Advanced Django Exercise: Student Registration System with Forms and CSRF

This advanced exercise will focus on building a **Student Registration System** that incorporates the newly introduced Django concepts such as **forms**, **CSRF protection**, **sessions**, and **HTTP methods (GET, POST)**. Students will also gain experience with **ORM**, **debug toolbars**, and learn how to implement **POST Redirect**.

Exercise Overview:

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

- Users can register new students using a form that submits data via the **POST** method.
- After successful registration, users will be redirected to a **success page** (using **POST-Redirect-GET**).
- The system will display a list of registered students.
- The system will handle CSRF protection, session management, and cookie-based authentication.

Tasks:

1. Set Up Django Project and App

- Create a new Django project called `student_registration`.
- Create a new app called `registration`.

---

2. Models

1. **Student**:
    - Fields: `name`, `email`, `age`, and `grade`.
    - Add a validation constraint that the student's `age` must be greater than 10.
2. **Registration**:
    - A model for tracking the registration time and session data.

---

3. Debug Toolbar

- Install and configure **Django Debug Toolbar** to monitor ORM queries and debug forms during development.

---

4. Building Forms with GET, POST, and CSRF

1. **Student Registration Form**:
    - Create a **Django Form** for registering a new student.
    - Ensure that the form uses CSRF protection.
    - Handle both **GET** and **POST** requests in the view.
    - On **GET**, display the form.
    - On **POST**, validate the data and save the student to the database.
    - After successful registration, implement **POST Redirect** to avoid form resubmission issues.
2. **Form Fields**:
    - Ensure that the form includes fields for `name`, `email`, `age`, and `grade`.
3. **CSRF Support**:
    - Ensure that CSRF tokens are included in the form to protect against CSRF attacks.

---

5. Using Cookies and Sessions

1. **Set Cookies**:
    - After a student is registered, set a cookie with the student's name.
    - Display the student's name in a welcome message using the cookie.
2. **Session Management**:
    - Use **Django sessions** to track whether the user has successfully submitted the form.
    - Display a message like "Welcome back!" when a user revisits the page within the same session.

---

6. Views and HTTP Methods

1. **GET**:
    - Display the registration form.
2. **POST**:
    - Handle form submissions.
    - Validate the form data, display error messages for invalid data, and save valid data.
    - Implement **POST Redirect** to avoid accidental resubmissions.
3. **Success Page**:
    - After the successful registration, redirect to a success page that displays a thank you message.

---

7. Templates

1. **Form Template**:
    - Build a template for the registration form using Django's template language.
    - Use Django template inheritance to extend a base layout.
    - Ensure the form uses `{% csrf_token %}` for CSRF protection.
2. **Success Page Template**:
    - Display a "Registration successful!" message with the name of the student that was registered.
    - Include a "Register another student" button that redirects back to the registration form.

---

8. Bonus Challenges

1. **Using ORM (Basic - Advanced)**:
    - Use Django ORM to display the list of students on the registration success page.
    - Implement filtering or ordering on the list of students (e.g., show students ordered by registration date or grade).
2. **CSRF Failure View**:
    - Create a custom **CSRF failure view** that renders a more user-friendly error message when CSRF validation fails.
3. **Session Expiration**:
    - Implement session expiration and ensure that after a certain period of inactivity, users are logged out automatically.

What Students Should Submit:

- A fully functional Django project with the `student_registration` project and `registration` app.
- Working models for **Student** and **Registration**.
- Correct implementation of forms using **GET** and **POST** methods.
- Proper use of **CSRF protection** in forms.
- Cookies set after successful registration, with session tracking.
- Redirect functionality after successful form submission (POST-Redirect-GET).

Learning Outcomes:

- Understand how to build and manage **HTML forms** in Django using **GET** and **POST** requests.
- Learn how to handle **Cross-Site Request Forgery (CSRF)** protection in forms.
- Implement the **POST-Redirect-GET** pattern to avoid form resubmission issues.
- Gain experience with **Django sessions** and **cookies** to manage user data.
- Practice using Django **ORM** for basic to advanced database queries and operations.
- Set up and use **Django Debug Toolbar** for debugging ORM queries and form data.

This exercise challenges students to go beyond basic form handling and use advanced concepts like **CSRF**, **sessions**, **cookies**, and **POST-Redirect-GET** to build a secure, functional system.