🧾 Case Study: Online Feedback Collection System

🎯 Objective:

An educational institution needs an online system to collect **student feedback** on courses or instructors. Students will fill out a form, and the data will be processed by a backend Java servlet hosted on a Tomcat 10+ server using Jakarta EE 9+ APIs.

📘 Scenario Description:

The institute offers various technical and non-technical courses. They want to capture feedback from students after course completion.
The feedback form includes fields like:

- Student Name
- Email Address
- Course Attended
- Feedback or Suggestions

The institution also wants to:

- Display a confirmation page showing submitted data.
- Optionally store this feedback in a database for future reference.
- Track if a student has already submitted feedback using session/cookie-based mechanisms.

🏗️ System Design Overview:

1. Frontend (User Interface):

- A JSP (or HTML) page with a form that asks students to enter their feedback.
- The form uses HTTP POST to submit data to the server.

2. Servlet Processing (Backend):

- A Java servlet receives the form data through a POST request.
- The servlet:
    - Extracts parameters (e.g., name, email, feedback).
    - Validates the input (e.g., non-empty fields).
    - Generates a response page thanking the student and echoing back the submitted data.

- Optionally, stores the feedback in a database or writes it to a file.

3. Optional Enhancements:

- Use **cookies** to track if a student has already submitted feedback.
- Use **HTTP sessions** to temporarily store user data across requests.
- Redirect to different pages based on whether the feedback was already submitted.

🔧 Tools & Technologies:

- **Jakarta EE 9+ (Servlet 5.0)** — For writing the HttpServlet
- **Apache Tomcat 10+** — Web server and servlet container
- **JSP/HTML** — Frontend form
- **Eclipse IDE / IntelliJ** — Development environment
- **Maven** or manual `.war` deployment — For packaging and deployment

🚀 Workflow:

1. **Student accesses** the feedback form via a browser (`index.jsp` or `feedback.html`).
2. **Student submits** the form after filling out the details.
3. **Servlet receives** the POST request, processes data, and optionally stores it.
4. **Response page** is generated by the servlet, confirming submission.
5. **Cookie or session tracking** prevents duplicate submissions.

📋 Key Servlet Concepts Demonstrated:

- `HttpServlet` usage (`doPost()` method)
- Request parameter extraction (`getParameter`)
- Response generation using `PrintWriter` or JSP forwarding
- `@WebServlet` annotation (alternative to web.xml mapping)
- Session and cookie handling (optional advanced part)
- Deployment on Tomcat 10+ with Jakarta namespace