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

```python
#!/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

```python
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'})
```

```python
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
```

```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):

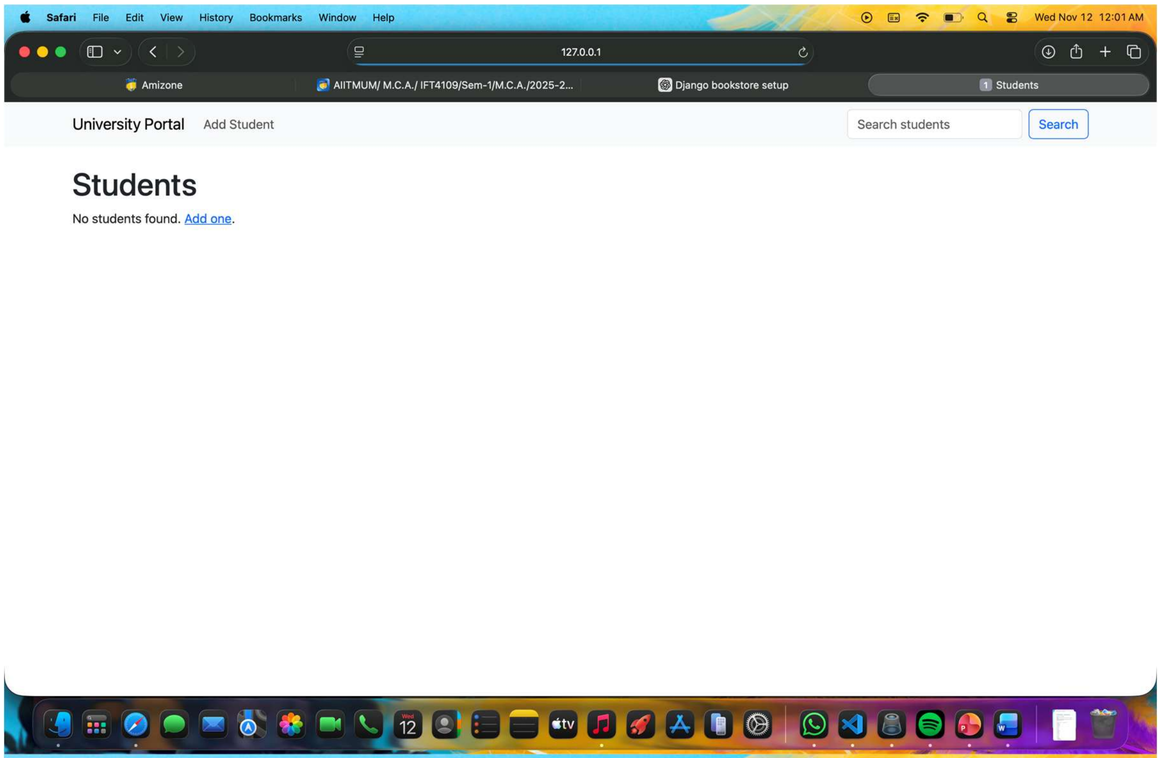
```bash
python manage.py createsuperuser
```

6. Run server:

```bash
python manage.py runserver
```

7. Visit:

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



Safari File Edit View History Bookmarks Window Help Wed Nov 12 12:01 AM

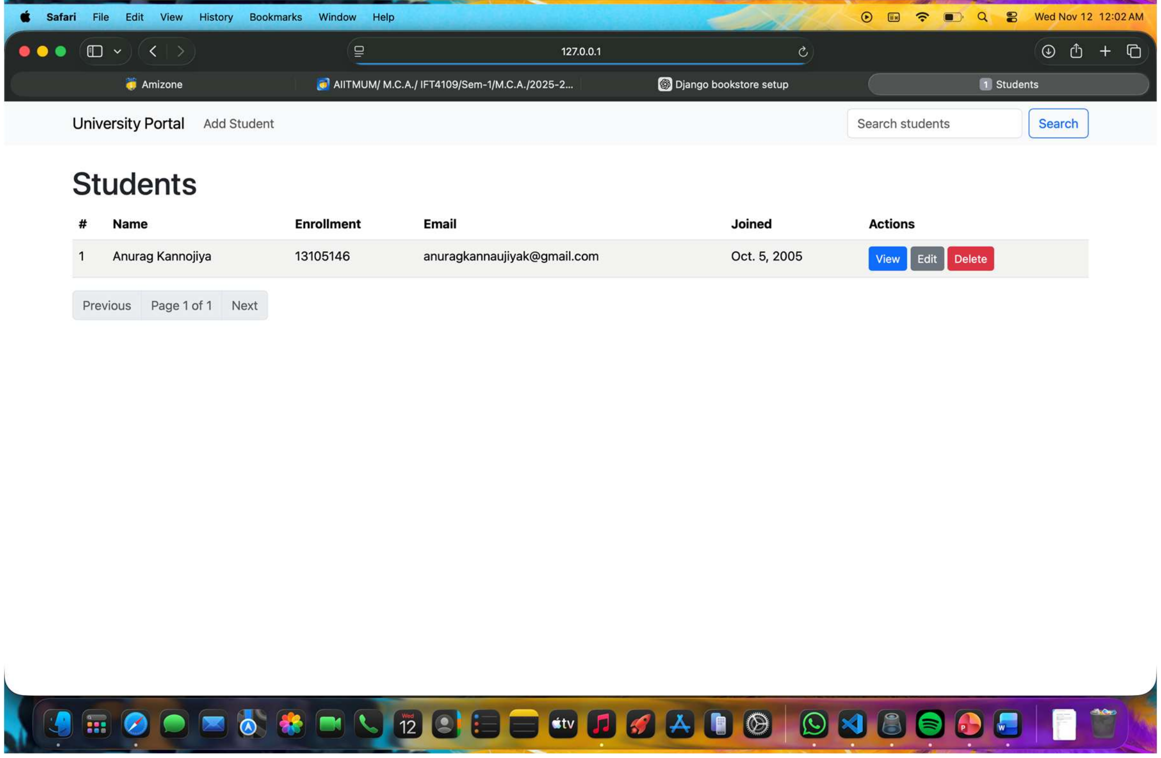
127.0.0.1

Amizone | AIITMUM/ M.C.A./ IFT4109/Sem-1/M.C.A./2025-2... | Django bookstore setup | Students

University Portal Add Student Search students Search

Students

No students found. Add one.



Safari File Edit View History Bookmarks Window Help Wed Nov 12 12:02 AM

127.0.0.1

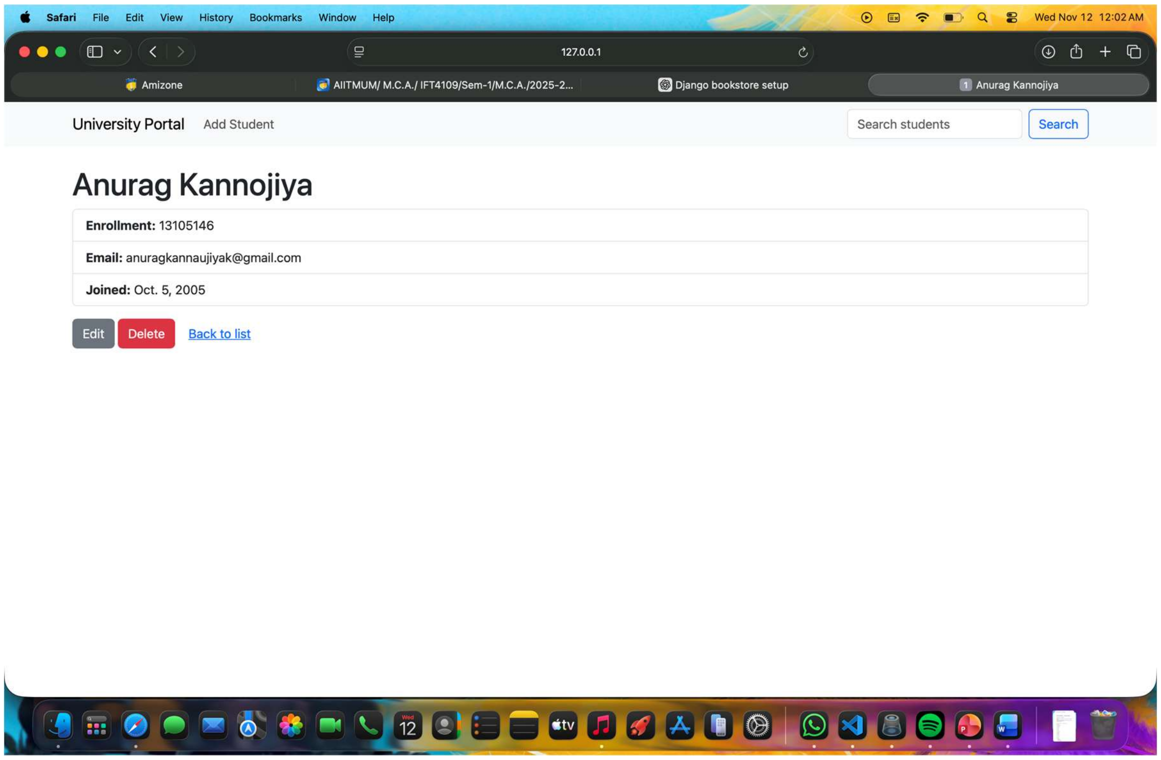
Amizone | AIITMUM/ M.C.A./ IFT4109/Sem-1/M.C.A./2025-2... | Django bookstore setup | Students

University Portal Add Student Search students Search

Students

#	Name	Enrollment	Email	Joined	Actions
1	Anurag Kannojiya	13105146	anuragkannaujiyak@gmail.com	Oct. 5, 2005	View Edit Delete

Previous Page 1 of 1 Next



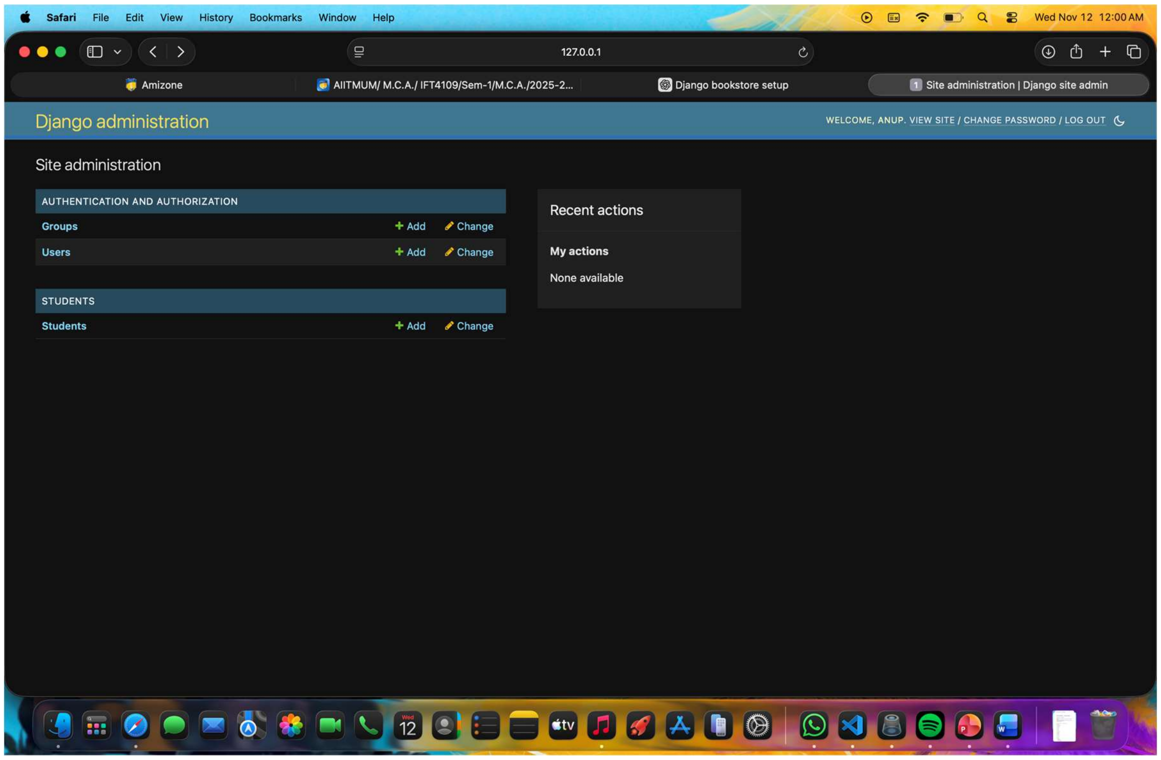
Anurag Kannojiya

Enrollment: 13105146

Email: anuragkannaujiyak@gmail.com

Joined: Oct. 5, 2005

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


Safari File Edit View History Bookmarks Window Help
Wed Nov 12 12:00 AM
127.0.0.1
Amizone
AIITMUM/ M.C.A./ IFT4109/Sem-1/M.C.A./2025-2...
Django bookstore setup
Site administration | Django site admin

Django administration

WELCOME, ANUP. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

STUDENTS		
Students	+ Add	Change

Recent actions

My actions

None available

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)