

Student Management

1. Project Setup

First, a virtual environment was set up. Then, after installing Django, the student_management project was created.

```
django-admin startproject student_management
```

2. Student Model

The Student model was created to store student data with the following fields:

- first_name
- last_name
- email
- date_of_birth
- enrollment_date
- grade

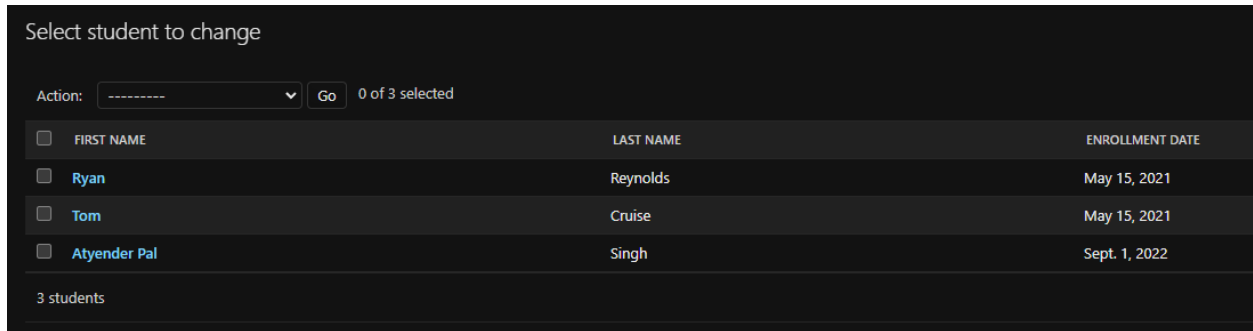
```
models.py

student_mgmt > students > models.py > ...
1  from django.db import models
2
3  # Create your models here.
4  class Student(models.Model):
5      first_name = models.CharField(max_length=100)
6      last_name = models.CharField(max_length=100)
7      email = models.EmailField()
8      date_of_birth = models.DateField()
9      enrollment_date = models.DateField()
10     grade = models.IntegerField()
11
12     def __str__(self):
13         return f"{self.first_name} {self.last_name}"
```

3. Admin Interface

Registered the Student model in the Django admin panel, and the list display was customized to show key fields like the first name, last name, and enrollment date.

```
class StudentAdmin(admin.ModelAdmin):  
    list_display = ['first_name', 'last_name', 'enrollment_date']
```



The screenshot shows the Django Admin interface for the 'Student' model. At the top, there is a header 'Select student to change'. Below it, there is an 'Action:' dropdown menu, a 'Go' button, and a status '0 of 3 selected'. The main part of the interface is a table with three columns: 'FIRST NAME', 'LAST NAME', and 'ENROLLMENT DATE'. The table contains three rows of student data. Each row has a checkbox in the first column. At the bottom of the table, it says '3 students'.

<input type="checkbox"/>	FIRST NAME	LAST NAME	ENROLLMENT DATE
<input type="checkbox"/>	Ryan	Reynolds	May 15, 2021
<input type="checkbox"/>	Tom	Cruise	May 15, 2021
<input type="checkbox"/>	Atyender Pal	Singh	Sept. 1, 2022

3 students

4. Views and Templates

Created views to display a list of students, view details of an individual student, add a new student, and edit information of an existing student.

Code from the Student List view:

```
def student_list(request):  
    students = Student.objects.all().order_by('first_name')  
    return render(request, 'students/student_list.html', {'students': students})
```

Code from the html file to display the Student List:

```
<ul>  
    {% for student in students %}  
        <li>  
            <a href="{% url 'student_detail' student.pk %}">{{ student.first_name }}  
            {{ student.last_name }}</a>  
        </li>  
    {% endfor %}  
</ul>
```

5. Forms

Used `ModelForm` in Django to create forms to add/edit student information.

Also included validation for the email field to be unique, and the grade field to ensure it's between 1 and 12.

```
class StudentForm(forms.ModelForm):
    class Meta:
        model = Student

        fields = ['first_name', 'last_name', 'email', 'date_of_birth',
                  'enrollment_date', 'grade']

    def clean_email(self):
        email = self.cleaned_data.get('email')

        student_id = self.instance.pk

        if Student.objects.filter(email=email).exclude(pk=student_id).exists():
            raise forms.ValidationError('A student with this email already exists.')

        return email

    def clean_grade(self):
        grade = self.cleaned_data.get('grade')

        if grade < 1 or grade > 12:
            raise forms.ValidationError('Grade must be between 1 and 12.')

        return grade
```

6. Authentication

Added a login system for authentication using Django's built-in system.

Authenticated users could add, edit, or delete students.

Users that are not logged in would see a message to login to unlock more features.

```
{% if user.is_authenticated %}
    <a href="{% url 'student_add' %}">Add New Student</a>
    <p>Logged in as {{ user.username }}.</p>
    <form method="post" action="{% url 'logout' %}">
```

```
{% csrf_token %}

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

</form>

{% else %}

  <a href="{% url 'login' %}">Login</a> to add more students!

{% endif %}
```

Student List

- [Atyender Pal Singh](#)
- [Tom Cruise](#)
- [Ryan Reynolds](#)

Page 1 of 1

[Login](#) to add more students!

Atyender Pal Singh

Email: atyenderpalsingh@gmail.com

Grade: 10

Date of Birth: Oct. 2, 2004

Enrollment Date: Sept. 1, 2022

[Login](#) to edit information!

Student List

- [Atyender Pal Singh](#)
- [Tom Cruise](#)
- [Ryan Reynolds](#)

Page 1 of 1

[Add New Student](#)

Logged in as atyen.

Atyender Pal Singh

Email: atyenderpalsingh@gmail.com

Grade: 10

Date of Birth: Oct. 2, 2004

Enrollment Date: Sept. 1, 2022

[Edit](#)

Logged in as atyen.

7. Search Functionality

A search bar was added to the student list page, allowing users to search for students by their first or last name.

```
def student_list(request):
    query = request.GET.get('q')
    if query:
        students = Student.objects.filter(first_name__icontains=query) |
Student.objects.filter(last_name__icontains=query)
    else:
        students = Student.objects.all()
    return render(request, 'students/student_list.html', {'page_obj': page_obj})
```

8. Pagination

Added pages to the Student List page using Paginator which is built into Django. Set a maximum of 10 students to be displayed on one page.

Also added a page indicator at the bottom of the list, showing the current and total pages, along with page navigation buttons.

```
paginator = Paginator(students, 10)
page_number = request.GET.get('page')
page_obj = paginator.get_page(page_number)
```

9. Error Handling and Validation

Added errors 404 and 500 pages. They are displayed to the user when the corresponding error occurs.

Also made sure proper validation was in place, like checking email formats and their uniqueness, empty fields, and incorrect grades or dates.

```
{% block content %}
<h1>Page Not Found (404)</h1>
<p>Sorry, the page you are looking for does not exist.</p>
<a href="{% url 'index' %}">Click here to go back to the home page</a>
{% endblock %}
```

Page Not Found (404)

Sorry, the page you are looking for does not exist.

[Click here to go back to the home page](#)

```
def clean_email(self):
    email = self.cleaned_data.get('email')
    student_id = self.instance.pk
    if Student.objects.filter(email=email).exclude(pk=student_id).exists():
        raise forms.ValidationError('A student with this email already exists.')
    return email
```

Styling

The main aspects of styling include centering everything, drawing a box around the content, and changing the background colours to follow a dark theme.

Also changed some text colours.

Added back and home buttons.

Here are some before and after screenshots:

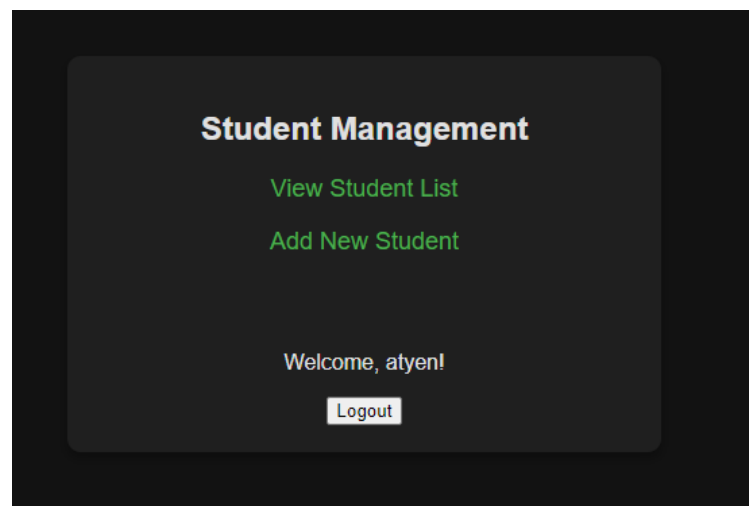
Student Management

[View Student List](#)

[Add New Student](#)

Welcome, atyen!

Logout



Student List

- [Atyender Pal Singh](#)
- [Tom Cruise](#)
- [Ryan Reynolds](#)

Page 1 of 1

[Add New Student](#)

Logged in as atyen.

Student List

Atyender Pal Singh

Ryan Reynolds

Tom Cruise

Page 1 of 1

[Add New Student](#)

Logged in as atyen.

[Back](#)

[Home](#)

Atyender Pal Singh

Email: atyenderpalsingh@gmail.com

Grade: 10

Date of Birth: Oct. 2, 2004

Enrollment Date: Sept. 1, 2022

[Edit](#)

Logged in as atyen.

Atyender Pal Singh

Email: atyenderpalsingh@gmail.com

Grade: 10

Date of Birth: Oct. 2, 2004

Enrollment Date: Sept. 1, 2022

[Edit](#)

Logged in as atyen.

[Back](#)

[Home](#)

Edit Student

First name:

Last name:

Email:

Date of birth:

Enrollment date:

Grade:

Logged in as atyen.

Add Student

First name:

Last name:

Email:

Date of birth:

Enrollment date:

Grade:

Logged in as atyen.

Edit Student

First name:

Last name:

Email:

Date of birth:

Enrollment date:

Grade:

Logged in as atyen.

[Back](#) [Home](#)

Add Student

First name:

Last name:

Email:

Date of birth:

Enrollment date:

Grade:

Logged in as atyen.

[Back](#) [Home](#)

Challenges Encountered

- **Email Duplication Check:** Initially, when editing a student's information, the system would raise an error because the email already existed (since it belonged to the student being edited). This was solved by adding a custom validation method in the form that excludes the current student's email from the uniqueness check.
- **Styling and Static Files:** When styling the pages, using an external CSS file led to problems. In the settings.py file, DEBUG was set to False to display the custom 404 and 500 error pages. But with DEBUG set to False, the external CSS file was not getting detected. After a lot of troubleshooting, I still couldn't get it to work and ended up styling the individual pages within each html file.