# Assessment: Java Servlet and Spring Boot REST API Development

**Objective:** This assessment aims to evaluate students' understanding of Java Servlets and Spring Boot applications for building RESTful APIs. Students will be required to create a simple servlet for handling requests and a Spring Boot application exposing REST endpoints.

## Part 1: Java Servlet Implementation

**Task:**

1. Create a Java Servlet named ProxySystem that listens on the /proxy-system endpoint.
2. The servlet should respond with a simple text message: “System reached” and should land to the default swagger ui of your main dockerized application (eg swagger-ui/index.html)
3. Deploy the servlet in a Java EE container (Tomcat). – No Docker is neede for this part, it is a native installation

**Requirements:**

- Use **javax.servlet.http.HttpServlet.**
- **Override** the doGet method to send the response.
- Configure the servlet in web.xml or use @WebServlet annotation.
- Run and test the servlet using a web browser or Postman.

---

## Part 2: Spring Boot REST API Implementation

**Task:**

1. Create a Spring Boot application named ProductService.
2. Implement a ProductController with the following REST API endpoints:
    - GET /products - Returns a list of products.
    - GET /products/{id} - Returns a product by its ID.
    - GET /products/{name} - Returns a product by its name.
    - POST /products - Adds a new product.
    - PUT /products/{id} - Updates an existing product.
    - DELETE /products/{id} - Deletes a product by its ID.
3. Use an external PostgreSQL database required (schema and initial data will be provided)

**Requirements:**

- Use Spring Boot with @RestController and @RequestMapping.
- Implement Product model class with id, name, and price fields.
- Use a simple list to store and manage products.
- Test API endpoints using Postman or cURL.
- Create respectively services, repository classes
- Provide minimum security at your end points (using Spring Boot Security)
- Dockerize everything: your application, your database and add a PgAdmin microservice

**Submission Guidelines:**

- Provide Extensive inline code documentation and respectively readme.md files. (with instructions on how to run both applications)
- Submit the Java Servlet source code along with web.xml (if used).
- Submit the Spring Boot project as a zipped file. Include your Dockerfile, docker-compose.yaml etc.
- Provide sample API requests and responses.

**Evaluation Criteria:**

| **Criteria** | **Max Points** |
|---|---|
| Correct servlet implementation | 20 |
| Working Spring Boot REST APIs | 30 |
| Code structure and readability | 20 |
| Proper use of annotations and conventions | 20 |
| API testing and sample responses | 10 |
| **Total** | **100** |

---

This assessment will help students understand the basics of Java EE servlets and Spring Boot REST APIs while practicing real-world development scenarios.

**SQL Schema**

```
CREATE TABLE products (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    price DECIMAL(10, 2) NOT NULL
);

INSERT INTO products (name, price) VALUES
('Laptop', 1200.00),
('Smartphone', 799.99),
('Tablet', 450.00),
('Monitor', 199.99),
('Keyboard', 49.99);
```