Dhaka International University

Department of Computer Science & Engineering

Django Lab Manual – 01

(A Practical Guide to Setting Up Django and Building a Simple Web Project)

Prepared by:

Md. Manirujjaman

Lecturer

Department of Computer Science and Engineering

Dhaka Internation University, Satarkul, Badda, Dhaka

Purpose of the Manual:

This manual is designed to help students understand the fundamental process of setting up Django, creating simple web applications

Semester: Fall 2025

Course Name: Markup and Scripting Language.

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.

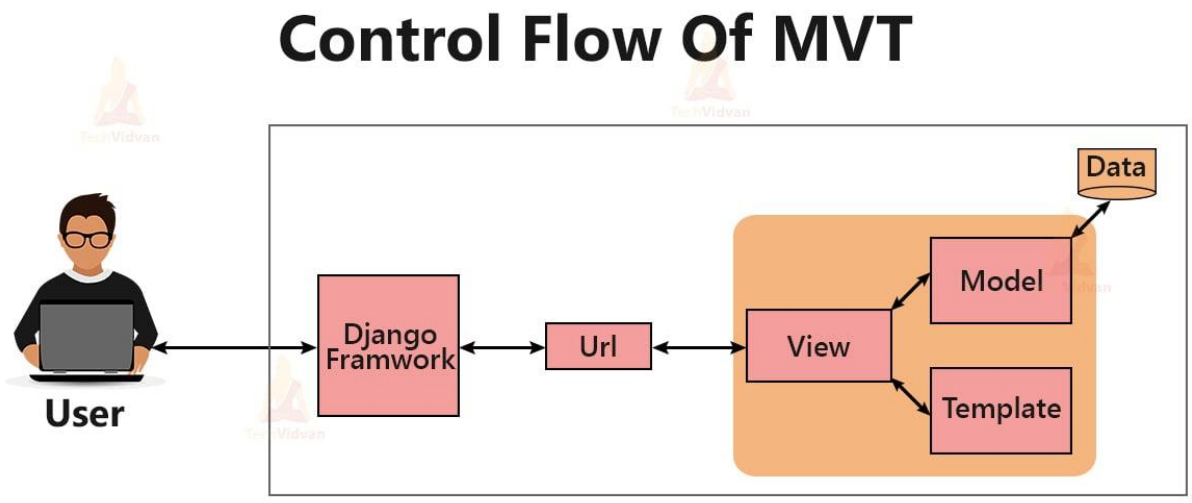
What is Django?

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:**

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

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
```

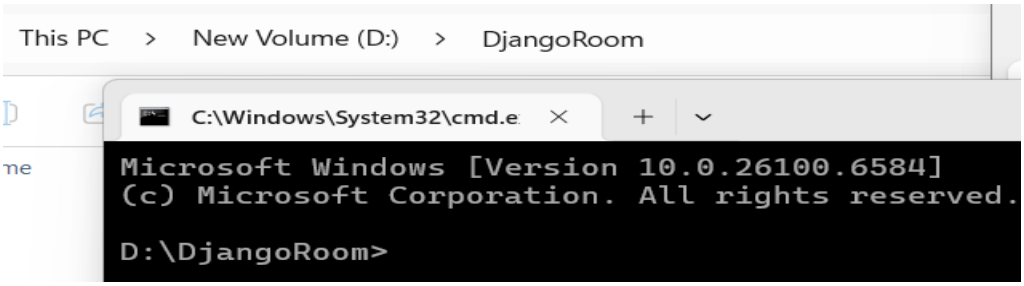
It will take some time.

Then check version

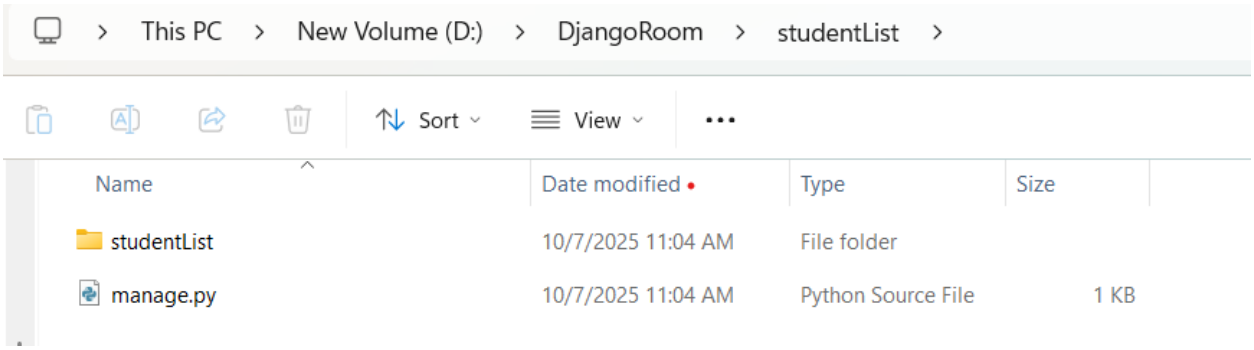
```
C:\Users\LENOVO>python -m django --version
5.2.7
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

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

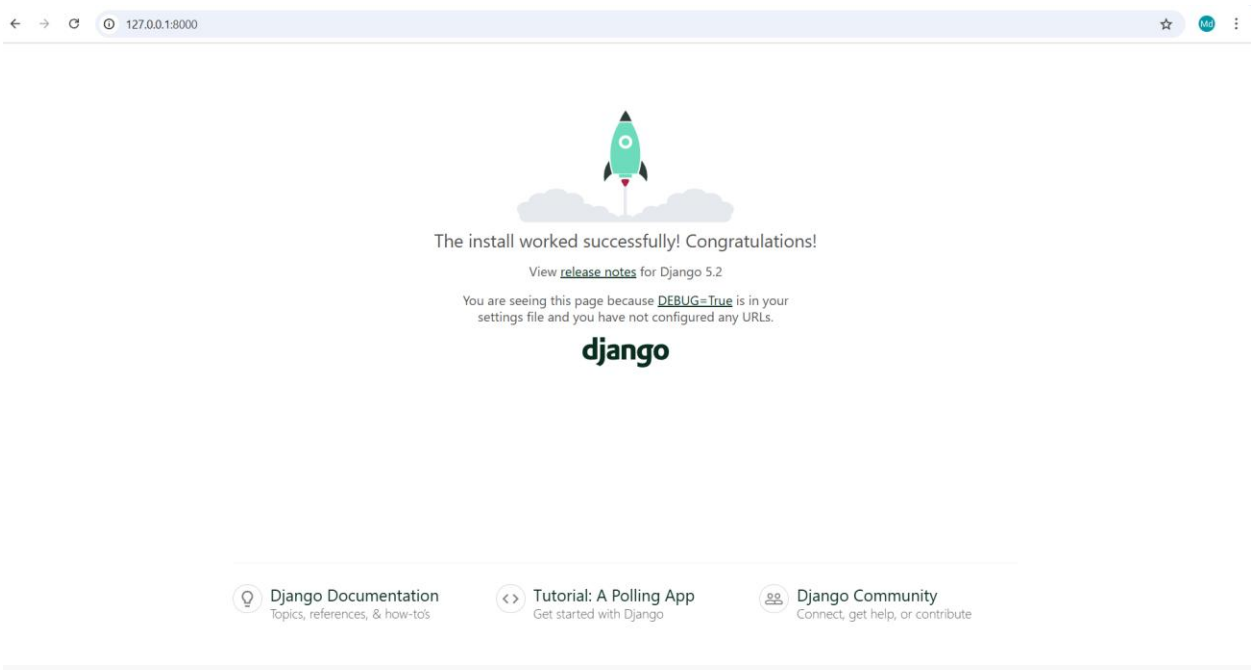


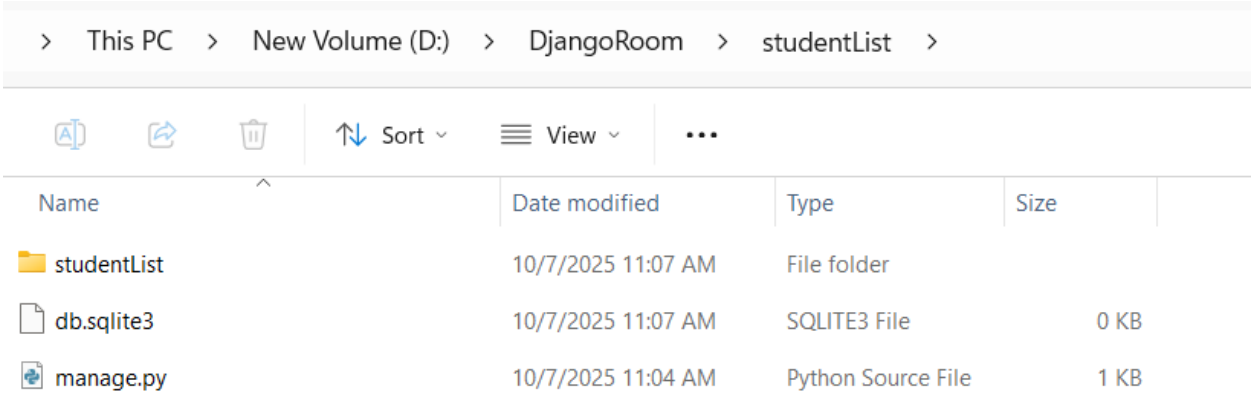

```
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\DjangoRoom>django-admin startproject studentList
```

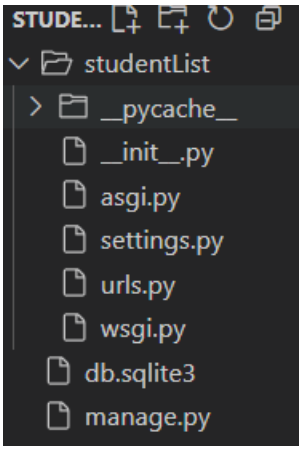



```
D:\DjangoRoom>cd studentList

D:\DjangoRoom\studentList>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until y
 auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
October 07, 2025 - 11:07:54
Django version 5.2.7, using settings 'studentList.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

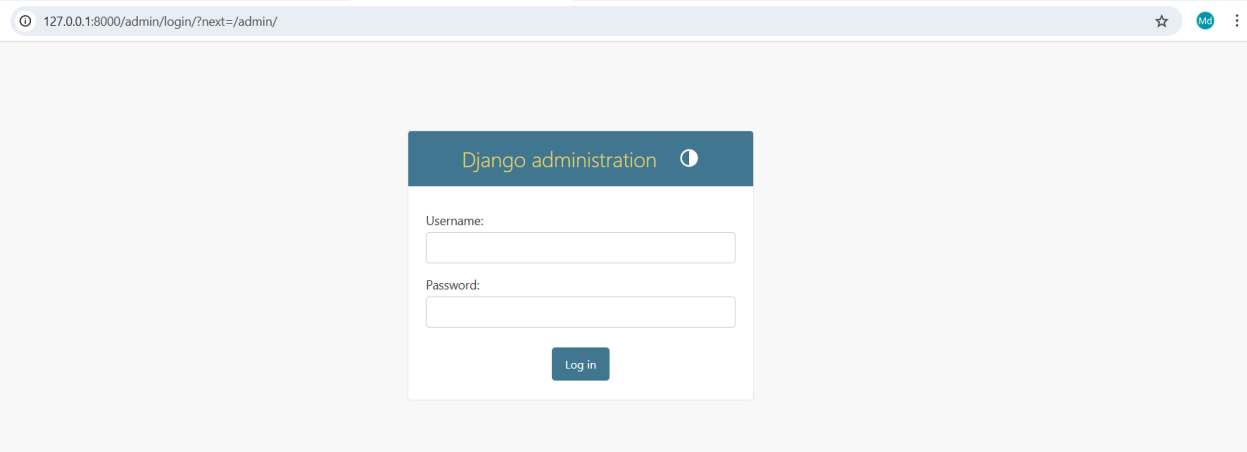
Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

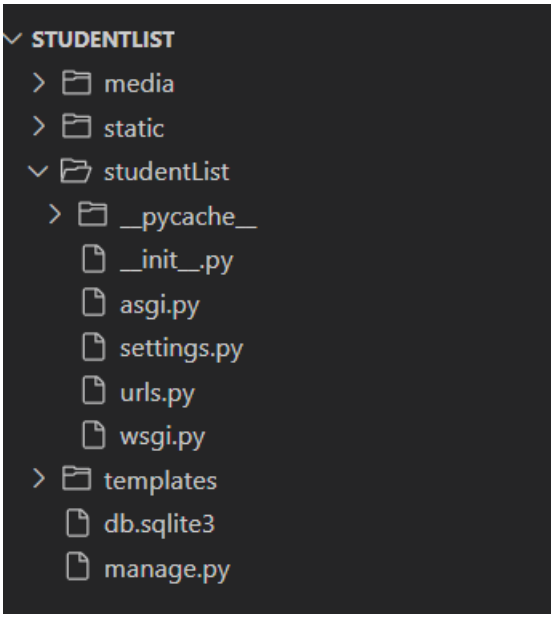






Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com





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.

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

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
```

- 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']

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.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.

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

'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',
    }
`
```

```
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']
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

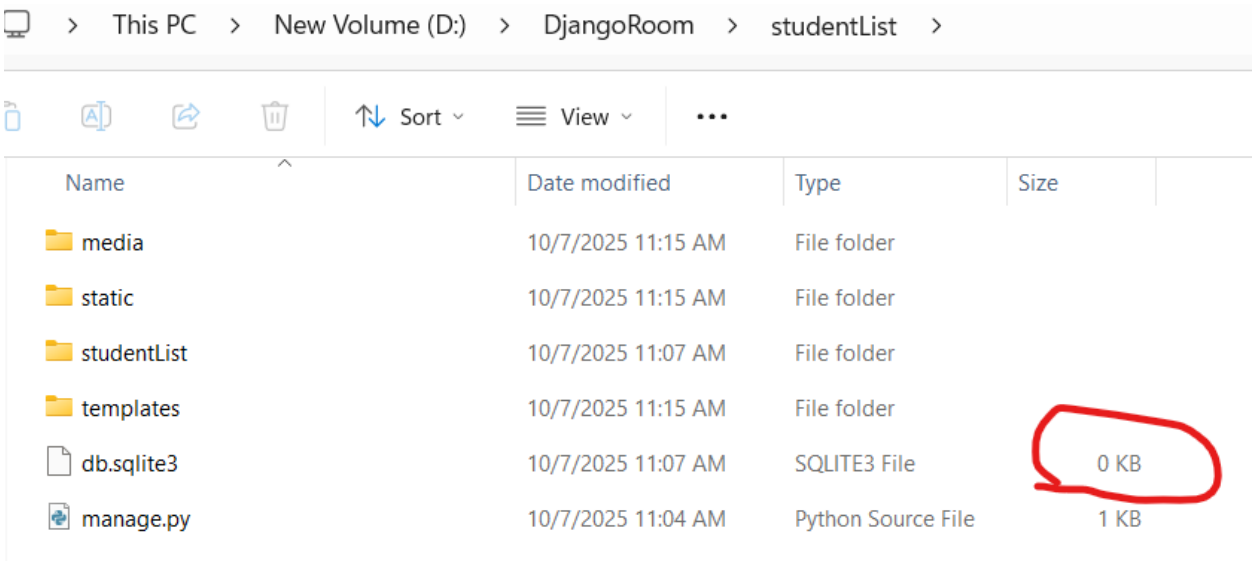
This tells Django that when you use {% static 'css/style.css' %} in HTML, it should look in the static folder.

- **Default Auto Field**

```
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

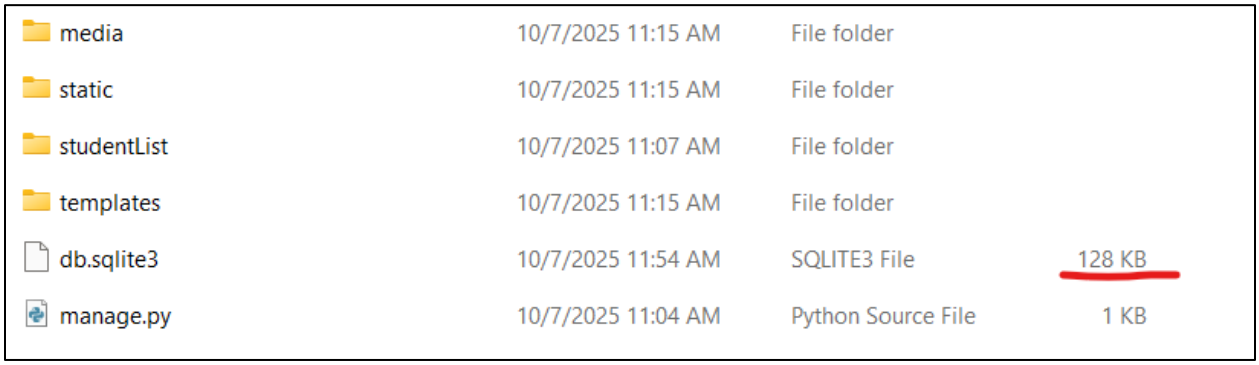
This sets how Django creates ID numbers for your database tables.
Leave it as it is.

➢ How to Migrate Default Migrations



Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

```
D:\DjangoRoom\studentList>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```



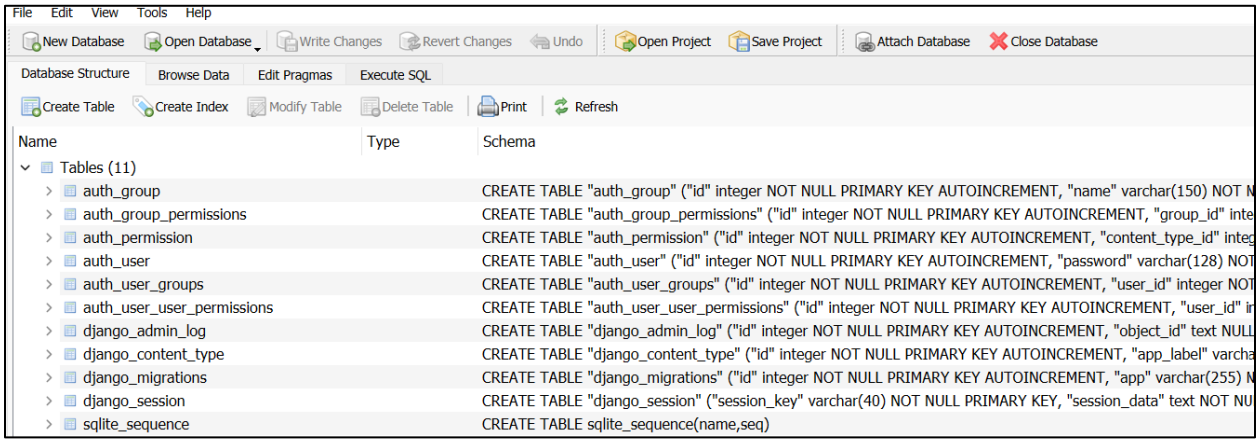
Install DB Browser for viewing SQLite3 tables. Install from following link.

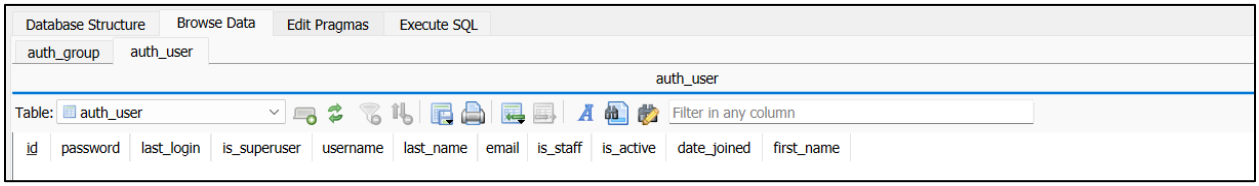
https://sqlitebrowser.org/dl/

- DB Browser for SQLite - Standard installer for 32-bit Windows
- DB Browser for SQLite - .zip (no installer) for 32-bit Windows
- DB Browser for SQLite - Standard installer for 64-bit Windows
- DB Browser for SQLite - .zip (no installer) for 64-bit Windows

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

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

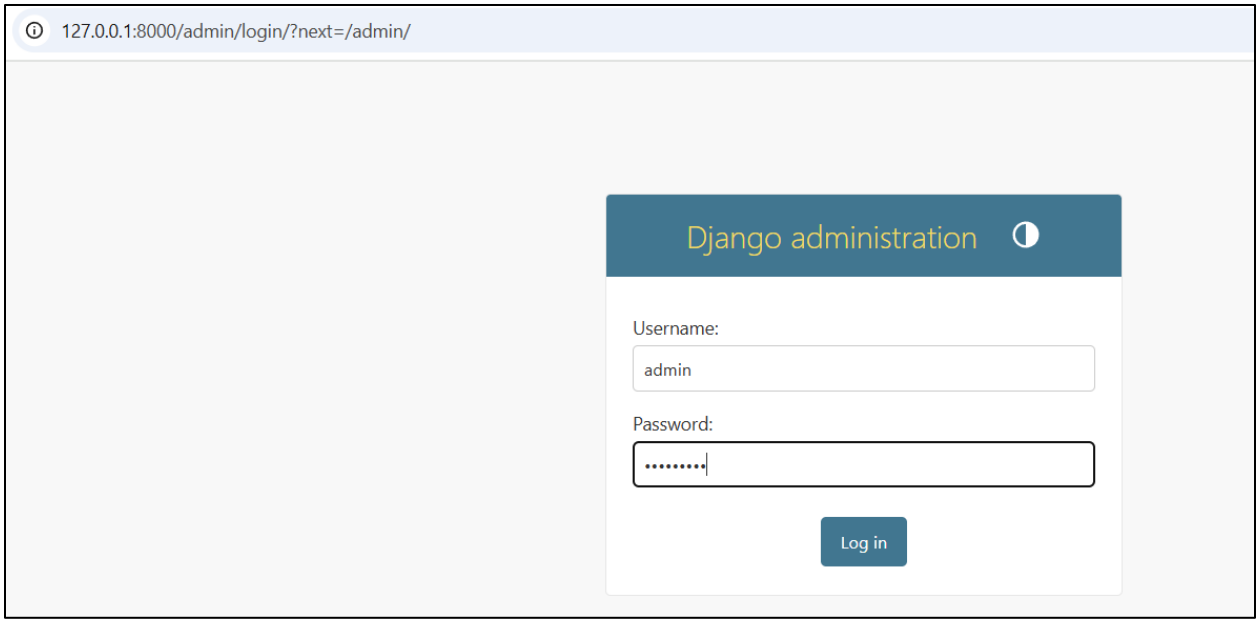




Now we will create super user to manage the admin panel.

```
D:\DjangoRoom\studentList>python manage.py createsuperuser
Username (leave blank to use 'lenovo'): admin
Email address: manirujjaman25803690
Error: Enter a valid email address.
Email address: manirujjaman25803690@gmail.com
Password:
Password (again):
The password is too similar to the username.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

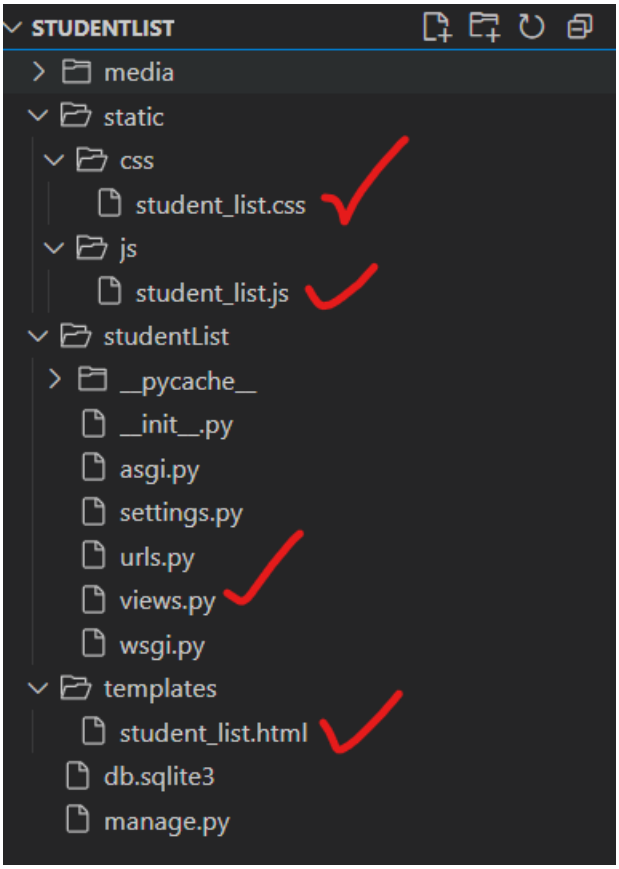
Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com



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.

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

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>
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

```
                    <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>
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

CSS: student_list.css

```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;
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

```
}
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')
```

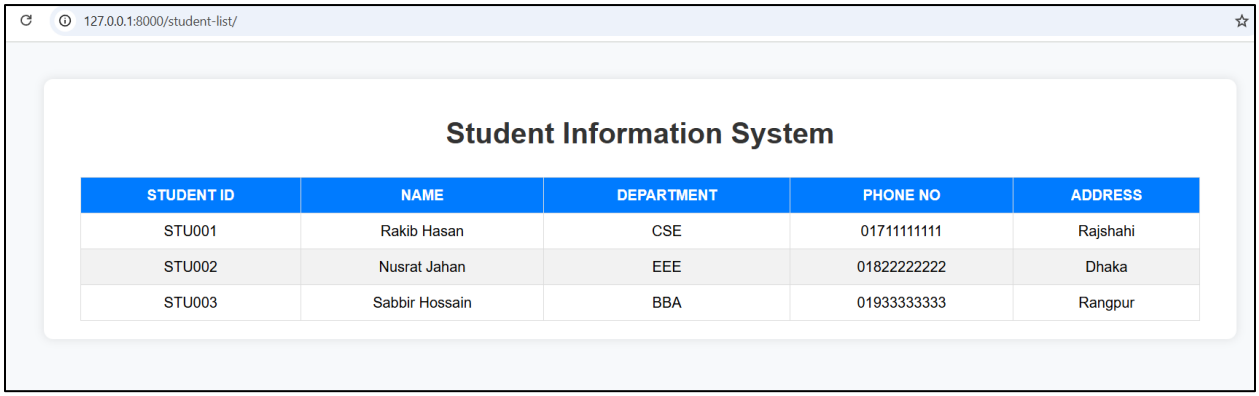
and also update **urls.py** as follows:

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

urlpatterns = [
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

```
    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
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

Step 2: Register the App

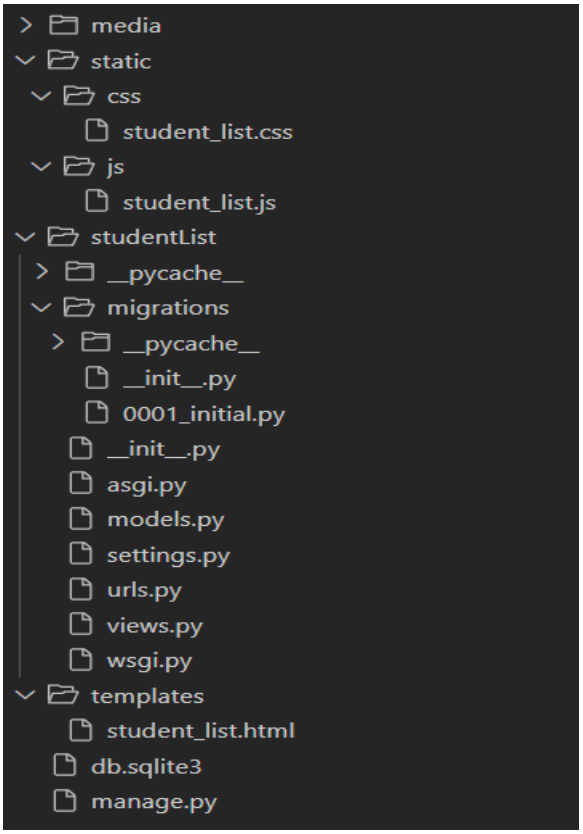
Open studentlist/settings.py

```
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

```
D:\DjangoRoom\studentList>python manage.py makemigrations studentList
Migrations for 'studentList':
  studentList\migrations\0001_initial.py
    + Create model Student
```

Your project folder will look like as following

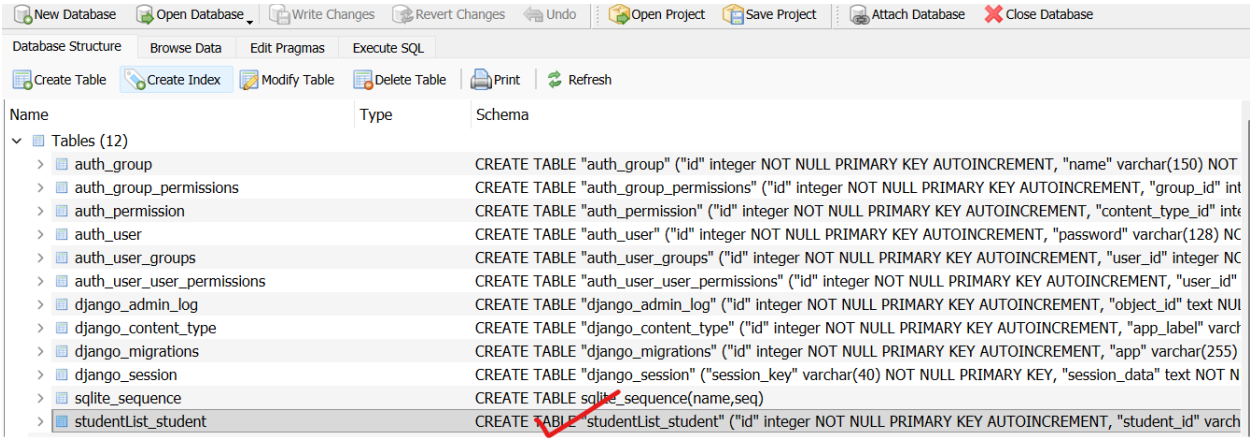


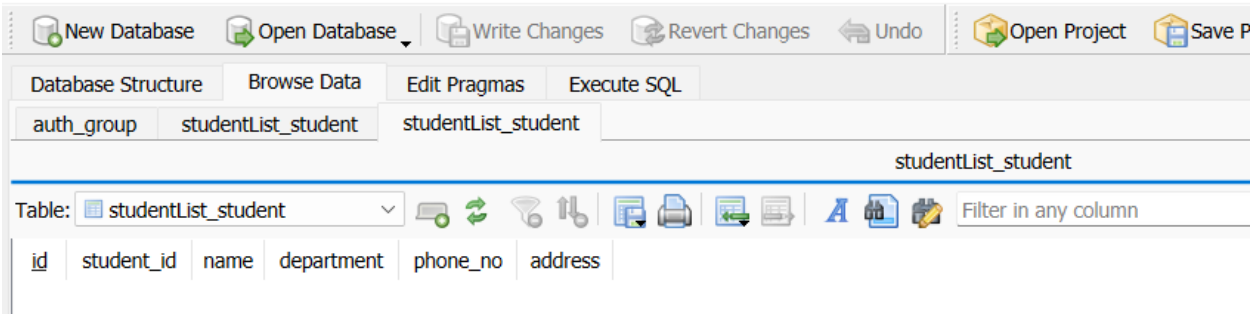
Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

Now write following command:

```
D:\DjangoRoom\studentList>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, studentList
Running migrations:
  Applying studentList.0001_initial... OK
```

studentList_student table is created in you SQLite3 database.





Add Some Data (for testing)

```
D:\DjangoRoom\studentList>python manage.py shell
```

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')
```


Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

```
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

```python
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.

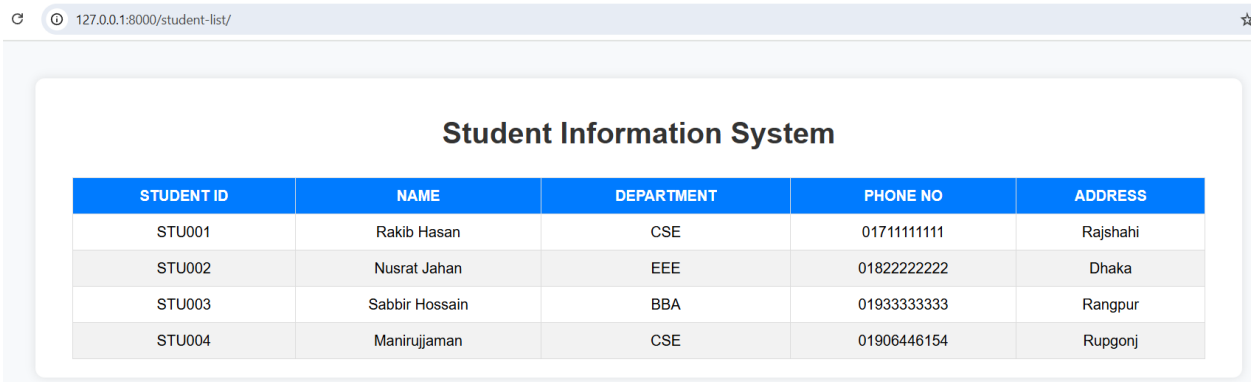
```
            <tbody>
                  {% for student in students %}
                 <tr>
                    <td>{{ student.student_id }}</td>
                    <td>{{ student.name }}</td>
                    <td>{{ student.department }}</td>
                    <td>{{ student.phone_no }}</td>
                    <td>{{ student.address }}</td>
                 </tr>
                {% empty %}
                  <tr>
                     <td colspan="5">No students found.</td>
                  </tr>
                {% endfor %}
             </tbody>
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

Now runserver again.

Add and entry to the database.

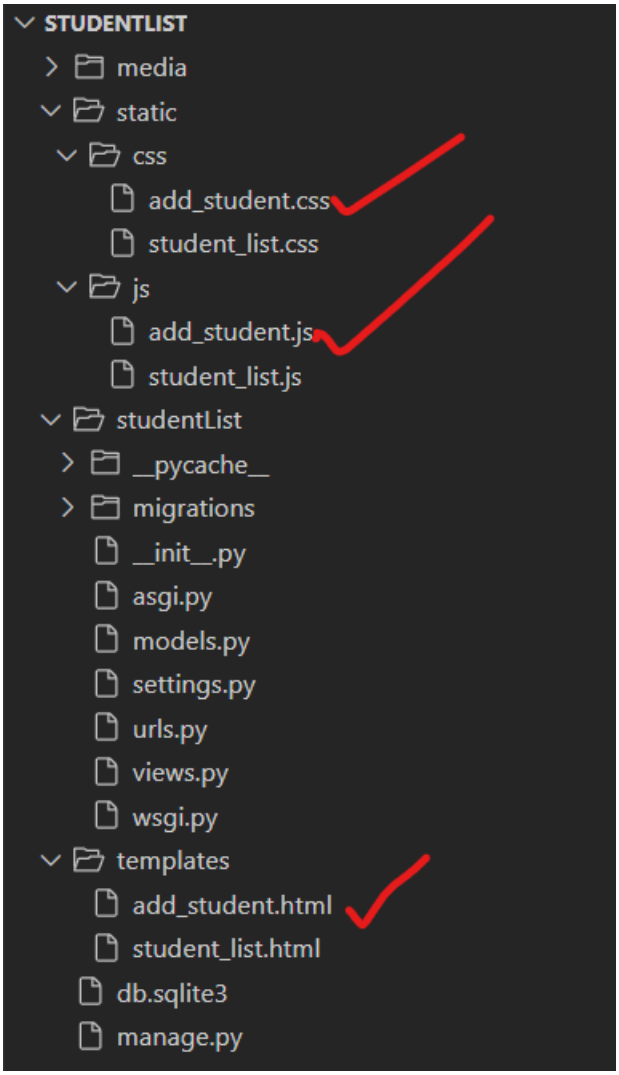
```
In [4]: Student.objects.create(student_id='STU004', name='Manirujjama
   ...: n',department = 'CSE',phone_no='01906446154', address = 'Rupg
   ...: onj')
Out[4]: <Student: Manirujjaman>

In [5]: exit()
```



Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

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!



Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

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>
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

```
        </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;
}
```

```css
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;
}
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

```
#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()
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

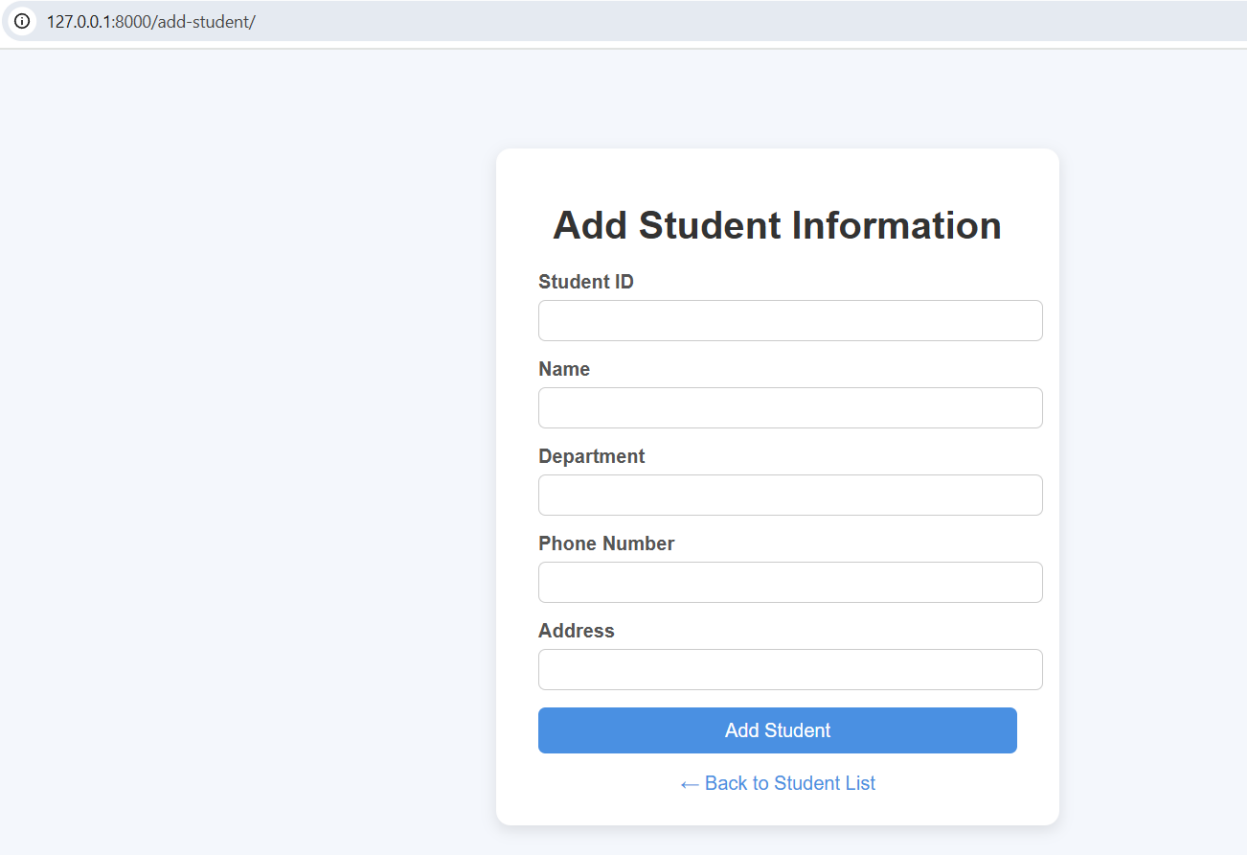
```
    context={'students':students}
    return render(request, 'student_list.html', context)

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'),
]
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

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)
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

```python
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

```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 -->
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

```
        <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>
```

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

Updated CSS.(add_student.css)

```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;
}
```

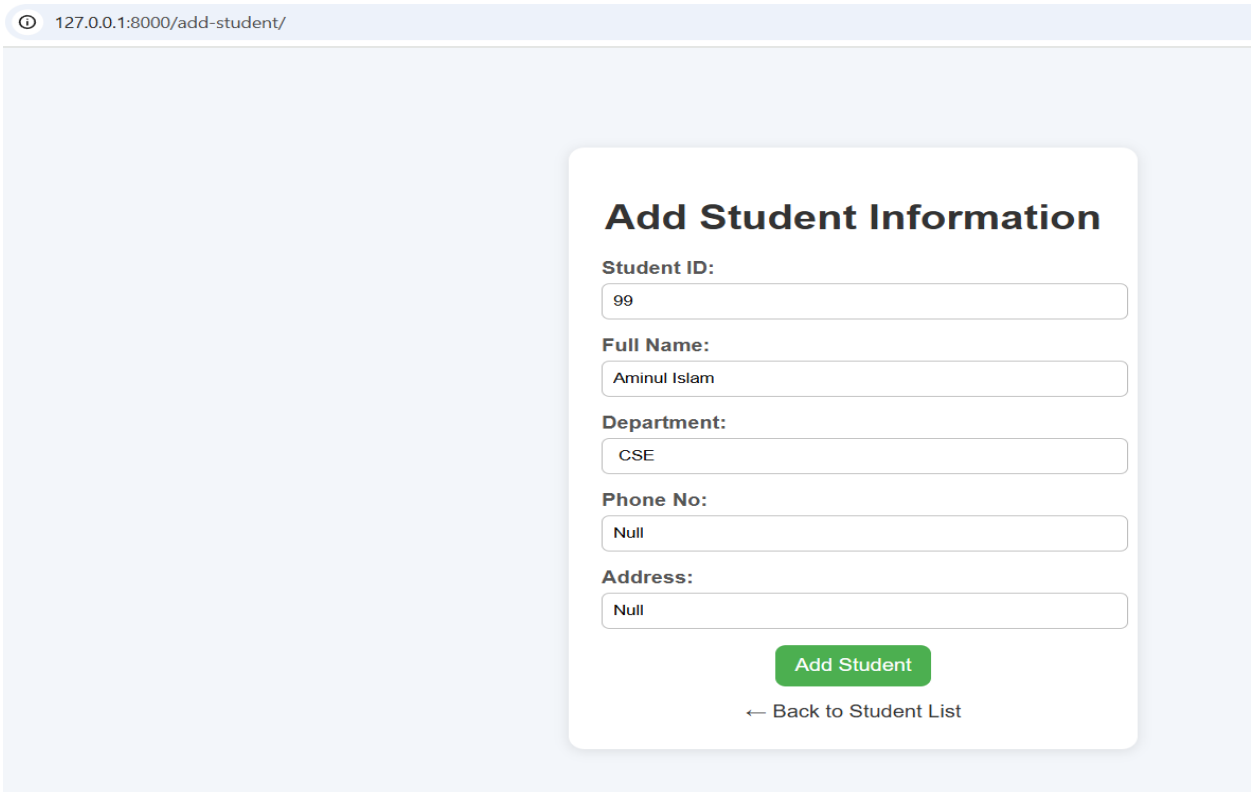
Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

```css
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;
}
```

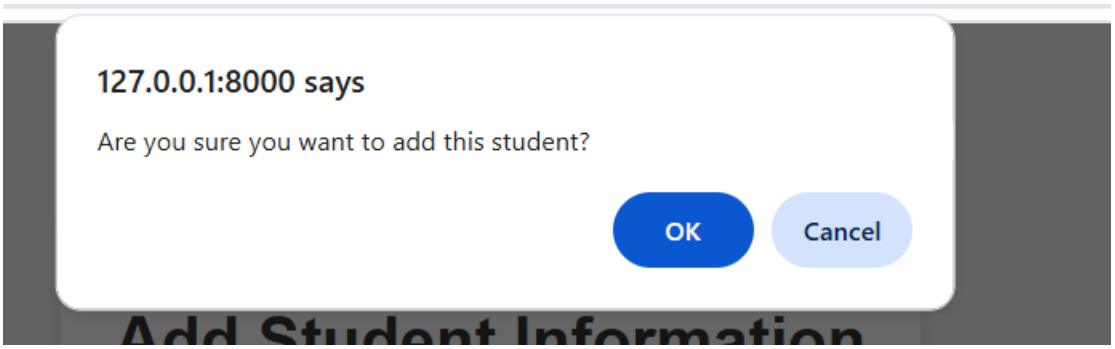
Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

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();
    }
});
```



Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com



Student Information System

STUDENT ID	NAME	DEPARTMENT	PHONE NO	ADDRESS
STU001	Rakib Hasan	CSE	01711111111	Rajshahi
STU002	Nusrat Jahan	EEE	01822222222	Dhaka
STU003	Sabbir Hossain	BBA	01933333333	Rangpur
STU004	Manirujjaman	CSE	01906446154	Rupgonj
STU005	Tanha	ME	01772430221	DHaka
STU013	Tonima	ECE	017906446154	Dinajpur
STU1005	Safu	English	01772430221	Goal
1000	Anmu	Industry	017	Dhaka
99	Aminul Islam	CSE	Null	Null

That’s All for Today

Md. Manirujjaman
Lecturer, CSE, DIU
manirujjaman25803690@gmail.com

Knowledge is Power
DIU
Estd. 7th April, 1995