Django Classroom Exercise (Without Code Guide)

In this exercise, students will create a **Student Management System** using Django. This system will allow users to manage student records and courses. The key objectives are to understand Django models, views, templates, and the admin interface, without relying on a code guide.

Exercise Description: Student Management System

You will create a **Student Management System** where:

- Students can be enrolled in multiple courses.
- You will manage both students and courses through the **Django Admin**.
- The application should display a list of students and their enrolled courses on the front end.
- Use **Django Views**, **Templates**, **URL Routing**, and **Reversing URLs**.

Tasks:

1. Set Up the Django Project

- Start a new Django project called `student_system`.
- Create a new app called `students`.

---

2. Define Data Models

- Create models for **Student** and **Course**.
- A student can be enrolled in multiple courses (many-to-many relationship).
- Each course should have a name, and students should have a name, age, and grade.
- Add a `__str__()` method for both models to make them readable in the admin panel.

---

3. Set Up Migrations

- After defining your models, create and apply migrations to set up your database.

---

4. Register Models in Admin

- Register the **Student** and **Course** models in the Django admin.
- Ensure you can add, update, and delete students and courses through the Django admin interface.

---

5. Create Views

- Create a view that lists all students and their enrolled courses.
- Create a view that shows the details of a specific student, including the courses they are enrolled in.

---

6. Set Up URL Routing

- Define URL patterns for the following views:
  - A view that lists all students.
  - A view that shows a single student's details.

---

7. Use Templates

- Create templates to render the list of students and the details of a single student.
- Use **template inheritance** to create a base template that all other templates extend.
- In the student detail view, display the student's name, age, grade, and a list of the courses they are enrolled in.

---

8. Use URL Reversing

- In your templates, use the `{% url %}` template tag to link to the student detail pages.

---

9. Bonus:

- Use Django's **HTML escaping** feature by trying to inject HTML into the student's or course's name to see how Django automatically escapes it.
- Add Django **template filters** to format the output of dates or text in your templates.

What Students Should Submit:

- A fully functional Django project with a `student_system` project and `students` app.
- Working models for **Student** and **Course**.
- Admin access for managing students and courses.
- Views for listing students and showing individual student details.
- Templates with proper **URL routing** and **template inheritance**.

Learning Outcomes:

- Practice defining **Django models** and setting up **database migrations**.
- Understand how to use the **Django Admin** to manage models.
- Gain experience in creating and connecting **Django views**, **templates**, and **URL routes**.
- Practice using **Django template inheritance** and **URL reversing**.
- Learn how Django automatically handles **HTML escaping** to protect against XSS attacks.