;

    background-color: #f3f6fa;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

.container {

    background-color: #fff;

    padding: 25px;

    border-radius: 12px;

    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

    width: 380px;

    text-align: center;

}

h1 {

    color: #333;

    margin-bottom: 20px;

}

.form-group {

    margin-bottom: 15px;

    text-align: left;

}

label {

    display: block;

    margin-bottom: 5px;

    font-weight: bold;

    color: #555;

}

input {

    width: 100%;

    padding: 8px;

    border: 1px solid #bbb;

    border-radius: 6px;

    outline: none;

}

input:focus {

    border-color: #4CAF50;

}

button {

    background-color: #4CAF50;

    color: white;

    border: none;

    padding: 10px 15px;

    border-radius: 8px;

    cursor: pointer;

    font-size: 16px;

}

button:hover {

    background-color: #45a049;

}

.back-link {

    display: inline-block;

    margin-top: 15px;

    color: #333;

    text-decoration: none;

}

.back-link:hover {

    text-decoration: underline;

}

.messages {

    margin-bottom: 15px;

}

.message {

    background-color: #e7f7e7;

    color: #2e7d32;

    border: 1px solid #b2d8b2;

    border-radius: 5px;

    padding: 10px;

}

**Updated: js. (add\_student.js)**

document.getElementById('addStudentForm').addEventListener('submit', function(event) {

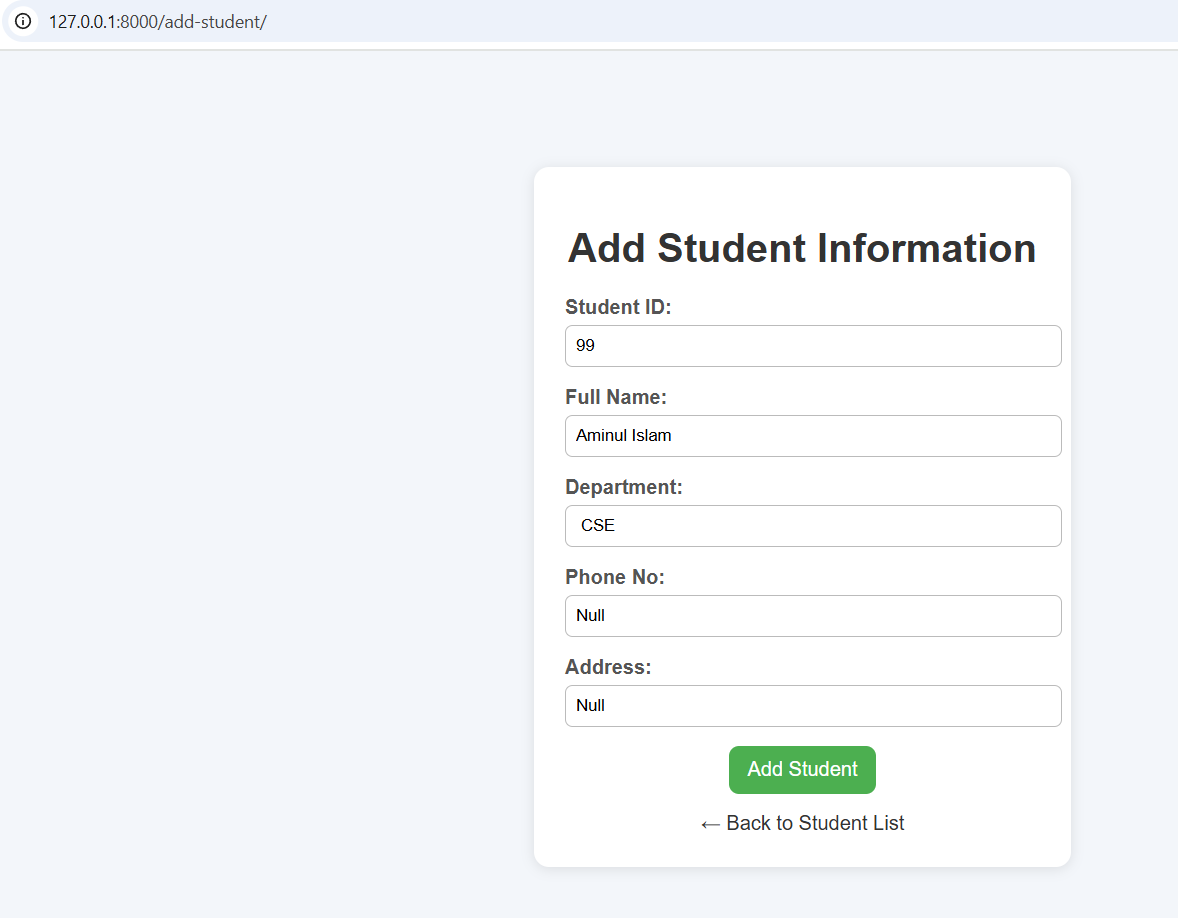
    const confirmSubmit = confirm("Are you sure you want to add this student?");

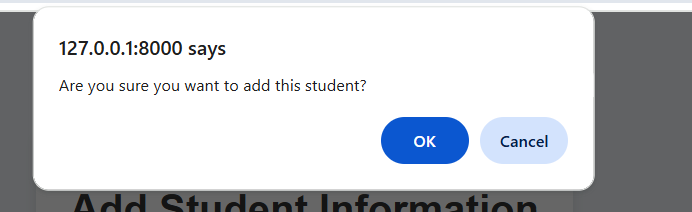
    if (!confirmSubmit) {

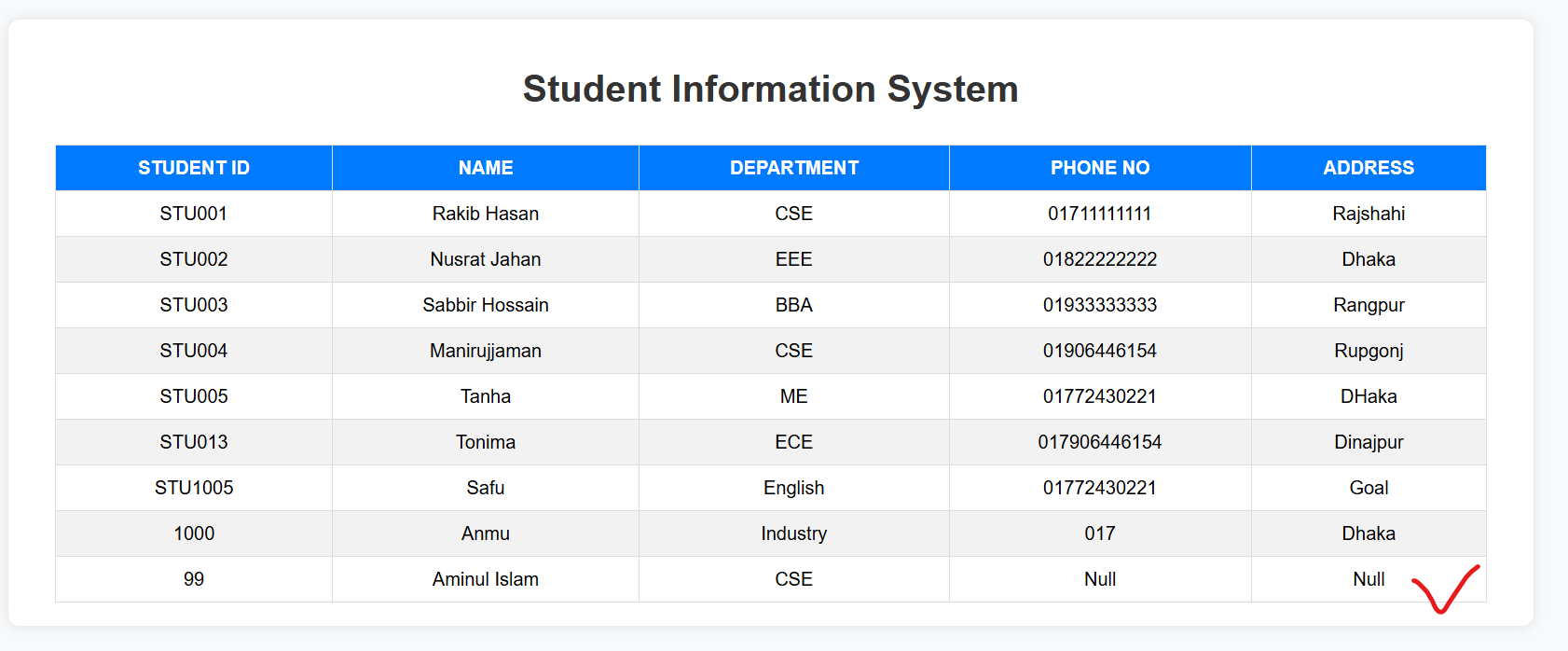
        event.preventDefault();

    }

});







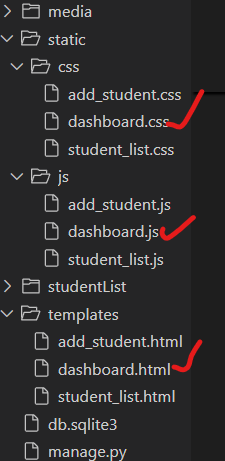
That’s All for Today

Now we’ll start with a **Home Dashboard page** showing demo data first (later we’ll connect it to our database).

This page will include:

* A **navbar** with your project name (e.g., *Student Management System we will change it later with actual navbar that will be shown to every pages. We will also footer to our website that will also be visible to every page*).
* Three **info cards** showing:
  + Total Students
  + Departments Count
  + Recently Added Students
* A simple **JavaScript animation** to make the numbers count up smoothly.
* Clean **CSS layout** using grid/flexbox.

**Now change the file structures as below:**



**Now dashboard.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Student Dashboard</title>

    {% load static %}

    <link rel="stylesheet" href="{% static 'css/dashboard.css' %}">

</head>

<body>

    <nav class="navbar">

        <h1>🎓 Student Management Dashboard</h1>

    </nav>

    <div class="container">

        <div class="card">

            <h2>Total Students</h2>

            <p class="count" data-target="150">0</p>

        </div>

        <div class="card">

            <h2>Departments</h2>

            <p class="count" data-target="5">0</p>

        </div>

        <div class="card">

            <h2>Recently Added</h2>

            <p class="count" data-target="3">0</p>

        </div>

    </div>

    <div class="recent">

        <h2>Recent Students</h2>

        <table>

            <tr>

                <th>ID</th>

                <th>Name</th>

                <th>Department</th>

                <th>Phone</th>

            </tr>

            <tr>

                <td>ST001</td>

                <td>Rafiul Islam</td>

                <td>CSE</td>

                <td>017xxxxxxxx</td>

            </tr>

            <tr>

                <td>ST002</td>

                <td>Meem Akter</td>

                <td>EEE</td>

                <td>018xxxxxxxx</td>

            </tr>

            <tr>

                <td>ST003</td>

                <td>Sabbir Rahman</td>

                <td>BBA</td>

                <td>019xxxxxxxx</td>

            </tr>

        </table>

    </div>

    <script src="{% static 'js/dashboard.js' %}"></script>

</body>

</html>

**Similarly copy CSS from dashboard.css**

body {

    font-family: 'Poppins', sans-serif;

    background-color: #f8f9fb;

    margin: 0;

    padding: 0;

}

.navbar {

    background-color: #4a69bd;

    color: white;

    text-align: center;

    padding: 20px;

    font-size: 1.5rem;

}

.container {

    display: flex;

    justify-content: center;

    flex-wrap: wrap;

    gap: 20px;

    margin: 30px auto;

    width: 80%;

}

.card {

    background-color: white;

    border-radius: 10px;

    box-shadow: 0 4px 8px rgba(0,0,0,0.1);

    width: 250px;

    padding: 20px;

    text-align: center;

    transition: transform 0.2s ease;

}

.card:hover {

    transform: scale(1.05);

}

.card h2 {

    color: #4a69bd;

    margin-bottom: 10px;

}

.count {

    font-size: 2.5rem;

    color: #222f3e;

    font-weight: bold;

}

.recent {

    width: 80%;

    margin: 40px auto;

}

.recent h2 {

    text-align: center;

    color: #333;

    margin-bottom: 15px;

}

table {

    width: 100%;

    border-collapse: collapse;

    background: white;

    box-shadow: 0 3px 6px rgba(0,0,0,0.1);

}

th, td {

    padding: 12px;

    border-bottom: 1px solid #ddd;

    text-align: center;

}

th {

    background-color: #4a69bd;

    color: white;

}

**Copy to dashboard.js**

// Animate number count-up

document.addEventListener("DOMContentLoaded", () => {

    const counters = document.querySelectorAll(".count");

    const speed = 200; // smaller is faster

    counters.forEach(counter => {

        const updateCount = () => {

            const target = +counter.getAttribute("data-target");

            const count = +counter.innerText;

            const increment = target / speed;

            if (count < target) {

                counter.innerText = Math.ceil(count + increment);

                setTimeout(updateCount, 20);

            } else {

                counter.innerText = target;

            }

        };

        updateCount();

    });

});

**Add following function to views.py**

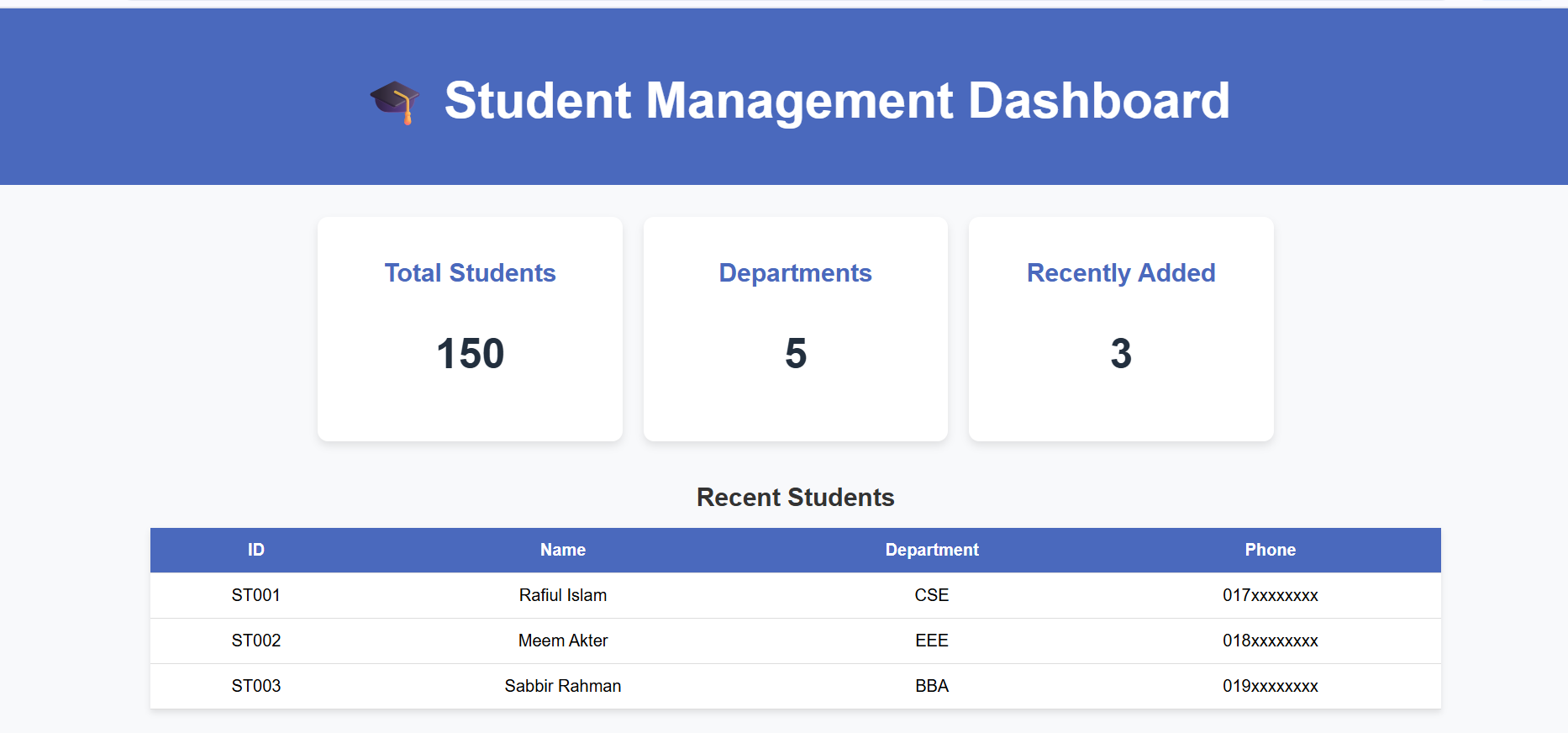
def dashboard(request):

    return render(request, 'dashboard.html')

**Now add the path to urls.py**

path('', views.dashboard, name='dashboard'),  # Map the root URL to the homePage view

**Now Run the Server you will get following page as home page with full of dummy data.**



Now let’s make our **Dashboard** page dynamic — meaning, all the counts and recent student data will automatically come from your database instead of demo values.

Now update the **dashboard()** function in **views.py** as below**:**

def dashboard(request):

    # Fetch all students

    students = Student.objects.all()

    # Total count

    total\_students = students.count()

    # Unique department count

    departments = students.values\_list('department', flat=True).distinct()

    department\_count = len(departments)

    # Get last 5 recently added students

    recent\_students = students.order\_by('-id')[:5]

    context = {

        'total\_students': total\_students,

        'department\_count': department\_count,

        'recent\_students': recent\_students,

    }

    return render(request, 'dashboard.html', context)

Remove the dummy data from the dashboard.html file and rewrite the html file as below:

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Student Dashboard</title>

    {% load static %}

    <link rel="stylesheet" href="{% static 'css/dashboard.css' %}" />

  </head>

  <body>

    <nav class="navbar">

      <h1>🎓 Student Management Dashboard</h1>

    </nav>

    <div class="container">

      <div class="card">

        <h2>Total Students</h2>

        <p class="count" data-target="{{ total\_students }}">

          {{ total\_students }}

        </p>

      </div>

      <div class="card">

        <h2>Departments</h2>

        <p class="count" data-target="{{ department\_count }}">

          {{ department\_count }}

        </p>

      </div>

      <div class="card">

        <h2>Recently Added</h2>

        <p class="count" data-target="{{ recent\_students.count }}">

          {{ recent\_students|length }}

        </p>

      </div>

    </div>

    <div class="recent">

      <h2>Recent Students</h2>

      <table>

        <tr>

          <th>ID</th>

          <th>Name</th>

          <th>Department</th>

          <th>Phone</th>

        </tr>

        {% for student in recent\_students %}

        <tr>

          <td>{{ student.student\_id }}</td>

          <td>{{ student.name }}</td>

          <td>{{ student.department }}</td>

          <td>{{ student.phone\_no }}</td>

        </tr>

        {% empty %}

        <tr>

          <td colspan="4">No students found.</td>

        </tr>

        {% endfor %}

      </table>

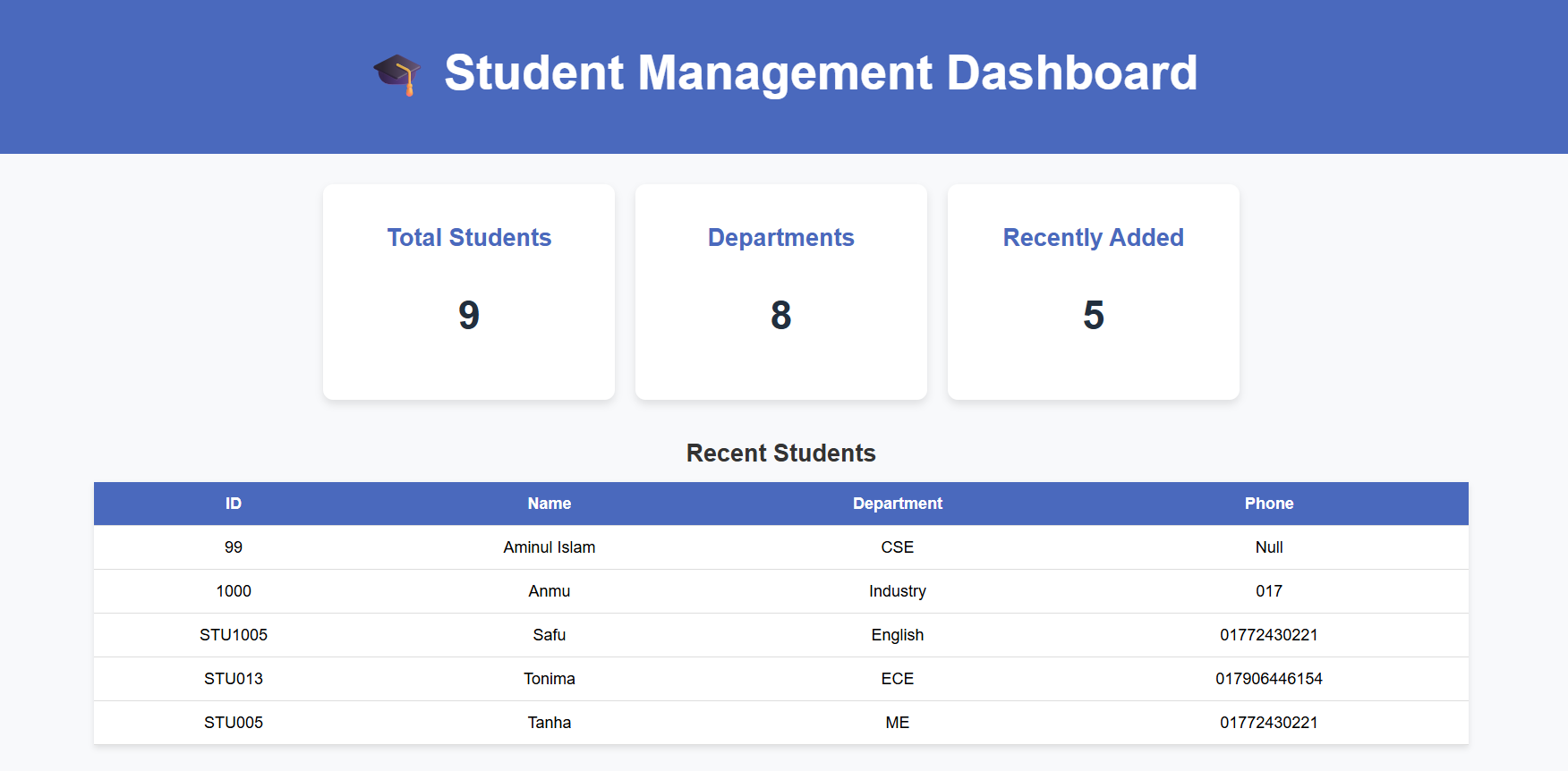
    </div>

    <script src="{% static 'js/dashboard.js' %}"></script>

  </body>

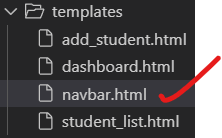
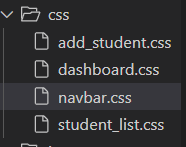
</html>

Reload you project you will get live data from ta database.



Now it’s time to modify our navbar. We will write the complete navbar code in a separate **navbar.html** file and include it in our **dashboard.html** page using Django’s Template Inheritance system ({% include %} or {% extends %}). This approach allows us to reuse the same navbar across multiple HTML pages.

**Now create a separate navbar file**

templates/navbar.html

{% load static %}

<!-- ✅ Navbar Section Only -->

<link rel="stylesheet" href="{% static 'css/navbar.css' %}">

<nav class="navbar">

    <div class="nav-container">

        <h1 class="logo">Student Management</h1>

        <ul class="nav-links">

            <li><a href="{% url 'dashboard' %}">Dashboard</a></li>

            <li><a href="{% url 'student\_list' %}">Student List</a></li>

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

            <li><a href="#">Delete Student</a></li>

            <li><a href="#">Modify Student</a></li>

        </ul>

    </div>

</nav>

**In static/css/navbar.css:**

/\* ====== Navbar Styling ====== \*/

.navbar {

    background-color: #1e3a8a; /\* deep blue \*/

    padding: 15px 40px;

    color: white;

    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.15);

    position: sticky;

    top: 0;

    z-index: 1000;

}

.nav-container {

    display: flex;

    justify-content: space-between;

    align-items: center;

    flex-wrap: wrap;

}

.logo {

    font-size: 22px;

    font-weight: bold;

    display: flex;

    align-items: center;

    gap: 10px;

    letter-spacing: 0.5px;

}

.nav-links {

    list-style: none;

    display: flex;

    align-items: center;

    gap: 30px;

    margin: 0;

    padding: 0;

}

.nav-links li a {

    color: white;

    text-decoration: none;

    font-weight: 500;

    font-size: 16px;

    display: flex;

    align-items: center;

    gap: 6px;

    transition: color 0.3s ease, transform 0.2s ease;

}

.nav-links li a:hover {

    color: #ffda79;

    transform: translateY(-2px);

}

/\* ====== Responsive Design ====== \*/

@media (max-width: 768px) {

    .nav-container {

        flex-direction: column;

        align-items: flex-start;

    }

    .nav-links {

        flex-direction: column;

        width: 100%;

        gap: 15px;

        margin-top: 10px;

        border-top: 1px solid rgba(255, 255, 255, 0.2);

        padding-top: 10px;

    }

    .nav-links li a {

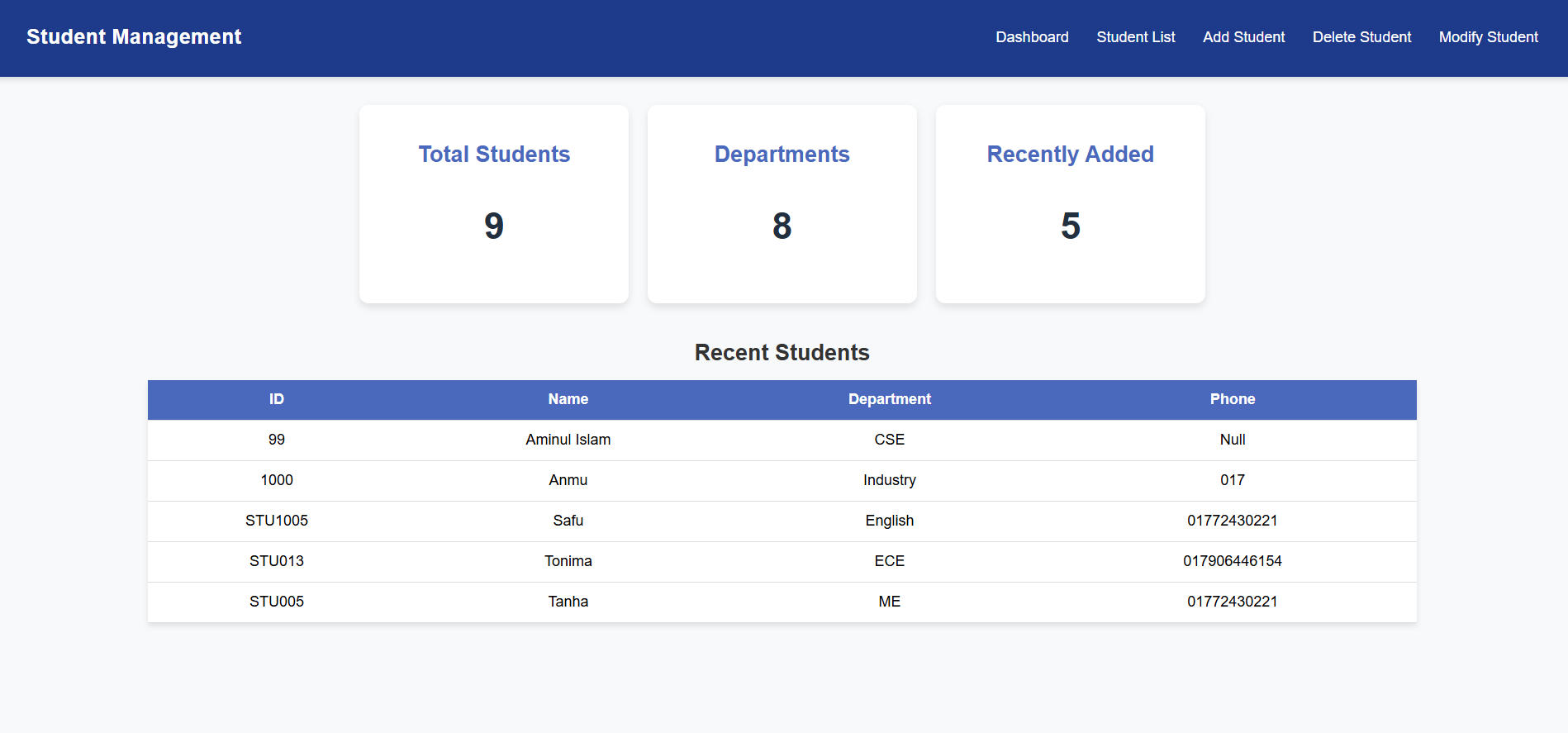
        font-size: 17px;

    }

}

**Now remove the navbar section from dashboard.html and include navbar.html in its place using:**

{% include 'navbar.html' %}



**I think that’s all for today.**