UniversityPortal — Documentation

# UniversityPortal — Documentation

## Overview

This archive contains a Django project named \*\*UniversityPortal\*\* with an application \*\*students\*\*.

The students app provides full CRUD (Create / Read / Update / Delete) operations via a simple Bootstrap UI and admin integration.

Files included (root of archive)

```

UniversityPortal\_with\_crud/ <- project root inside zip

├─ manage.py

#!/usr/bin/env python

import os

import sys

if \_\_name\_\_ == '\_\_main\_\_':

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'UniversityPortal.settings')

try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed?"

) from exc

execute\_from\_command\_line(sys.argv)

├─ UniversityPortal/

│ ├─ \_\_init\_\_.py

│ ├─ settings.py

from pathlib import Path

import os

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

SECRET\_KEY = 'replace-this-with-a-secure-key'

DEBUG = True

ALLOWED\_HOSTS = []

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'students',

]

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',

]

ROOT\_URLCONF = 'UniversityPortal.urls'

TEMPLATES = [

{

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

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

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

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

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

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

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

],

},

},

]

WSGI\_APPLICATION = 'UniversityPortal.wsgi.application'

DATABASES = {

'default': {

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

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

}

}

AUTH\_PASSWORD\_VALIDATORS = []

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'Asia/Kolkata'

USE\_I18N = True

USE\_TZ = True

STATIC\_URL = '/static/'

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

│ ├─ urls.py

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

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

]

│ ├─ wsgi.py

import os

from django.core.wsgi import get\_wsgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'UniversityPortal.settings')

application = get\_wsgi\_application()

│ └─ asgi.py

import os

from django.core.asgi import get\_asgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'UniversityPortal.settings')

application = get\_asgi\_application()

├─ students/

│ ├─ \_\_init\_\_.py

│ ├─ admin.py

from django.contrib import admin

from .models import Student

@admin.register(Student)

class StudentAdmin(admin.ModelAdmin):

list\_display = ('first\_name', 'last\_name', 'enrollment\_number', 'email', 'joined\_date')

search\_fields = ('first\_name', 'last\_name', 'enrollment\_number', 'email')

│ ├─ apps.py

from django.apps import AppConfig

class StudentsConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'students'

│ ├─ forms.py

from django import forms

from .models import Student

class StudentForm(forms.ModelForm):

joined\_date = forms.DateField(widget=forms.DateInput(attrs={'type': 'date'}))

class Meta:

model = Student

fields = ['first\_name', 'last\_name', 'enrollment\_number', 'email', 'joined\_date']

│ ├─ models.py

from django.db import models

class Student(models.Model):

first\_name = models.CharField(max\_length=100)

last\_name = models.CharField(max\_length=100)

enrollment\_number = models.CharField(max\_length=50, unique=True)

email = models.EmailField(blank=True)

joined\_date = models.DateField()

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

return f"{self.first\_name} {self.last\_name} ({self.enrollment\_number})"

│ ├─ urls.py

from django.urls import path

from . import views

urlpatterns = [

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

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

path('edit/<int:pk>/', views.student\_edit, name='student\_edit'),

path('delete/<int:pk>/', views.student\_delete, name='student\_delete'),

path('view/<int:pk>/', views.student\_detail, name='student\_detail'),

]

│ └─ views.py

from django.shortcuts import render, get\_object\_or\_404, redirect

from django.core.paginator import Paginator

from django.db.models import Q

from .models import Student

from .forms import StudentForm

def student\_list(request):

q = request.GET.get('q', '')

students = Student.objects.all().order\_by('-joined\_date')

if q:

students = students.filter(Q(first\_name\_\_icontains=q) | Q(last\_name\_\_icontains=q) | Q(enrollment\_number\_\_icontains=q) | Q(email\_\_icontains=q))

paginator = Paginator(students, 10)

page\_number = request.GET.get('page')

page\_obj = paginator.get\_page(page\_number)

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

def student\_add(request):

if request.method == 'POST':

form = StudentForm(request.POST)

if form.is\_valid():

form.save()

return redirect('student\_list')

else:

form = StudentForm()

return render(request, 'students/student\_form.html', {'form': form, 'title': 'Add Student'})

def student\_edit(request, pk):

student = get\_object\_or\_404(Student, pk=pk)

if request.method == 'POST':

form = StudentForm(request.POST, instance=student)

if form.is\_valid():

form.save()

return redirect('student\_detail', pk=student.pk)

else:

form = StudentForm(instance=student)

return render(request, 'students/student\_form.html', {'form': form, 'title': 'Edit Student'})

def student\_delete(request, pk):

student = get\_object\_or\_404(Student, pk=pk)

if request.method == 'POST':

student.delete()

return redirect('student\_list')

return render(request, 'students/student\_confirm\_delete.html', {'student': student})

def student\_detail(request, pk):

student = get\_object\_or\_404(Student, pk=pk)

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

├─ templates/

│ └─ students/

│ ├─ base.html

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>{% block title %}University Portal{% endblock %}</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light mb-4">

<div class="container">

<a class="navbar-brand" href="{% url 'student\_list' %}">University Portal</a>

<div class="collapse navbar-collapse">

<ul class="navbar-nav me-auto">

<li class="nav-item"><a class="nav-link" href="{% url 'student\_add' %}">Add Student</a></li>

</ul>

<form class="d-flex" method="get" action="{% url 'student\_list' %}">

<input class="form-control me-2" name="q" placeholder="Search students" value="{{ q|default:'' }}">

<button class="btn btn-outline-primary" type="submit">Search</button>

</form>

</div>

</div>

</nav>

<div class="container">

{% block content %}{% endblock %}

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

│ ├─ student\_list.html

{% extends 'students/base.html' %}

{% block title %}Students{% endblock %}

{% block content %}

<h1>Students</h1>

{% if page\_obj.object\_list %}

<table class="table table-striped">

<thead><tr><th>#</th><th>Name</th><th>Enrollment</th><th>Email</th><th>Joined</th><th>Actions</th></tr></thead>

<tbody>

{% for student in page\_obj.object\_list %}

<tr>

<td>{{ forloop.counter0|add:page\_obj.start\_index }}</td>

<td>{{ student.first\_name }} {{ student.last\_name }}</td>

<td>{{ student.enrollment\_number }}</td>

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

<td>{{ student.joined\_date }}</td>

<td>

<a class="btn btn-sm btn-primary" href="{% url 'student\_detail' student.pk %}">View</a>

<a class="btn btn-sm btn-secondary" href="{% url 'student\_edit' student.pk %}">Edit</a>

<a class="btn btn-sm btn-danger" href="{% url 'student\_delete' student.pk %}">Delete</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

<nav>

<ul class="pagination">

{% if page\_obj.has\_previous %}

<li class="page-item"><a class="page-link" href="?page={{ page\_obj.previous\_page\_number }}{% if q %}&q={{ q }}{% endif %}">Previous</a></li>

{% else %}

<li class="page-item disabled"><span class="page-link">Previous</span></li>

{% endif %}

<li class="page-item disabled"><span class="page-link">Page {{ page\_obj.number }} of {{ page\_obj.paginator.num\_pages }}</span></li>

{% if page\_obj.has\_next %}

<li class="page-item"><a class="page-link" href="?page={{ page\_obj.next\_page\_number }}{% if q %}&q={{ q }}{% endif %}">Next</a></li>

{% else %}

<li class="page-item disabled"><span class="page-link">Next</span></li>

{% endif %}

</ul>

</nav>

{% else %}

<p>No students found. <a href="{% url 'student\_add' %}">Add one</a>.</p>

{% endif %}

{% endblock %}

│ ├─ student\_form.html

{% extends 'students/base.html' %}

{% block title %}{{ title }}{% endblock %}

{% block content %}

<h1>{{ title }}</h1>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button class="btn btn-primary" type="submit">Save</button>

<a href="{% url 'student\_list' %}" class="btn btn-secondary">Cancel</a>

</form>

{% endblock %}

│ ├─ student\_detail.html

{% extends 'students/base.html' %}

{% block title %}{{ student.first\_name }} {{ student.last\_name }}{% endblock %}

{% block content %}

<h1>{{ student.first\_name }} {{ student.last\_name }}</h1>

<ul class="list-group">

<li class="list-group-item"><strong>Enrollment:</strong> {{ student.enrollment\_number }}</li>

<li class="list-group-item"><strong>Email:</strong> {{ student.email }}</li>

<li class="list-group-item"><strong>Joined:</strong> {{ student.joined\_date }}</li>

</ul>

<p class="mt-3">

<a href="{% url 'student\_edit' student.pk %}" class="btn btn-secondary">Edit</a>

<a href="{% url 'student\_delete' student.pk %}" class="btn btn-danger">Delete</a>

<a href="{% url 'student\_list' %}" class="btn btn-link">Back to list</a>

</p>

{% endblock %}

│ └─ student\_confirm\_delete.html

{% extends 'students/base.html' %}

{% block title %}Delete Student{% endblock %}

{% block content %}

<h1>Delete {{ student.first\_name }} {{ student.last\_name }}</h1>

<p>Are you sure you want to delete this student?</p>

<form method="post">

{% csrf\_token %}

<button class="btn btn-danger" type="submit">Yes, delete</button>

<a href="{% url 'student\_detail' student.pk %}" class="btn btn-secondary">Cancel</a>

</form>

{% endblock %}

├─ requirements.txt

Django==4.2

├─ README.md

**# UniversityPortal (starter)**

This project was scaffolded to satisfy the request: create a Django project named UniversityPortal

and an app called students, register it, and verify access to Django admin.

Quick start:

1. Create and activate a virtual environment:

python3 -m venv venv

source venv/bin/activate # mac/linux

venv\Scripts\activate # windows

2. Install dependencies:

pip install -r requirements.txt

3. Apply migrations:

python manage.py migrate

4. Create a superuser:

python manage.py createsuperuser

5. Run the server:

python manage.py runserver

6. Visit:

- Home: http://127.0.0.1:8000/ (students home)

- Admin: http://127.0.0.1:8000/admin/ (use superuser creds)

## Purpose

- Provide a starter University Portal with a `students` app.

- Demonstrate Django patterns: models, admin registration, ModelForm, views, pagination, search, templates, and routing.

- Ready for local development and extension.

---

## Project structure (key files)

### `students/models.py`

Defines a `Student` model with fields:

- `first\_name` `CharField`

- `last\_name` `CharField`

- `enrollment\_number` `CharField` (unique)

- `email` `EmailField` (optional)

- `joined\_date` `DateField`

### `students/forms.py`

Defines `StudentForm` (ModelForm) using `joined\_date` widget type=date.

### `students/views.py`

Provides:

- `student\_list` — list with search & pagination

- `student\_add` — create

- `student\_edit` — update

- `student\_delete` — delete (confirmation)

- `student\_detail` — view single student

### `students/urls.py`

Routes:

```

/ -> student\_list (name='student\_list')

/add/ -> student\_add

/edit/<pk>/ -> student\_edit

/delete/<pk> -> student\_delete

/view/<pk> -> student\_detail

```

### `UniversityPortal/urls.py`

Includes `students.urls` at root and the admin at `/admin/`.

### Templates

Located in `templates/students/` and use Bootstrap CDN for styling. `base.html` is the layout template.

---

## Quick start (local)

1. Unzip the archive and `cd` into the project folder (the folder containing `manage.py`).

2. Create and activate a virtual environment:

- mac/linux:

```bash

python3 -m venv venv

source venv/bin/activate

```

- windows:

```bash

python -m venv venv

venv\Scripts\activate

```

3. Install dependencies:

```bash

pip install -r requirements.txt

```

4. Apply migrations:

```bash

python manage.py makemigrations

python manage.py migrate

```

5. Create a superuser (for admin access):

```bash

python manage.py createsuperuser

```

6. Run server:

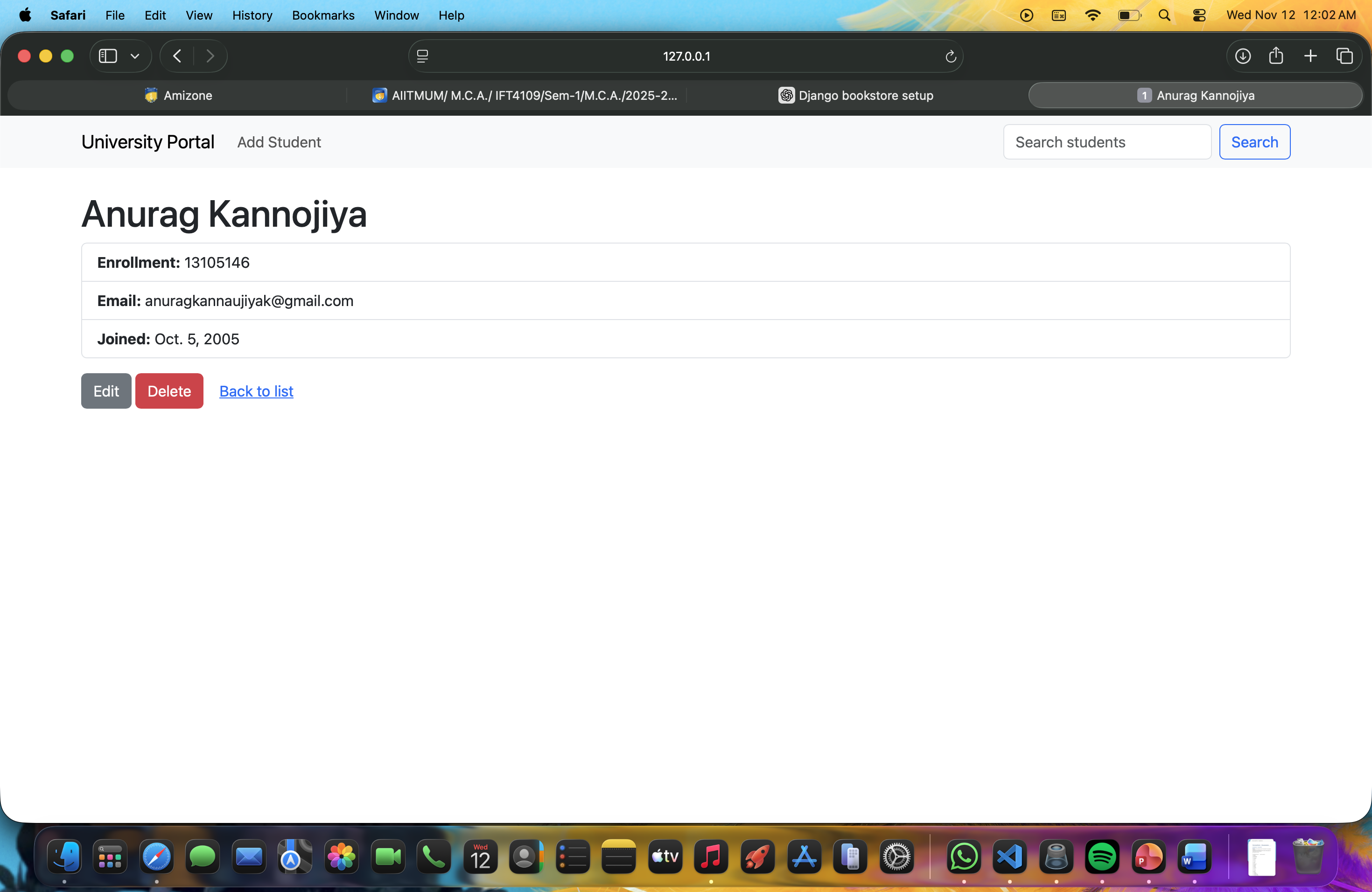
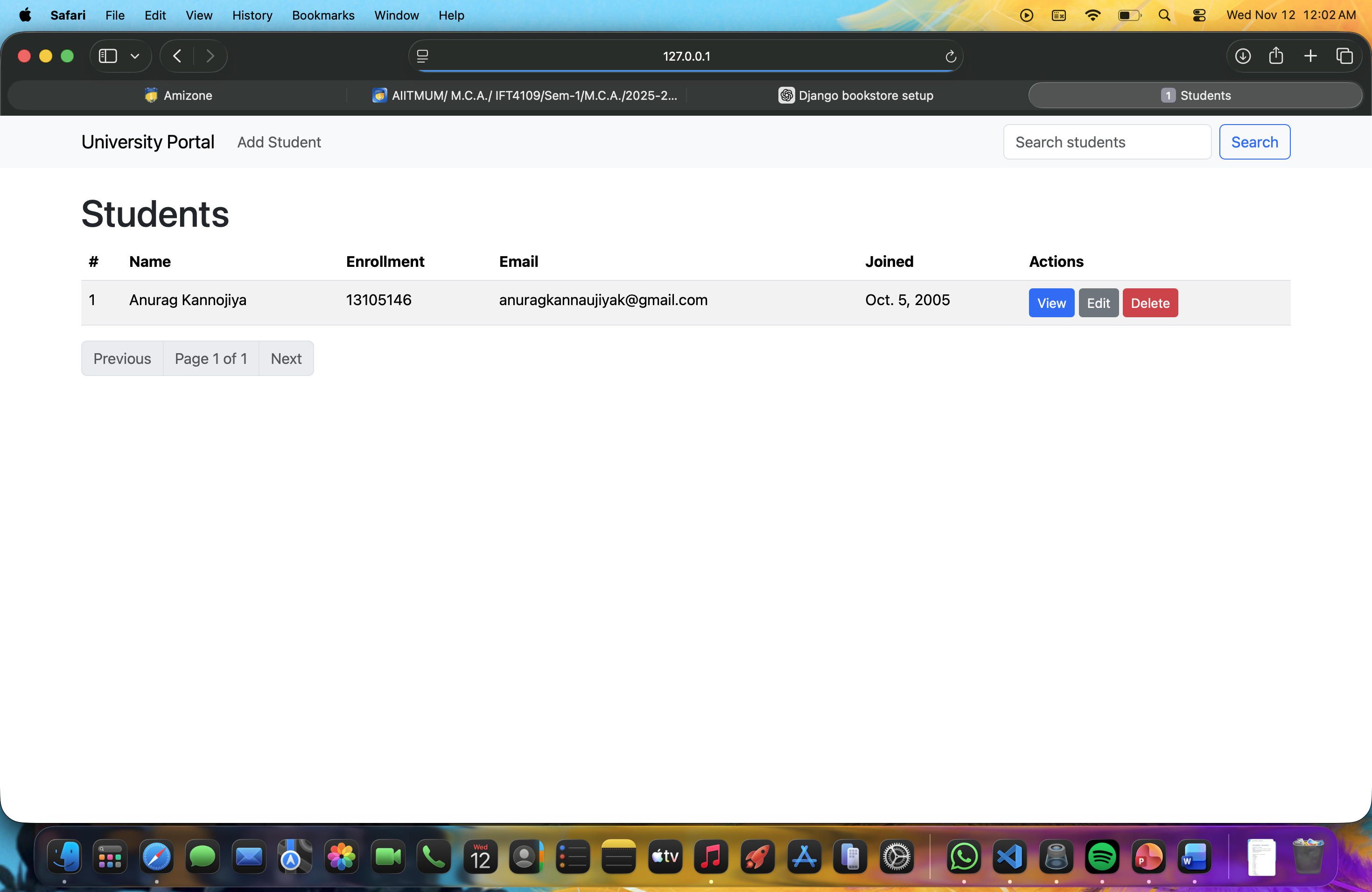
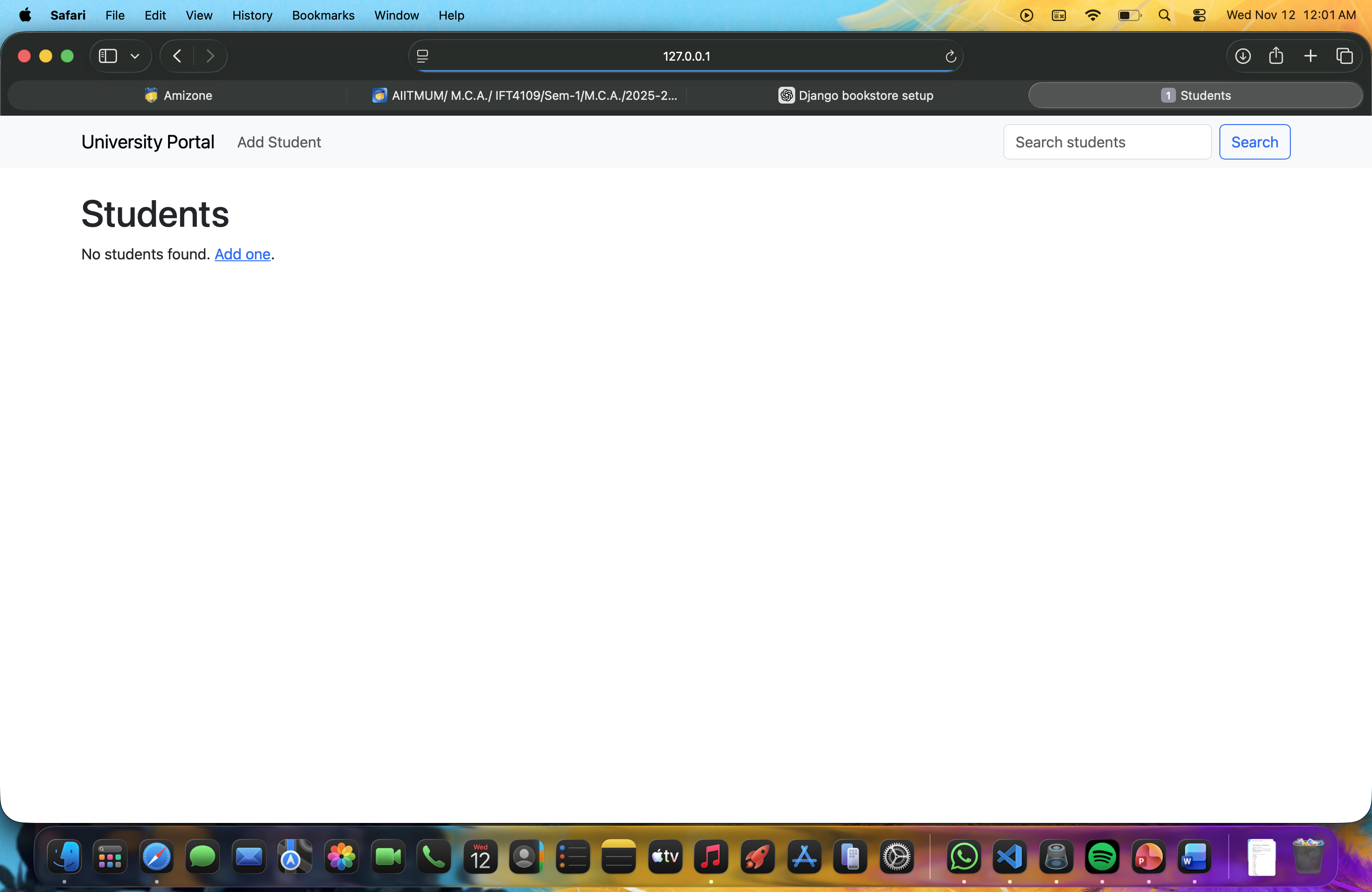
```bash

python manage.py runserver

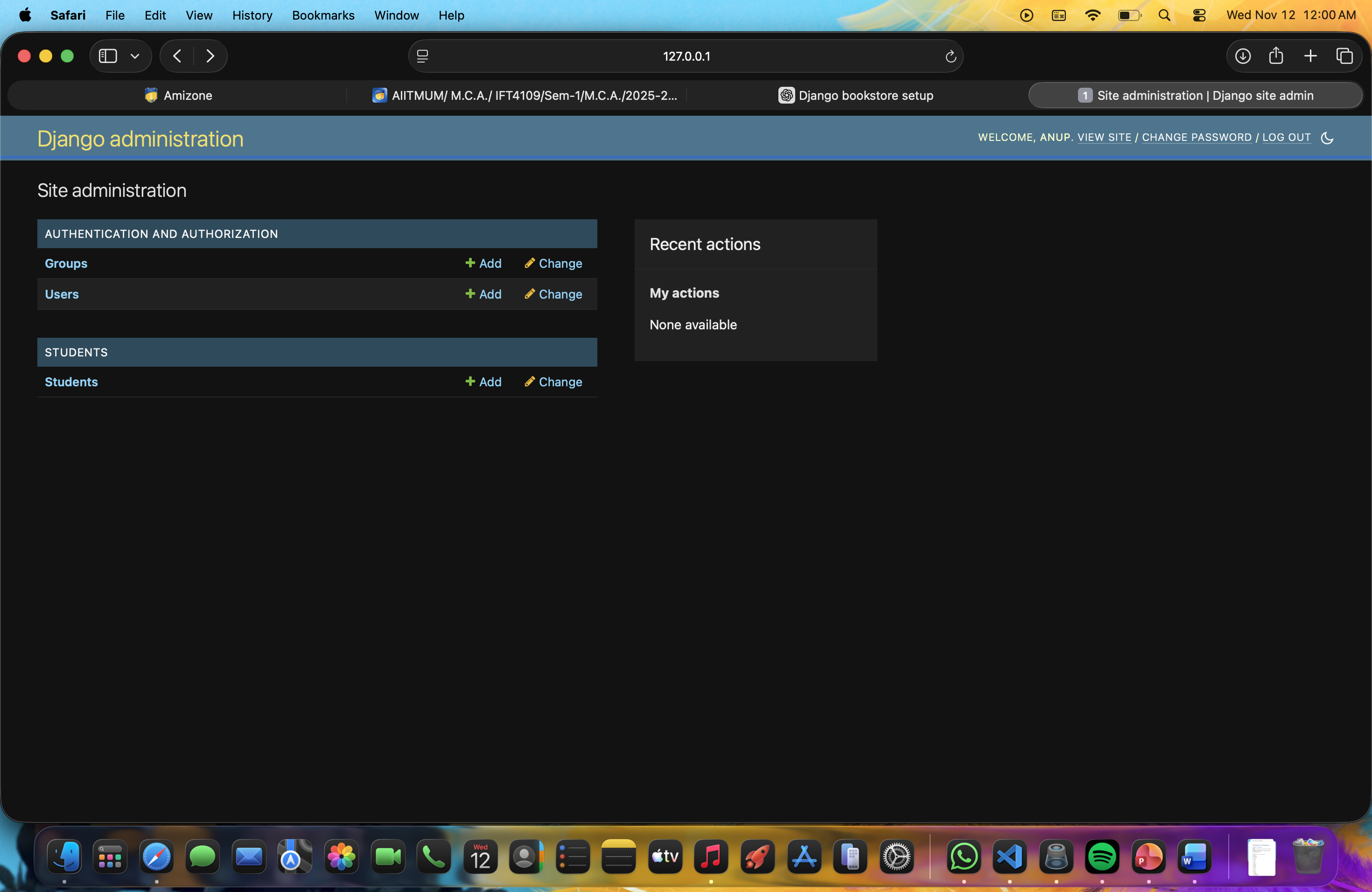
```

7. Visit:

- Students list: `http://127.0.0.1:8000/`



- Admin: `http://127.0.0.1:8000/admin/`



## Common issues & fixes

### `python` not found

Install Python and add it to PATH. On macOS/Linux use `python3` if necessary.

### `OperationalError: no such table: students\_student`

You need to run migrations:

```bash

python manage.py makemigrations

python manage.py migrate

```

If you previously used a different DB file or want a fresh start in dev:

```bash

rm db.sqlite3

rm -rf students/migrations/0\*.py

python manage.py makemigrations

python manage.py migrate

```

\*\*Warning:\*\* deleting `db.sqlite3` removes all development data.

### Admin page error like `AttributeError: 'super' object has no attribute 'dicts'`

- Try clearing `.pyc` and `\_\_pycache\_\_`:

```bash

find . -name "\_\_pycache\_\_" -type d -exec rm -rf {} +

find . -name "\*.pyc" -delete

```

- Ensure there is no in-project `context.py` or a variable named `super` that shadows builtins. Search the project for suspicious overrides:

```bash

grep -RIn "class Context" .

grep -RIn "super\s\*=" .

```

- If the issue appears only on Python 3.14, consider using Python 3.11/3.12 in a virtualenv to test compatibility.

---

## Extending the project (ideas)

- Add authentication for staff-only CRUD pages.

- Add Course and Enrollment models (many-to-many).

- Add CSV import/export for Students.

- Add image upload (profile photo) and media serving.

- Add Django REST Framework API endpoints.

- Add basic unit tests (model and view tests).

---

## Useful management commands

- Run checks:

```bash

python manage.py check

```

- Show migrations:

```bash

python manage.py showmigrations

```

- Run Django shell:

```bash

python manage.py shell

```

- Create sample data quickly:

```python

from students.models import Student

Student.objects.create(first\_name='Test', last\_name='User', enrollment\_number='ENR100', joined\_date='2025-11-11')

```

---

## Contact / Next steps

Tell me which of the following you want next and I’ll prepare a new zip or patch:

1. Add admin-only access for CRUD pages.

2. Auto-create a dev superuser + seed sample students.

3. Add CSV import/export.

4. Add Docker + docker-compose.

5. Convert to use Django REST Framework (API).

---

Generated on: 2025-11-11 (Asia/Kolkata)